AD-VAT+: An Asymmetric Dueling Mechanism for Learning and Understanding Visual Active Tracking

Fangwei Zhong, Peng Sun, Wenhan Luo, Tingyun Yan, and Yizhou Wang

Abstract—Visual Active Tracking (VAT) aims at following a target object by autonomously controlling the motion system of a tracker given visual observations. To learn a robust tracker for VAT, in this paper, we propose a novel adversarial reinforcement learning (RL) method which adopts an Asymmetric Dueling mechanism, referred to as AD-VAT. In the mechanism, the tracker and target, viewed as two learnable agents, are opponents and can mutually enhance each other during the dueling/competition: *i.e.*, the tracker intends to lockup the target, while the target tries to escape from the tracker. The dueling is asymmetric in that the target is additionally fed with the tracker's observation and action, and learns to predict the tracker's reward as an auxiliary task. Such an asymmetric dueling mechanism produces a stronger target, which in turn induces a more robust tracker. To improve the performance of the tracker in the case of challenging scenarios such as obstacles, we employ more advanced environment augmentation technique and two-stage training strategies, termed as AD-VAT+. For a better understanding of the asymmetric dueling mechanism, we also analyze the target's behaviors as the training proceeds and visualize the latent space of the tracker. The experimental results, in both 2D and 3D environments, demonstrate that the proposed method leads to a faster convergence in training and yields more robust tracking behaviors in different testing scenarios. Potential of the active tracker is also shown in real-world videos.

Index Terms—Active Object Tracking, Adversarial Training, Reinforcement Learning.

1 Introduction

Visual Active Tracking (VAT) aims at following a target object by autonomously controlling the motion system of a tracker given visual observations. VAT is demanded in many real-world applications such as autonomous vehicle fleet (*e.g.*, a slave-vehicle should follow a master-vehicle ahead), service robots and drones (*e.g.*, a drone is required to follow a person when recording a video). To accomplish the VAT task, one typically needs to perform a sequence of tasks such as recognition, localization, motion prediction, and camera control. However, conventional visual tracking *e.g.* [1], [2], [3], [4], [5], [6] aims to solely propose a 2D bounding box of the target frame by frame, and does not actively take into consideration the camera-control. Thus, compared to the problem of "passive" tracking, VAT is more practical and challenging.

With the advancement of deep reinforcement learning *e.g.* [7], [8], [9], [10], training an end-to-end deep neural network via reinforcement learning for VAT is shown to be feasible [11]. The authors learn a policy that maps raw-pixel observation to control signal straightly with a Conv-LSTM network. Not only can it save the effort in tuning an extra camera controller, but also does it outperform the conventional methods where the passive tracker is equipped with a hand-engineered camera-control module.

- *Fangwei Zhong, Tingyun Yan, and Yizhou Wang are with the Nat'l Eng. Lab. for Video Technology, Key Lab. of Machine Perception (MoE), Computer Science Dept., Peking University, Beijing, 100871, P.R. China. E-mail: {zfw, yanty18, yizhou.wang}@pku.edu.cn*
- *Peng Sun and Wenhan Luo are with Tencent AI Lab, Shenzhen 518057, P.R. China. E-mail: {pengsun000, whluo.china}@gmail.com*
- *Yizhou Wang is also with Peng Cheng Lab and Deepwise AI Lab.*

However, the performance of such a deep reinforcement learning based tracker is still limited by the training methods. Due to the "trial-and-error" nature of reinforcement learning, it is infeasible to directly train the tracker in real world. Alternatively, virtual environments are always utilized to generate sufficient data for training without tedious human labeling. Nevertheless, to deploy the trained tracker in the real world, one has to overcome the gap between virtual and real environments. One solution can be building a great numbers of high-fidelity environments [12]. However, it is expensive and tedious to build such environments for VAT. Both the visual rendering (illumination, texture, *etc.*) and the physical properties should be carefully designed to emulate the real world. Suppose we carry out VAT where the target is a pedestrian. To build the environment, one has to not only model the human's appearance, but also design physical rules and the pedestrian's trajectory so that it moves naturally like a human beings. Recently, Luo *et al.* [11] tried to overcome the virtual-real gap by applying the so-called environment augmentation technique. They diversify the visual appearance by changing the placement of the background objects and by flipping left-right the screen frame. However, they neglect another important factor, that is, *the motion of the target* for VAT task. Intuitively, the complexity and diversity of the target motion in training will impact the generalization of the tracker. For example, if the target only moves forward during training, the tracker may over fit to move straightly and fail to correctly react to other motion patterns, like a sharp turn.

In this work, we propose a novel adversarial RL method for learning VAT, referred to as AD-VAT (*Asymmetric Dueling mechanism for learning Visual Active Tracking*). In the

