Temporal Difference Rewards for End-to-end Vision-based Active Robot Tracking using Deep Reinforcement Learning

Pavlos Tiritiris
Dept. of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
pavlosti@csd.auth.gr

Nikolaos Passalis
Dept. of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
passalis@csd.auth.gr

Anastasios Tefas
Dept. of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
tefas@csd.auth.gr

***Abstract*—Object tracking allows for localizing moving objects in sequences of frames providing detailed information regarding the trajectory of objects that appear in a scene. In this paper, we study active object tracking, where a tracker receives an input visual observation and directly outputs the most appropriate control actions in order to follow and keep the target in its field of view, unifying in this way the task of visual tracking and control. This is in contrast with conventional tracking approaches, as typically developed by the computer vision community, where the problem of detecting the tracked object in a frame is decoupled from the problem of controlling the camera and/or the robot to follow the object. Deep Reinforcement Learning (DLR) methods hold the credentials for overcoming these issues, since they allow for tackling both problems, i.e., detecting the tracked object and providing control commands, at the same time. However, DRL algorithms require a significantly different methodology for training compared to traditional computer vision models, e.g., they rely on dynamic simulations for training instead of static datasets, while they are often notoriously difficult to converge, often requiring reward shaping approaches for increasing convergence speed and stability. The main contribution of this paper is a DRL, vision-based active tracking method, along with an appropriately designed reward shaping approach for active tracking problems. The developed methods are evaluated using a state-of-the-art robotics simulator, demonstrating good generalization on various dynamic trajectories of moving objects under a wide range of different setups.**

***Index Terms*—Active Tracking, Deep Reinforcement Learning, Reward Shaping, Webots**

I. Introduction

In recent years, object tracking has shown a significant growth and is used in a wide range of applications [1], [2], including but not limited to robotics applications [3] and human computer interaction [4]. Object tracking aims to localize a moving object in a stream of frames, enabling us to keep detailed information regarding the *trajectory* of the corresponding object through time. This information can be used either for extracting higher level information regarding the qualities and behavior the tracked object, or for controlling various parameters of the camera used for acquiring the video stream. The latter is especially important in various applications, such as autonomous cinematography [5], or even just ensuring that an appropriate view of the object of interest has been obtained, e.g., in surveillance applications [6], [7].

It is worth noting that the goal in many tracking applications is *control* [8], instead of tracking *per se*, leading to *passive tracking* approaches, where tracking is disconnected from the actual task at hand. On the other hand, the emergence of powerful Deep Reinforcement Learning (DRL) approaches [9], [10], enabled the development of *active tracking* approaches [11], [12]. Such approaches are capable of learning end-to-end control policies, directly from raw RGB input, without any intermediate pre-processing step. In this way, the developed algorithms can be directly optimized for the task at hand, without involving separate intermediate tasks. At the same time, this unified approach also holds the credentials for providing less complex and more robust systems [12].

Despite their apparent advantages, little work has been done so far for active tracking approaches [11], [12], since active tracking requires using advanced and realistic simulation environment for simulating the effects of various control commands instead of relying on static datasets. At the same time, active tracking typically relies on reward signals for the training process instead of ground truth bounding boxes that are usually used in passive tracking approaches. However, defining the appropriate reward functions for such tasks is not trivial, since there are many alternative ways to formulate the goal of the system [13]. This is in contrast with other DRL applications, such as games [14], where the reward function is intrinsic to the problem. Indeed, the way that the reward function is defined can have a significant effect on the behavior of the resulting DRL model, especially for complex control tasks [15]. At the same time, providing additional rewards, in the form of *reward shaping* [16]–[18], can often allow for further increasing the stability of the training process, along with its convergence speed.