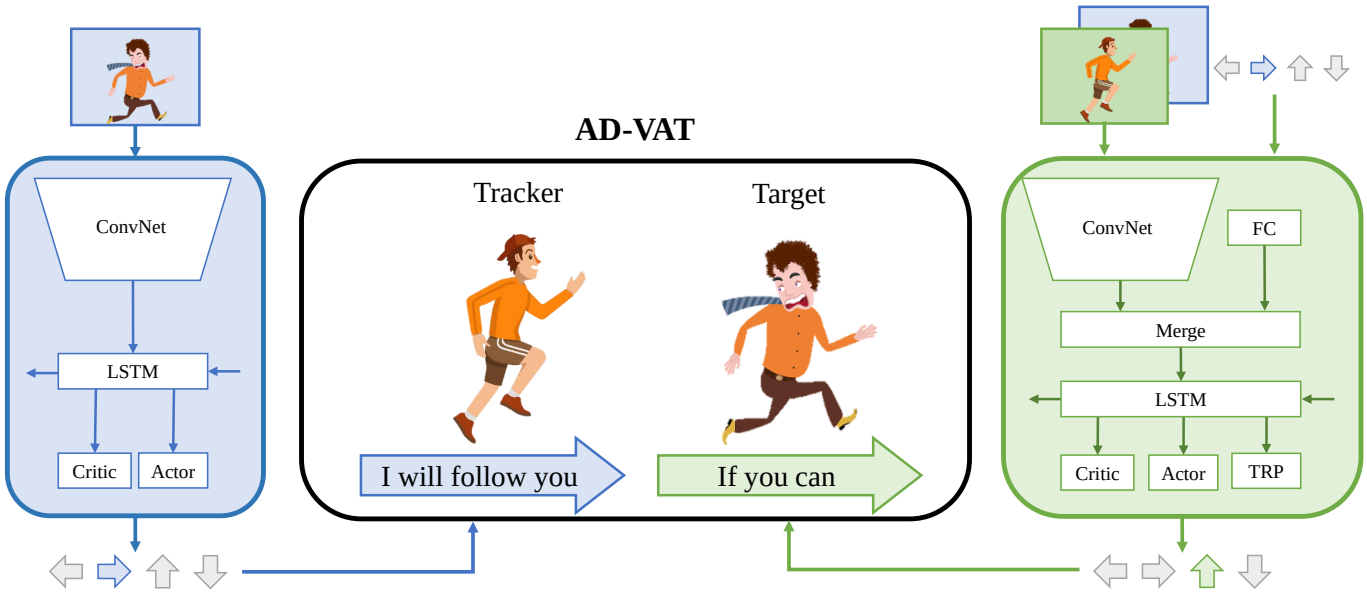

Fig. 1. An overview of AD-VAT. The tracker and target are "dueling". The tracker intends to lockup the target, while the target tries to escape from the tracker. They both use a Conv-LSTM network to encode the input and output the action. Differently, the target is additionally fed with the tracker's observation and action, and learns to predict the trackers reward as an auxiliary task. Note that TRP is Tracker Reward Predictor. The blue and green represent the component of tracker and target, respectively.

mechanism, the tracker and the target object, viewed as two learnable agents, are opponents and can mutually enhance during dueling/competition (see Fig. 1). As the training proceeds, the environments of AD-VAT naturally compose a curriculum, because the tracker is more likely to compete with a target at a level of appropriate difficulty when both agents are becoming stronger simultaneously. When exploring the escape policy, the target consequently generates various trajectories to train the tracker. By the dueling/adversarial mechanism, the target is encouraged to discover the weakness of the tracker more often, which could serve as a kind of "weakness-finding" that makes the tracker be more robust.

However, in practice, directly using RL method for training VAT is unstable and slow to converge. To address these issues, we devise two components in AD-VAT: the "partial zero-sum reward" (PZR) and the "tracker-aware model" (TAM) for target. PZR is a hybrid reward structure. It only encourages the tracker-target competition, a zero-sum game, when they are reasonably close; whereas beyond the zero-sum zone, it is a non-zero-sum game, in which the target is penalized for running too far to track. Such reward structure is designed to avoid an observed phenomenon, that when the target quickly learns to be far away from the tracker while the tracker has no chance to see the target any more and henceforth gets plateaus during training. TAM is proposed to help the target learn the optimal policy to escape, via getting and learning more information about the tracker. In TAM, besides the target's own observation, the observation and actions of the tracker are also fed to the escaping policy network. In addition, to shape a better representation about the tracker, we add an auxiliary task for the target by forcing it to learn to predict the immediate reward of the tracker. We argue that through such an "asymmetric dueling" mechanism we are able to learn a stronger target, which in turn yields a more robust tracker ultimately. By means of PZR and TAM, AD-VAT [13] is shown to rapidly learn a robust tracker in the scenarios of no or few occlusions.

However, more difficulties will be encountered in complicated scenarios. For example, the tracker will suffer from partial observation caused by the environmental factors including object occlusion, illumination change, *etc*. It hurts the tracking precision, and even fails the tracking. The most prominent factor probably is the occlusion due to the obstacles in environment, which can cause a situation where the target completely disappears in tracker's view-field. In this case, the tracker is difficult to infer the target's position solely from the visual observation and thus unable to make reasonable decision for a successful tracking. Such an incompetence would limit the application of AD-VAT in various real-world scenarios. It is thus necessary yet challenging for a tracker to learn how to perform camera control in order to overcome/avoid occlusion and recover tracking when losing the target in its view-field.

To address the issues, in this study we attempt to improve AD-VAT in regards of both the environment setting and the way of training, which we referred to as AD-VAT+. First, a more advanced environment augmentation is adopted, where we generate obstacles of random colors, sizes and appearance textures and place them in random locations in the environment. This treatment produces more complicated environments, and significantly improves the potential room for learning the tracking (for tracker) or escaping (for target) policies. Second, we propose a two-stage training. Specifically, in Stage-I we still rely on the PZR for a fast learning, but in Stage-II we gradually shrink the penalty factor and reduce it to a pure zero-sum reward. This way, the Stage-II training will lead to a more intense duelling between tracker and target, hence encouraging both to learn more profound polices. For example, we observe that the target

can learn to hide behind an obstacle, while the tracker learns to "find" target by walking around an obstacle. We argue that the two-stage training can unleash the power of the randomized obstacles and fully utilize the potentials of such an advanced environment augmentation. With a pure PZR (*i.e.*, only performing in Stage-I), we did not observe the emergence of those complex behaviors described above.

The experiment is conducted in various 2D and 3D environments for further studying AD-VAT+. The 2D environment is a matrix map where obstacles are randomly placed. In the 2D environments, we evaluate and quantify the effectiveness of our approach in an ideal condition, lacking of noise in observation and action. We also conduct an ablation study to show the effectiveness of each contributed component in AD-VAT+. The 3D environments are built on Unreal Engine 4, a popular game engine for building high-fidelity environments. We build a simple environment augmented room for training, where the texture of the background/players and the illumination are randomized. We also extend the room via randomly placing obstacles with randomized size, shape and texture in the room to further demonstrate the effectiveness of the AD-VAT+ in complex and challenging environments. Evaluation in three realistic scenarios demonstrate that the tracker trained in AD-VAT+ is capable of generalizing to high-fidelity environments even it is trained in a simple environment. We also test the tracker in real-world video clips, showing the potential of transferring it to real-world scenarios. Furthermore, to better understand AD-VAT+, we also use a number of well-controlled synthetic videos to analyze the latent space of the tracker, showing that the AD-VAT+ tracker learned a good representation, which is sensitive to motion but robust to appearance.

The contributions of our work can be summarized as follows:

- We propose a novel Adversarial Reinforcement Learning method for VAT task, *i.e.*, the Asymmetric Dueling mechanism (AD-VAT). In AD-VAT, the target learns to generate diverse trajectories when competing with the tracker, which in turn helps train a more robust tracker.
- We provide two techniques to guarantee an efficient yet effective AD-VAT. 1) A partial zero-sum reward structure, which significantly stabilizes the training. 2) A tracker-aware model for the target, which yields better escaping policy and consequently better tracking policy.
- We improve AD-VAT to handle more complicated scenarios and develop AD-VAT+, which is trained in augmented environments with randomized obstacles. More importantly, to endow a successful training, we develop a two-stage training method.
- We further analyze our proposed mechanism in terms of the target's behavior during training and testing, the feature of the learned latent space, which helps to better understand the mechanism. Its potential of transferring to the real-world scenarios is also verified.

This study extends the previous conference paper [13] in several aspects. Several improvements have been proposed to further enhance the robustness of the tracker in challenging cases, *e.g.*, when target is occluded by obstacles. Specifically, 1) We extend the environment augmentation method to diversify the training environment by randomly placing obstacles with randomized shape, size, and textures. 2) We develop a two-stage training strategy to enable the tracker to learn a robuster tracking policy from a stronger target. 3) Additionally, we also further evaluate and analyze our proposed mechanism by: a) understanding the training process via visualizing the distributions of the target position in different training stages. b) taking an adversarial testing to further evaluate the robustness of the trackers. and, c) understanding the representation for tracking via visualizing the t-SNE projection of the latent space. 4) We demonstrate the generalization of the tracker on the real-world video clips.

2 Related Work

2.1 Active Object Tracking

As described above, active object tracking deals with object tracking and camera control at the same time. This problem attracts less attention compared with traditional object tracking (or visual object tracking) [14]. In general, this problem could be addressed in a two-step manner or in an end-to-end manner. In the two-step solution, traditional object tracking and camera control are conducted sequentially to obtain tracking results and manipulate camera. Great progress has been achieved in traditional object tracking in recent decades [1], [2], [3], [4], [5], [6]. Thus one can utilize successful visual tracking algorithms [15], [16], [17] to accomplish the passive tracking task. According to the tracking results, camera control module could be developed to actively follow a target. For example, in [18] a two-stage method is proposed to handle robot control by motion detection and motion tracking. Kim *et al.* [19] detect moving objects and track them using an active camera with pan/tilt/zoom. In [20], two modules, a perception module and a control policy module, are learned separately to train an agent accomplishing both an obstacle avoidance task and a target following task.

Admittedly, tracking algorithms have been successful while still not perfect. Camera control based on tracking results is challenging due to factors such as the unknown correspondence between image space camera parameter space. Additionally, joint tuning of visual tracking and camera control is expensive and encounters the problem of trial-and-errors in real world. In end-to-end methods, direct mapping between raw input frame and camera action is established. Thus the intermediate visual tracking results are not necessarily required. For instance, by reinforcement learning, camera is controlled by signal outputted from a Conv-LSTM network given raw input frame [11]. This end-to-end solution verifies the effectiveness, while not efficient enough to solve this problem. To improve its potential for generalization, environment augmentation is conducted in this work. However, the properties of target itself, like the path, motion pattern are fixed in their augmented environments, which is believed to limit the performance. This inspires us to resort to the idea of the dueling mechanism in this paper, *i.e.*, the target learns to get rid of the tracker by itself, guiding the learning of the tracker.

2.2 Adversarial Reinforcement Learning/Self-Play

Reinforcement Learning (RL) is a paradigm where an Agent interacts with an Environment and intends for maximizing the expected return from the environment by adopting a proper policy [7]. The environment dynamics is assumed to be Markovian, therefore the corresponding stochastic state transition is stationary. Thanks to this property, a popular RL methodology, called Model Free, arises, where the agent can learn its policy from the trajectories generated during the interaction without knowing the environment dynamics. Representative concrete methods include value iteration [21] and policy gradient [22].

Using an adversarial framework to improve the RL agent's robustness is not a new concept. In [23], they add nonrandom adversarial noise to state input for the purpose of altering or misdirecting policies. Roughly speaking, in previous methods the adversary is viewed as a ghost/virtual player, which is unseen and could only challenge the protagonist by adding noise in the observation [23], action [24] and the system dynamics [25], or by generating the goal/initial position for navigation task [26], [27]. In this paper, we design a two-agent no-cooperative game for VAT task, where the tracker intends to lockup the target, while the target tries to escape from the tracker. Unlike the aforementioned previous work, the adversary (target to be tracked) in AD-VAT is a physical player, which could fully control the movement of the target at any time step. In this paper, such two-physical-player-competition is referred to as "dueling". We argue that such a fully controllable opponent could bring to the protagonist more challenges during training and hence produces a more robust visual tracker.

In adversarial RL or Self-play [9], [28], [29], two agents are competitive during training, with the other agent being or not being the duplicate of the first agent. In this case, Nash Equilibrium (NE) [30] is a useful solution concept for characterizing the policies of the two agents. By definition, when accomplishing NE policies, each agent cannot acquire more expected return by unilaterally altering its own policy. In case the rewards sum to zero between the two agents, it becomes the so-called max-min game, where one agent's benefit always comes from the other's loss, thus the first agent should maximize the second agent's minimum return.

For two-agent zero-sum game, it seems problematic if each agent performs an independent RL, as the dynamics absorbed into the agent's environment becomes non-stationary due to the varying policy of the other agent. However, independent RL is reported to be effective in some applications, despite lacking theoretical justification [31]. Several alternatives to independent RL were developed in the literature. A conventional method is the Joint-Value [32], which extends the RL value function by defining it over the joint actions from all agents. However, it is inapplicable when the observation is partial and the common state is unavailable. Recently, an opponent pool sampling method is proposed and shown to be successful in several non-trivial applications [29], [33]. Lanctot *et al.* [34] have shown that such an opponent pool sampling actually approximates by Monte-Carlo sampling the so-called "fictitious self-play", which is a classical NE finding algorithm known to Computational Game Theory community for long [35]. In general, any no-regret learning (*e.g.*, fictitious self-play, CFR [36]) performed by each agent guarantees an NE finding algorithm. However, they are difficult to implement with Neural Networks, requiring non-standard machinery/widget to handle the terms or algorithmic processes arising in the particular no-regret learning [37], [38].