The contribution of this work is two-fold. First, we develop and evaluate an active tracking simulation environment, demonstrating that active tracking methods can be indeed trained in an end-to-end fashion operating *directly* on raw RGB inputs. To this end, a realistic robot simulation environment

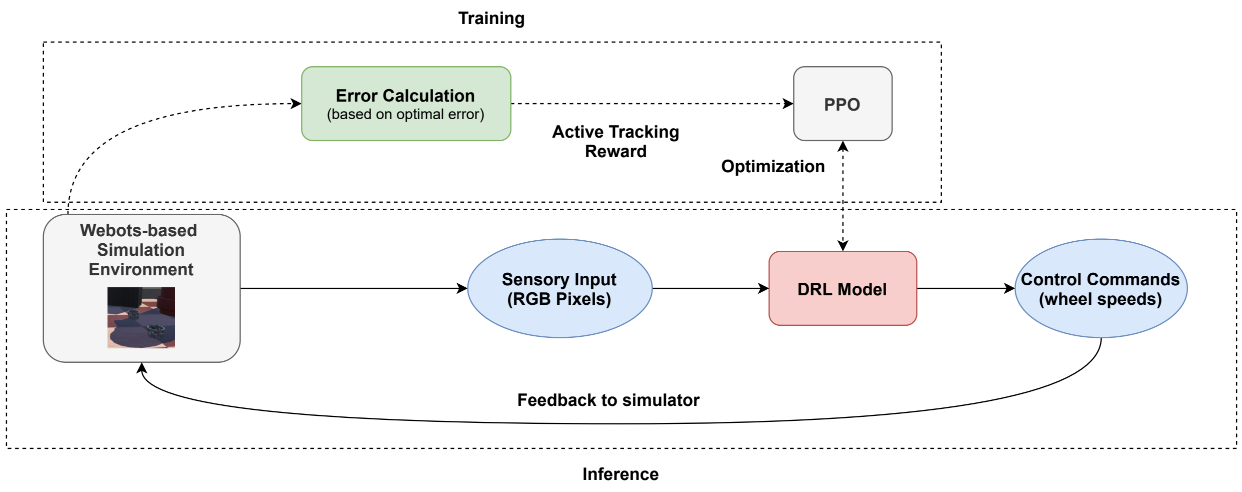

Fig. 1. Overview of the proposed DRL-based active tracking setup. First, the agent (DRL Model) is trained using the developed Webots-based simulation environment. Then, the trained agent is evaluated on a number of different scenarios (described in Section III).

was constructed using the Webots simulator [19], while the proposed method was evaluated using a real robot, the e-puck robot [20], and two different setups, involving different movement patterns of the robots. Furthermore, two different control scenarios were evaluated, corresponding to the two main approaches that are typically used for control: a) control using discrete actions/steps and b) control using continuous action spaces. Second, a simple, yet effective temporal difference-based reward function is introduced and evaluated for active tracking, improving the tracking error and allowing for directly performing control based on a tracking objective. Indeed, it is demonstrated, through extensive experiments, that the employed reward function can indeed lead to improvements in tracking accuracy, under a wide range of different setups.

The rest of the paper is structured as follows. First, background information and the proposed method are presented in Section II. Then, the experimental evaluation is proposed in Section III. Finally, conclusions are drawn in Section IV.

II. Proposed Method

The proposed approach is provided in this Section. First, we briefly introduce the developed simulation environment. Then, the employed DRL method is provided, along with the network architectures used for training the corresponding agents. Finally, three different reward functions for active tracking are introduced and discussed. An overview of the proposed approach for developing DRL agents for active tracking is provided in Fig. 1.

A. Simulation Environment and DRL agent

The simulation environment was built using the Webots simulator [19], which is an open source highly realistic robot simulator. The developed simulation environment contains two GCtronic e-puck robots [20], with the first one being the tracked target and the second one carrying an RGB camera (64×64 pixels) and aiming to track the first robot. E-puck robot moves by appropriately setting the velocity of its two wheels. Two different kinds of agents were employed in this paper: a) discrete action space agents and b) continuous action space agents. For agents that work with discrete actions, there are $2k + 1$ possible velocity values within the range $[-maxSpeed, +maxSpeed]$, splited by $2k$ equally spaced partitions. Therefore, the total number of actions for the agent is $(2k + 1)^2$, given that both wheels of the robot must be controlled. On the other hand, for continuous agents, two real numbers, one for each wheel, are used. The output of the agent is constrained between the minimum and the maximum speed supported by the robot, ensuring that all control actions will be within the hardware capabilities of the robots.