In this work, we decide to adopt independent RL approach due to its simplicity of implementation. Also, our empirical results are satisfactory when applying our proposed techniques described in Sec. 3.

3 METHODOLOGY

In this section, we introduce our proposed method: *Asymmetric Dueling mechanism for learning Visual Active Tracking (AD-VAT+)*. At first the proposed mechanism is formulated as a Partial Observable Two-Agent Game. Then we introduce the three key components in AD-VAT+: a hybrid reward structure, a tracker-aware target, and a two-stage training strategy.

3.1 Formulation

We adopt the Partial Observable Two-Agent Game settings [39], which extends the Markov Game [32] to partial observation. For the notations of our two-agent game, let subscript 1 denote the tracker (agent 1) and subscript 2 denote the target (agent 2). The game is governed by the tuple $< \mathcal{S}, \mathcal{O}_1, \mathcal{O}_2, \mathcal{A}_1, \mathcal{A}_2, r_1, r_2, \mathcal{P} >$, where $\mathcal{S}, \mathcal{O}, \mathcal{A}, r, \mathcal{P}$ denote state space, observation space, action space, reward function and environment state transition probability, respectively. Let subscript $t \in \{1, 2, ...\}$ denote the time step. In the case of partial observation, we have the observation $o_{1,t} = o_{1,t}(s_t, s_{t-1}, o_{t-1})$, where $o_t, o_{t-1} \in \mathcal{O}$, $s_t, s_{t-1} \in \mathcal{S}$. It reduces to $o_{1,t} = s_t$ in case of full observation. The counterpart notation $o_{2,t}$ is defined likewise. When the two agents take simultaneous actions $a_{1,t} \in \mathcal{A}_1, a_{2,t} \in \mathcal{A}_2$, the updated state s_{t+1} is drawn from the environment state transition probability $\mathcal{P}(\cdot|s_t, a_{1,t}, a_{2,t})$. Meanwhile, the two agents receive rewards $r_{1,t} = r_{1,t}(s_t, a_{1,t})$, $r_{2,t} = r_{2,t}(s_t, a_{2,t})$. The policy of the tracker, $\pi_1(a_{1,t}|o_{1,t})$, is a distribution over tracker action $a_{1,t}$ conditioned on its observation $o_{1,t}$. We rely on model-free independent Reinforcement Learning to learn π_1. Specifically, the policy takes as function approximator a Neural Network with parameter θ_1, written as

$$\pi_1(a_{1,t}|o_{1,t}; \theta_1). \tag{1}$$

Likewise, the policy of the target can be written as

$$\pi_2(a_{2,t}|o_{2,t}; \theta_2). \tag{2}$$

Note that we further extend the policy to a tracker-aware policy as Eq. (7). The tracker intends to maximize its expected return

$$\mathbb{E}_{\pi_1,\pi_2}\left[\sum_{t=1}^{T} r_{1,t}\right] \tag{3}$$

by learning the parameter θ_1, where T denotes the horizon length of an episode and $r_{1,t}$ is the immediate reward of the tracker at time step t. In contrast, the target tries to maximize

$$\mathbb{E}_{\pi_1,\pi_2}\left[\sum_{t=1}^{T} r_{2,t}\right] \tag{4}$$

by learning θ_2.

3.2 Reward Structure

The conventional adversarial methods [24], [29] usually formulate the policy learning as a zero-sum game. In the zero-sum game, the sum of the reward of each agent is always 0, $r_{1,t} + r_{2,t} = 0$. However, such a kind of zero-sum game is not suitable for the VAT task. Considering a case that, when the two opponents are too far to observe each other, their taken actions can hardly influence the observation of their opponents directly. In this case, the sampled latent MDP transition is usually meaningless and ineffective for improving the skill level of the agent, especially at the beginning of the learning. So constraining the competition in the observable range would make the learning more efficient. Motivated by this, we shape a partial zero-sum reward structure, which utilizes the zero-sum reward only when the target is observed by tracker, but gives additional penalties to the target when they are far. In the following, we will introduce the details of the partial zero-sum reward structure for visual active tracking.

Reward for tracker. The reward for tracker is similar to that in [11], composing of a positive constant and an error penalty term. Differently, we do not take the orientation discrepancy between the target and the tracker into consideration. Considering the model of the camera observation, we measure the relative position error based on a polar coordinate system, where the tracker is at the origin $(0, 0)$. In this tracker-centric coordinate system, the target's real and expected position are represented by (ρ_2, θ_2) and (ρ_2^*, θ_2^*), respectively. Note that ρ is the distance to the tracker, θ is the relative angle to the front of the tracker. With a slight abuse of notation, we can now write the reward function as

$$r_1 = A - \zeta \frac{|\rho_2 - \rho_2^*|}{\rho_{max}} - \xi \frac{|\theta_2 - \theta_2^*|}{\theta_{max}}, \tag{5}$$

here $A > 0$, $\zeta > 0$, $\xi \geq 0$ are tuning parameters, $\xi = 0$ in the 2D environment. We do not use the direction error as part of the penalty, for the reason that the observation is omnidirectional in the 2D environments. ρ_{max} is the max observable distance to the tracker. θ_{max} is the max view angle of the camera model, which equals to the Field of View (FoV). Besides, the reward is clipped to be in the range of $[-A, A]$ to avoid over punishment when the object is far away from the expected position.

Reward for target. The reward for the target object is closely related to the reward of the tracker, written as:

$$r_2 = -r_1 - \mu \cdot \max(\rho_2 - \rho_{max}, 0) - \nu \cdot \max(|\theta_2| - \frac{\theta_{max}}{2}, 0), \tag{6}$$

where r_1 is the reward of the tracker as defined in Eq. (5), $\mu > 0$, $\nu \geq 0$ are tuning parameters controlling the factor of each penalty term. ν is 0 in the 2D environment, as the angular penalty factor ξ in Eq. (5). The target is in the nearly observable range, where $\rho_2 < \rho_{max}$ and $|\theta_2| < \theta_{max}$. In the observable range, the reward function is simplified to $r_2 = -r_1$, which means that the target and tracker play a zero-sum game. When the target gets out of the observable range, the penalty term will take effect on the reward. The farther the target goes out of the range, the greater the penalty it gets. By applying this reward function, the optimal policy for the target we expect should be escaping and disappearing from the observable range of the tracker but keeping close to the edge of the range. r_2 is also clipped in the range of $[-A, A]$.

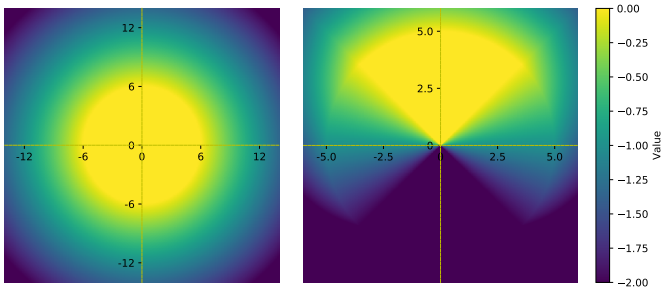

Fig. 2. Visualizing $r_1 + r_2$ as a heatmap in $x - y$ plane with the tracker in the image center. The yellow area indicates the zone playing zero-sum game. Left: the reward adopted in our 2D environment experiment. Right: the reward adopted in our 3D environment experiment.

Visualize the reward distribution. To help better understand the reward structure given in Eq. (5) and (6), we visualize the sum $r_1 + r_2$ as heatmap in $x - y$ plane. See Fig. 2.

For the 2D environment, the observations for both the tracker and target are bird-views. We want to penalize the case that the target gets too far away from the tracker. Therefore, the zero-sum area is a circle (Fig. 2, Left), where the tracker is in the centre. With the increasing of the distance, the penalty term in r^β (see Eq. (6)) starts taking effect on the sum. It causes the sum to reduce gradually until the target reaches the dark area, where $r_2 = -A$.

For the 3D environment, the observations for both the tracker and target are front-views. We want to penalize that the target gets too far away from the tracker or that the target cannot be seen by the tracker. Thus, the zero-sum area is a sector area (Fig. 2, Right), which approximately fits the Field of View (FoV) of the tracker's camera. Note that the FoV in our experiment is 90 degree. Both the relative angle θ and distance ρ contribute to the penalty term in r^β. Thus the sum decreases like a divergence sector.

3.3 Tracker-Aware Target

By tracker-awareness, the target would be "stronger" than the tracker, as it knows what the tracker knows. This idea manifests an ancient Chinese proverb, "know the enemy, know yourself, and in every battle you will be victorious", from the masterpiece Sun Tzu on the Art of War [40]. The conventional adversary usually uses only its own observation [29] or shares the same observation as the protagonist [24]. Recall the target policy $\pi_2(a_{2,t}|o_{2,t}; \theta_2)$ written as in Eq. (2). However, the imperfect/partial $o_{2,t}$ observation seriously degrades the performance of the adversary. Thus, we propose a "tracker-aware" model for the target. Besides the target's own observation, we additionally feed the observation and action from the tracker into the target network, in order to enrich the input information of the target. Moreover, we add an auxiliary task, which predicts the immediate reward of the tracker (see the TRP module in Fig. 1). This auxiliary task can be treated as a kind of "opponent modeling", and alleviates the difficulty in its

own policy learning. By doing so, we can write the output heads of such a "tracker-aware" policy as:

$$\pi_2\left(a_{2,t}, \hat{r}_{1,t} | o_{1,t}, a_{1,t}, o_{2,t}; \theta_2\right) \tag{7}$$

where $\hat{r}_{1,t}$ is the predicted immediate reward for the tracker and $o_{1,t}$, $a_{1,t}$ are respectively the observation and the action of the tracker. Empirical results show that the tracker-aware target yields more diversified escaping policy and finally helps producing a more robust tracker. Note that we cannot apply the trick for a tracker, as the tracker has to use its own observation during testing/deployment.

3.4 Two-Stage Training

The penalty item in the target's reward could constrain the tracker-target competition in a local range, where the target could be frequently observed. It could make the adversarial training converge efficiently, especially learning basic tracking skills. Because following a visible target is the most usual cases in visual active tracking. However, it also restricts the development of the adversarial behavior of the target, *e.g.*, learning to use the obstacles to get away as far as possible. The constrained behavior of the target leads to the limited capability of the tracker in challenging scenarios. To address this problem, we propose a two-stage training strategy to first learn to track in most of the common cases and then learn to track in difficult cases. The two-stage training strategy we take could make the tracker learn from an adversarial target with more suitable skill level, at different training stages. The idea, learning skills from a progressively more skillful adversary, could be regarded as a kind of curriculum learning.

Stage 1: learn to follow observable target. The agents are trained from scratch. At this stage, the target should focus on mining hard examples around the observable range, especially finding various examples about motions and appearances. To realize it, the penalties in Eq. (6) take effect on the target when it is out of the observable range ($\mu \geq 0, \nu \geq 0$). Thus, the tracker could efficiently learn a representation to track, from the cases produced by the target.

Stage 2: learn to overcome challenging cases. At this stage, the target is encouraged to derive a policy to get disappeared from the observable area, *e.g.*, using the obstacles to get rid. Meanwhile, the tracker has to be more capable of overcoming these challenging cases, *e.g.*, recovering from lost, and handling the occlusion. The factors of the penalties for target are adjusted to zero or a negative value ($\mu \leq 0, \nu \leq 0$), to encourage the target to escape. When the penalties are eliminated ($\mu = 0, \nu = 0$), the target and tracker play a conventional zero-sum game. When the factors are negative ($\mu < 0, \nu < 0$), the target will get additional positive reward, instead of penalty. This stage aims to fine-tune the agents trained at the first stage, so the learning rate usually is smaller than that in the first stage.

4 Experimental Setup

4.1 Environments

We build a number of 2D and 3D environments for training and testing. Although 2D environments exhibit unreality to some extent, they are ideal for evaluating and quantifying the effectiveness of each method, sidestepping the uncontrolled noise in observation and action. The 3D environments show high fidelity, aiming to mimic the real-world active tracking scenarios. We use 3D environments to demonstrate that our tracker, trained in a virtual environment, can transfer well to unseen environments even real-world scenarios.