For training the agent, a state-of-the-art DRL optimization method, the Proximal Policy Optimization (PPO) [9], was employed. An actor-critic architecture was used to this end, where the actor is responsible for selecting the next action, while the critic was used for estimating the value of each state. PPO relies on the *advantage* for each performed action a when the agent is at state s:

$$A_\theta(s, a) = Q_\theta(s_t) - \sum_{t=0}^{T-1} \gamma^t r_t, \tag{1}$$

where $Q_\theta(s_t)$ denotes the critic's network estimated value, γ is the discount factor and r_t is the obtained reward at time t for a total number of T time-steps. To ensure the stability of the optimization process, PPO uses a probability ratio between new and old policy. Clipping this ratio, as proposed in [9], improves the behavior of the training process. Finally, note that in training phase PPO collects samples from the older policy using importance sampling.

The architecture used for both the actor and critic is depicted in Fig. 2. Three 2D convolutional layers were used, followed by three fully connected layers. The ReLU activation function was used after each layer [21], while average pooling

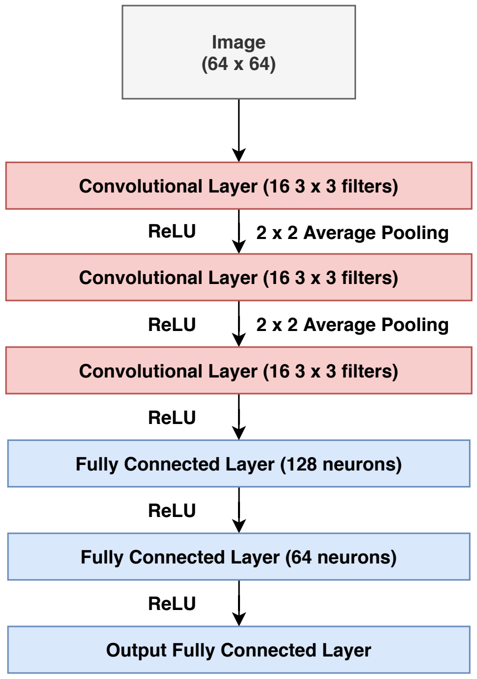
Image
(64 x 64)
Convolutional Layer (16 3 x 3 filters)
ReLU
2 x 2 Average Pooling
Convolutional Layer (16 3 x 3 filters)
ReLU
2 x 2 Average Pooling
Convolutional Layer (16 3 x 3 filters)
ReLU
Fully Connected Layer (128 neurons)
ReLU
Fully Connected Layer (64 neurons)
ReLU
Output Fully Connected Layer

Fig. 2. Network architecture used for the actor and critic models

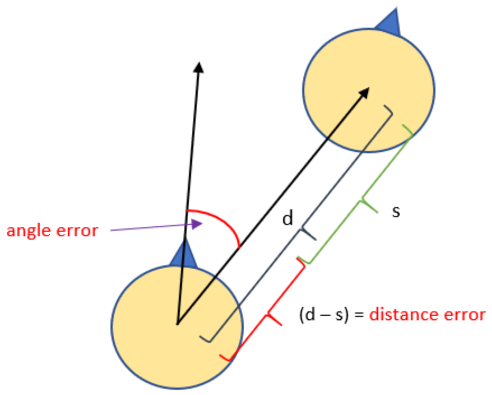

Fig. 3. Distance and angular errors

was employed instead of max pooling to avoid discarding potentially useful information. For discrete agents, the final layer of the actor has the same number of neurons as the number of possible actions, i.e., $(2k+1)^2$, while for continuous agents two output neurons are used (one for controlling the speed of each wheel). In the latter case, the output of the network is passed through the *tanh* activation, to ensure that it will range between -1 (maximum speed backward) and 1 (maximum speed forward). When the network outputs a value of 0, then the corresponding wheel stays stationary.