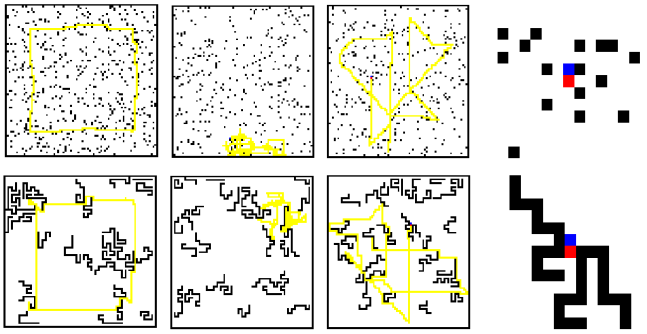

Fig. 3. Examples of the six testing environments. The top shows the examples of "block" map with the historical trajectories of three baselines (from left to right: RPF, Ram, Nav) and the tracker's observation (on the most right). The bottom is about the "maze" map.

4.1.1 2D Environments

In 2D Environments, maps are represented by a 80×80 matrix, where free space, obstacle, tracker, and target are denoted by 0, 1, 2, and 4, respectively. We generate the layout of the obstacles with two patterns, "maze" and "bloc" (see examples on Fig. 3). The agents are trained in the "block" maps only and tested in both kinds of maps. The observation of each agent is a 13×13 matrix around its own. The tracker's goal is placing the target as close as possible to the center of the observed matrix. During each episode, the tracker starts from a free space in the map randomly, and the target starts around the tracker in a 3×3 tracker-centric matrix. At each step, the agent could take an action to move towards one of four directions. The experiments in the 2D environments are dedicated to evaluate and quantify the effectiveness of our approach in ideal conditions.

4.1.2 3D Environments

The 3D environments show high fidelity, aiming to mimic the real-world active tracking scenarios. The 3D environments are built on the Unreal Engine, which could flexibly simulate a photo-realistic world. We employ UnrealCV [41], which provides convenient APIs, along with a wrapper [42] compatible with OpenAI Gym [43], for interactions between RL algorithms and the environment. The observation is an image of the first-person view of the world as seen by the agent. The action space is discrete with seven candidate motions, *move-forward*, *move-backward*, *turn-left*, *turn-right*, *turn-left-and-move-forward*, *turn-right-and-move-forward*, and *no-op*.

For training, we built a room equipped with two controllable players (target and tracker) and environment augmentation techniques (EA) [44], namely *EnvAug Room* (see the top row of Fig. 4). The environment augmentation techniques are employed to help agents learn better feature representation in terms of visual observation. The more

complex the training environment is, the stronger the RL agent would be. Motivated by this, we build a variant of *EnvAug Room* by randomly placing a number of objects with different shapes, sizes, and textures, namely *EnvAug+Obstacle Room* (see the middle row of the Fig. 4). Hence, the tracker and target could be further improved and some human-like behaviour emerge, *e.g.*, the target uses the interference of obstacles to get rid of the tracker, the tracker learns to infer the target's movement to track when the target is occluded by obstacles. Details of the two training environments are:

- EnvAug Room is a plain room comprised of floor and walls only, but the textures and illumination conditions are randomized. For the textures, we randomly choose pictures from a texture dataset [45] and place them on the surface of walls, floor, and the players. For the illumination condition, we randomize the intensity and color of each light source as well as each position and direction.
- EnvAug+Obstacle Room is built based on EnvAug Room, additionally placing a number of obstacles at random. Specifically, we randomly place 20 objects as obstacles in different locations, with various shapes, sizes, and textures. There are four candidate shapes to select, including cube, sphere, cylinder, and cone. The surface textures are also randomized, following the same rule as other objects. We reset all of the obstacles at the begin of each episode. Benefiting from the environment, the tracker could be more robust, especially in more complex scenarios.

To evaluate the generalization of the tracker, we build a set of realistic environments as testing environments. These environments mimic different real-world scenarios, including urban, villages, underground parking lots, and gardens. Details of the five testing environments are:

- *Urban City* is a high-fidelity street view of an urban city, including well-modeled buildings, streets, trees and transportation facilities. Besides, there are some puddles on the road, reflecting the objects and buildings.
- *Snow Village* is a flat land with a few trees, bushes and some cabins. The challenge in the environment is the falling snowflakes and halo caused by the sun would distract the tracker.
- *Parking Lot* is an underground parking lot with complex illumination condition. The lack of light source makes the illumination uneven, *i.e.*, some places are bright but the others are dark. Besides, the pillars may occlude the target to track.
- *Urban Road* is of similar the style to *UrbanCity*. But it is a long street view with a number of obstacles, including roadblocks, flowerbed, lamppost, and *etc.*
- *Garden* is a grassland with trees, fences and big stones. These objects frequently block the target, causing the tracker to fail to observe the target. The difficulty increases with the complex illumination, which makes the appearance of the scene vary in a wide range.

4.2 Baselines

We provide three kinds of rule-based targets as baselines to compare with, namely *Rectangular Path Follower (RPF)*, *Rambler (Ram)* and *Navigator (Nav)*.

- *RPF* walks along a fixed pre-defined path. In our experiment, the agent follows a rectangular route.
- *Ram* walks randomly without any destination like a human. Technically, it randomly samples actions from the action space and keeps executing the action n times, where n is also a random integer in the range of $(1, 10)$.
- *Nav* is a navigator, planning the shortest path to a goal using the Astar algorithm [46]. Thus, it could explore most of the free space in the map. We diversify the trajectory patterns by randomly sampling the goal coordinate from the free space.

In most cases, *Ram* prefers to walk around in a local area repeatedly. In contrast, *Nav* would like to explore the map globally, shown as the yellow trajectories in Fig. 3. Thus we regard trajectories from *Ram* as easier cases, and trajectories from *Nav* as more difficult cases for the tracker.

4.3 Evaluation Metric

Two metrics are employed for the experiments. Specifically, *Accumulated Reward* (AR) and *Episode Length* (EL) of each episode are calculated for quantitative evaluation. *AR* is a comprehensive metric, representing the tracker's capability about precision and robustness. It is effected by the immediate reward and the episode length. Immediate reward measures the goodness of tracking. EL roughly measures the duration of good tracking, because the episode is done when the target is lost for continuous 10 steps or the max episode length is reached.

4.4 Implementation Details

Each agent is trained by A3C [10], a commonly used reinforcement learning algorithm. The code for A3C is based on a pytorch implementation [47]. Multiple workers are running in parallel when training. Specifically, 16 workers are used in the 2D experiment, and 6 workers are used in the 3D experiment. Validation is performed in parallel and the best validation network model is applied to report performance in testing environments. Note that the validation environment is of the same settings as training, except that the target is controlled by a *Nav* agent. Compared with the *Ram* agent, the *Nav* agent is more challenging, thus is more suitable for validation.

Network Architecture. For the tracker, we follow the end-to-end Conv-LSTM network architecture as [11]. Differently, there is no fully-connected layer between the Conv-Net and the LSTM-Net in this study. The Conv-Net is a two-layer CNN for the 2D experiments and four-layer CNN for the 3D experiments. In the 3D experiments, the input color images are transformed to gray-scale and the pixel values are scaled to $[-1, 1]$. we also develop the same Conv-LSTM network architecture for the target, but different in the input and output, as shown in Fig. 1. The network parameters are updated with a shared Adam optimizer.

Fig. 4. The examples of the third-person views of the 3D environments used for training and testing. From top to bottom, they are EnvAug Room, EnvAug+Obstacle Room, and five realistic testing environments (from left to right: Urban City, Snow Village, Parking Lot, Urban Road, and Garden)

TABLE 1
The details of parameters of the rewards in the 2D and 3D experiments. Note that $A = 1$ in both cases.

	ζ	ξ	μ	ν	μ'	ν'	ρ_2^*	ρ_{max}	θ^*	θ_{max}
2D	2	0	1	0	-0.5	0	$0grid$	$6grid$	\	$360°$
3D	2	2	2	2	0	0	$2.5m$	$5.0m$	0	$90°$

Hyper Parameters. For the tracker, the learning rates δ_1 and δ_1' in 2D and 3D environments are 0.001 and 0.0001, respectively. The reward discount factor $\gamma = 0.9$, generalized advantage estimate parameter $\tau = 1.00$, and regularizer factor for tracker $\lambda_1 = 0.01$. Comparing to the tracker, a higher regularizer factor is used to encourage the target to explore, $\lambda_2 = 0.2$ in 2D and $\lambda_2' = 0.05$ in 3D. The more exploration taken by target, the more diverse the generated trajectories are. It is useful for the learning of the tracker. The parameters of the rewards are summaried in Table. 4.4

The parameter updating frequency n is 20, and the maximum global iterations for training is 150K in 2D and EnvAug Room, and 500K in EnvAug+Obstacle Room. For the two-stage training, there are some hyper parameters to be adjusted. At the first stage, we adopt the same hyper parameters as above, but take 80K iterations in 2D, and 300K in EnvAug+Obstacle Room. At the second training stage, the factors of the penalties in the target reward is adjusted, $\mu' = -0.5$ in 2D and $\mu' = \nu' = 0$ in EnvAug+Obstacle Room. Besides, the learning rates are smaller than those at the first stage, *i.e.*, $\delta_1 = 0.0003$ and $\delta_1' = 0.00003$.

5 Empirical Results

5.1 Results on the 2D Environment

Firstly, we quantitatively evaluate the performance of our approach in testing environments, comparing to the three baselines. Secondly, we conduct an ablation study to demonstrate the contribution of each proposed technique. Thirdly, we analyze the training process by visualizing the position distributions of the target during training. To further evaluate the robustness of the tracker, we conduct adversarial testing, using the tracker-aware target to attack the tracker.

5.1.1 Evaluation with Baselines

We evaluate the trackers, trained with different targets, in six testing environments. Considering the random seed of the environments, we conduct 100 runs in each and report the mean and standard deviation of AR and EL, shown in Table 2. The max episode length is 500, so the upper bound of EL is 500. Thus, when EL equals to 500, we could infer that the tracker performs robustly without losing the target. The average results in the bottom line represent the comprehensive capability of the trackers. As we can see from the table, for all the six testing environments AD-VAT+ learns a better policy in term of both mean reward and mean episode length. The quantitative results in the table suggest the effectiveness of our proposed method. Specifically, both our AD-VAT and AD-VAT+ methods significantly outperform all of the baselines regarding the average results. In *Block* map, our tracker obtains a comparable result to the tracker trained in the testing environment, *e.g.*the AD-VAT and *Ram* both obtain 363 in *Block-Ram*. In *Maze* map, the two AD-VAT trackers gain higher reward than baselines. These evidence indicates that the AD-VAT tracker is capable of generalizing to robustly track unseen targets in unseen maps. The two-stage training

TABLE 2
Evaluation in the 2D environments.

Environment		Accumulated Reward (AR)					Episode Length (EL)				
Map	Target	RPF	Ram	Nav	AD-VAT	AD-VAT+	RPF	Ram	Nav	AD-VAT	AD-VAT+
Maze	RPF	277±89	217±152	265±121	300±65	**310**±50	460±116	377±192	425±156	490±60	**494**±42
	Ram	268±110	347±34	317±63	**352**±25	351±21	419±148	498±23	480±80	**500**±0	**500**±0
	Nav	257±99	254±131	282±103	305±52	**316**±28	448±126	416±170	442±141	**493**±53	**500**±0
Block	RPF	296±84	183±167	212±151	**302**±58	294±60	476±99	327±208	352±199	**500**±0	496±43
	Ram	278±118	363±17	312±87	363±19	**367**±16	421±153	500±0	464±108	**500**±0	**500**±0
	Nav	260±111	220±158	260±135	319±30	**320±25**	438±140	356±210	406±183	**500**±0	**500**±0
Average		273±103	264±143	275±119	324±52	**326**±45	444±133	412±173	428±155.9	497±33	**498**±25