B. Reward

The RL agents must learn to follow the moving target as close as possible over a safe distance, while the camera looking at the center of the target. To this end, we define the distance error as:

$$e_d(t) = |d(t) - s|, \tag{2}$$

where $d(t)$ is the distance between the robots at time t and s is a predefined distance that must be kept from the robot (safe distance). Also, we define the angular error $e_a(t)$ at time t as the absolute value of the angle between the optimal look-at vector of the camera and the speed vector of the target robot, as shown in Fig. 3. A separate reward is calculated for each of these two errors, i.e., $r_d(t)$ and $r_a(t)$ respectively, while the total reward is calculated as the sum between the corresponding rewards:

$$r(t) = r_d(t) + r_a(t) \tag{3}$$

For defining these two individual rewards, i.e., $r_d(t)$ and $r_a(t)$, that contribute to the final reward provided in (3) several methods can be considered. Perhaps the simplest one is to directly use the negative of error, i.e.,

$$r(t) = -error(t), \tag{4}$$

where $r(t)$ refers to either $r_d(t)$ and $r_a(t)$ (according to the value used in place of $error(t)$), aiming to directly minimize the control error. However, as we experimentally demonstrate in Section III, this naive way of defining the reward leads to suboptimal results. Recent works in vision-based DRL control proposed to use hint-based rewards inspired by reward shaping methods [15], [22], i.e.,

$$r(t) = \begin{cases} c_g, & error(t) < margin \\ c_m, & error(t) < error(t-1) \\ c_p, & \text{otherwise} \end{cases}, \tag{5}$$

where c_g corresponds to the reward for achieving the desired target, c_m to the reward for moving towards a direction that reduces the error , while c_p corresponds to the penalty the agent receives in any other case (its absolute value must be larger than c_g to avoid oscillations). We set these parameters to $c_g = 1$, $c_m = 0.5$ and $c_p = -1.2$ following the protocols proposed in the literature and experiments conducted to select the optimal parameters. However, setting these parameters can require a significant amount of effort, involving repeated experiments to determine the optimal values. Therefore, to overcome this limitation, in this work we propose using a simpler, yet more effective approach for calculating the reward based on the *temporal difference* between two subsequent error values, i.e.,

$$r(t) = error(t-1) - error(t). \tag{6}$$

This approach does not require any kind of additional fine-tuning, while it leads to more accurate control, as we demonstrate in Section III. Note that the first and third reward functions are directly linked to the (unnormalized) values of

TABLE I
HYPERPARAMETER USED FOR ALL THE CONDUCTED EXPERIMENTS

Clip parameter (PPO)	0.2
Number of update iterations (PPO)	10
Gradient norm clipping (max)	0.5
Batch size	64
Actor / Critic learning rate	0.0001 / 0.0003
Number of episodes	100

the error, therefore min-max normalization is required after the end of each episode, to ensure the stability of the optimization process. This is not necessary for the hint-based reward, since the values of this reward are already bounded.

III. EXPERIMENTAL EVALUATION

The different agents and reward functions are evaluated under different setups in this Section. The hyperparameters used for all the conducted experiments are summarized in Table I. The number of update iterations refers to the number of gradient descent steps performed after each simulation episode, while the gradient norm clipping refers to the clipping of the training gradients that are larger than the specific threshold, which improves training stability. The former was set to 10, while the later to 0.5 The safe distance was set to $s = 0.2$m (meters), while the margin for the hint-based reward function was set to 0.05.

The developed agents were evaluated in two different setups:

1) *random movement*, where the velocity of both wheels of the front robot is selected randomly (therefore the first robot moves on a non-straight trajectory). The velocities remain the same during the entire episode. This setup was used both for training and evaluation of the trained agents.
2) *random movement with velocity changing*, where the same setup as before is used, but the speed of the wheels changes every 200 steps. As a result, the trajectory of the first robot changes several times within the same episode, requiring the tracking robot to promptly adjust in order to avoid losing the target robot. This setup was used for evaluating the agents trained with the first setup.

Each episode consists of 1000 steps, while an episode ends early when the tracking robot loses its target. For evaluation we report two different metrics:

1) *average distance error* ("Distance error"), which measures whether the agents keeps the desired distance from the tracked robot, and
2) *average control steps per episode* ("Steps per episode"), which measures the ability of the agent to track the front robot (note that an episode ends when the tracking robot loses its target).

First, the proposed method was evaluated using the first setup (random movement evaluation). The evaluation results for the discrete agent are reported in Table II, while for the continuous agent in Table III. The best discount factor was selected through validation experiments for all the evaluated reward functions ($\gamma = \{0.01, 0.25, 0.5, 0.75, 0.99\}$). First, note that for all the evaluated reward functions, the agents are indeed successfully trained, since they solve almost all the test episodes perfectly (the average number of steps is larger than 950 out of a maximum of 1000). At the same time, the temporal difference reward seems to lead to the overall best distance error (both for the discrete and continuous agent). However, it leads to slightly worse behavior for the discrete agent, failing to correctly track the target robot in a few cases, since the average number of steps per episode is reduced from 1000 to 969.

However, the opposite behavior is observed in the second evaluation setup reported in Table IV, where the proposed reward leads to both the best distance error, as well as leads to never loosing the target robot for all the evaluated episodes. The same results are achieved for the hint-based reward, even though hint-based rewards seem to always lead to higher distance error. This can be explained, since hint-based rewards are disconnected from the actual value of the error. Even

TABLE II
EVALUATION SETUP 1: DISCRETE AGENT

Reward	Distance error	Steps per episode
Negative of error	0.014	1000
Hint-based	0.054	969
Proposed	**0.009**	969

TABLE III
EVALUATION SETUP 1: CONTINUOUS AGENT

Reward	Distance error	Steps per episode
Negative of error	0.02	954
Hint-based	0.03	966
Proposed	**0.01**	1000

TABLE IV
EVALUATION SETUP 2: DISCRETE AGENT

Reward	Distance error	Steps per episode
Negative of error	0.016	985
Hint-based	0.035	1000
Proposed	**0.007**	1000

TABLE V
EVALUATION SETUP 2: CONTINUOUS AGENT

Reward	Distance error	Steps per episode
Negative of error	0.014	1000
Hint-based	0.030	974
Proposed	**0.006**	1000

TABLE VI
EVALUATION SETUP 2 (WITH OBSTACLES): CONTINUOUS AGENT

Reward	**Distance error**	**Steps per episode**
Negative of error	0.037	961
Hint-based	0.072	853
Proposed	**0.007**	965

though in previous works this has been shown to improve the training stability when combined with Q-learning based algorithms [15], this reward function does not seem to lead to similar improvements for the problem at hand. The ability of the proposed reward function to improve the distance error is again validated using the second evaluation setup using continuous action spaces (Table V).

Finally, we also evaluated the ability of the trained agents to work on an even more challenging environment by including additional obstacles, in the second setup. The evaluation results for the continuous agent are reported in Table VI. Again, it is confirmed that the proposed reward leads to the best results for the continuous agents, both regarding the distance error and the ability of the agent to keep the target in view, despite the presence of obstacles.

IV. CONCLUSIONS

In this paper a deep reinforcement learning-based approach for solving active tracking problems was presented. The proposed method can be trained in end-to-end fashion, while operating directly on raw RGB inputs without requiring solving any intermediate tracking problem, like most of the existing passive tracking methods. At the same time, a simple, yet effective reward function for active tracking problems was introduced and demonstrated that it can lead to improved tracking performance, under two different evaluation setups and DRL agents using a realistic simulation environment developed using the Webots simulator and a model of a real robot, the e-puck robot. The results provided in this paper pave the way for developing more sophisticated active tracking methods, ranging from methods that can track a wide variety of objects, to methods that can control simultaneously both the robot and the camera, choosing the most appropriate action to be performed, increasing the perception accuracy.

Acknowledgments: This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR). This publication reflects the authors views only. The European Commission is not responsible for any use that may be made of the information it contains

REFERENCES

[1] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Computing Surveys*, vol. 38, no. 4, pp. 13–58, 2006.

[2] P. Li, D. Wang, L. Wang, and H. Lu, "Deep visual tracking: Review and experimental comparison," *Pattern Recognition*, vol. 76, pp. 323–338, 2018.

[3] C. T. Chou, J.-Y. Li, M.-F. Chang, and L. C. Fu, "Multi-robot cooperation based human tracking system using laser range finder," in *Proceesings of the IEEE International Conference on Robotics and Automation*, 2011, pp. 532–537.

[4] M. Abbaszadegan, S. Yaghoubi, and I. S. MacKenzie, "Trackmaze: A comparison of head-tracking, eye-tracking, and tilt as input methods for mobile games," in *Proceedings of the International Conference on Human-Computer Interaction*, 2018, pp. 393–405.

[5] J. Capitan, A. Torres-Gonzalez, and A. Ollero, "Autonomous cinematography with teams of drones [j]," in *Proceedings of the Workshop on Aerial Swarms. IEEE International Conference on Intelligent Robots and Systems*, vol. 1, 2019, pp. 1–3.

[6] Z. Pan, S. Liu, A. K. Sangaiah, and K. Muhammad, "Visual attention feature (vaf): a novel strategy for visual tracking based on cloud platform in intelligent surveillance systems," *Journal of Parallel and Distributed Computing*, vol. 120, pp. 182–194, 2018.

[7] G. Amato, F. Falchi, C. Gennaro, F. V. Massoli, N. Passalis, A. Tefas, A. Trivilini, and C. Vairo, "Face verification and recognition for digital forensics and information security," in *Proceedings of the International Symposium on Digital Forensics and Security (ISDFS)*, 2019, pp. 1–6.

[8] W. Ye, Z. Li, C. Yang, J. Sun, C.-Y. Su, and R. Lu, "Vision-based human tracking control of a wheeled inverted pendulum robot," *IEEE Transactions on Cybernetics*, vol. 46, no. 11, pp. 2423–2434, 2015.

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

[10] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Proceedings oft he AAAI Conference on Artificial Intelligence*, 2018.

[11] W. Luo, P. Sun, F. Zhong, W. Liu, T. Zhang, and Y. Wang, "End-to-end active object tracking via reinforcement learning," in *Proceedings of the International Conference on Machine Learning*, 2018.

[12] ——, "End-to-end active object tracking and its real-world deployment via reinforcement learning," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[13] R. Pocius, D. Isele, M. Roberts, and D. W. Aha, "Comparing reward shaping, visual hints, and curriculum learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[14] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of the International Conference on Machine Learning*, 2016, pp. 1928–1937.

[15] N. Passalis and A. Tefas, "Deep reinforcement learning for controlling frontal person close-up shooting," *Neurocomputing*, vol. 335, pp. 37–47, 2019.

[16] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proceedings of the International Conference on Machine Learning*, vol. 99, 1999, pp. 278–287.

[17] A. C. Tenorio-Gonzalez, E. F. Morales, and L. Villasenor-Pineda, "Dynamic reward shaping: training a robot by voice," in *Ibero-American conference on artificial intelligence*. Springer, 2010, pp. 483–492.

[18] S. M. Devlin and D. Kudenko, "Dynamic potential-based reward shaping," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, 2012, pp. 433–440.

[19] O. Michel, "Cyberbotics ltd. webots™: professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, p. 5, 2004.

[20] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli, "The e-puck, a robot designed for education in engineering," in *Proceedings of the Conference on Autonomous Robot Systems and Competitions*, vol. 1, 2009, pp. 59–65.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

[22] A. Tzimas, N. Passalis, and A. Tefas, "Leveraging deep reinforcement learning for active shooting under open-world setting," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, 2020, pp. 1–6.