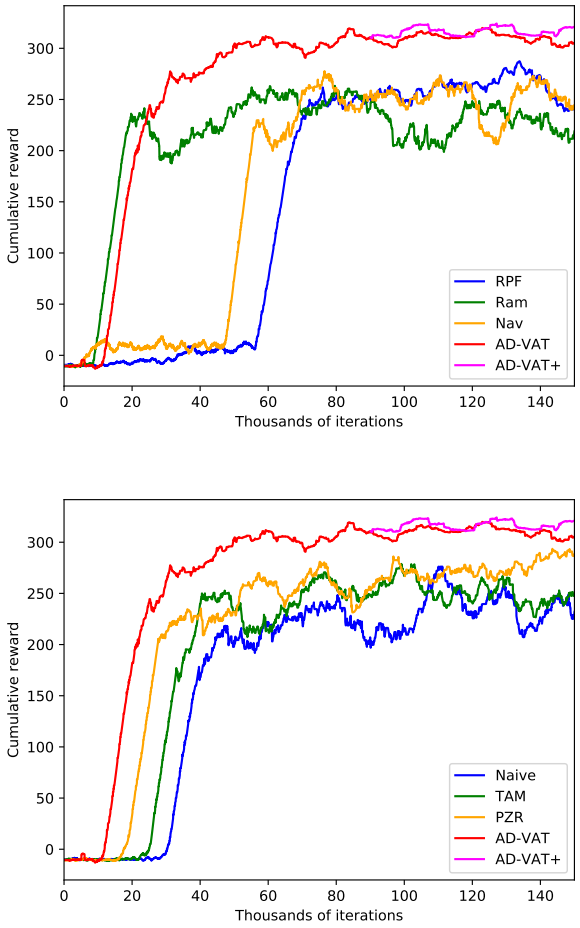

Fig. 5. The average cumulative reward of the tracker validated in the Block-Nav environment, as a function of iterations times. On the top, we compare AD-VAT to the baselines. On the bottom, we compare AD-VAT to its ablations.

of AD-VAT+ also leads to additional improvement. On the contrary, the performance of the baselines usually decreases in the case of unseen target and unseen map. For example, the *Ram* tracker performs well in its training environment *Block-Ram*, gaining reward 363 in average. But it gains only 183 in *Block-RPF*, much lower than the result of the tracker trained in *Block-RPF*, which obtains 296. Similarly, the results of *RPF* and *Nav* tracker also demonstrate that they fail to generalize well to most of unseen environments, *e.g.*, the *RPF* tracker get only 278 in *Block-Ram*. Compared with AD-VAT with the one-stage training, the AD-VAT+ tracker's ability is further enhanced by the two-stage training, especially in challenging environments (*Maze-Nav* and *Maze-RPF*).

We also report the mean of cumulative rewards (over the nearest 100 episodes) in the validation environment as a function of the training iterations, shown as the top sub-figure in Fig 5. It consistently shows the advantage of the proposed *AD-VAT+* in sample-efficiency and performance. We can see the curve of the AD-VAT+ tracker starts to rise rapidly after $10K$ iterations, which is close to the earliest one *Ram*. However, the mean of the cumulative reward of *Ram* is much lower than ours after convergence. Regarding the other baselines *RPF* and *Nav*, the mean of cumulative reward of them rises up slowly during the beginning $45K$ iterations. They also fail to exceed our tracker.

The impressive results of AD-VAT and AD-VAT+ in sample-efficiency and robust performance are mainly benefited from that the adversarial target could automatically produce curriculum under the dueling mechanism. We note that, at the beginning of learning, the adversarial target usually walks randomly around the start point, performing similar policy as *Ram*. Such a target is easier to be found and observed, even though the tracker is in exploration. Thus, the tracker could warm up faster. With the growth of the tracker, the target gradually explores other motion patterns that the tracker more likely to fail, to further reinforce the tracker. In Sec.5.1.3, we provide more evidences about the curriculum learning process via visualizing the position distribution during the training process.

5.1.2 Ablation Study

Since AD-VAT+ combines three individual ideas into a single mechanism, we conducted additional experiments to understand the contribution of the three techniques: partial zero-sum reward structure (PZR), tracker-aware model (TAM), and two-stage training strategy.

Firstly, We start from an intuitive idea that target only uses its own observation without any auxiliary tasks, guided by a zero-sum reward, *i.e.*, $r_{2,t} = -r_{1,t}$, named as *Naive*. Secondly, we add PZR and TAM on *Naive*, respectively. Thirdly, we combine both PZR and TAM and train the model by the one-stage (AD-VAT) and two-stage (AD-VAT+) strategies. We plot the learning curve of each tracker to show the effectiveness of these three techniques, shown as Fig. 5. These three techniques are important as they influence the natural curriculum for the tracker. From Fig. 5, we can see that PZR and TAM could improve the sample-efficiency and tracking performance, even each works separately. Since

the PZR performs better than TAM, we suspect that PZR plays a more important role than TAM. Moreover, when combining PZR and TAM, both sample-efficiency and the performance are boosted, comparing to TAM, PZR, and Naive. The second stage training is conducted after the first stage gets converged, so it plays a role in further improving the performance, without contributing to the training efficiency. In summary, all the ideas contribute to the overall performance improvement, and AD-VAT+ performs best when combining all of them.

5.1.3 Visualizing the Training Process

For a better understanding of the training process, we record trajectories of the target and the tracker during different training stages. Specifically, we have 6 stages, ranging from early training to late training. For each stage we record 100 episodes. For ease of visualization, we plot the distribution of the position of target in a tracker-centric coordinate system(see Fig. 6), instead of the trajectory itself. Such the distribution could help us intuitively understand the behavior of the target at different stage. For example, when the target is closely tracked by the tracker, the distribution is a simple dark cross symbol in the center. On the contrary, when the target frequently escapes along, the area it has passed becomes dark. In most cases, different escape behaviors correspond to different patterns of distribution.

At early training stages (see the left of Fig. 6), *AD-VAT* and *Ram* generate similar distributions, which means that target uniformly samples actions to walk around the tracker. The randomly-walking trajectories help the tracker observe the target's appearance in various positions,as a result of which more diverse experiences are sampled, accelerating the learning of the tracker at the early stage. It also explains why our approach achieves comparable sample efficiency to *Ram* at the beginning. In contrast, *Nav* and *RPF* target usually go along a straight line to the goal position, causing the target frequently appeared in some specific directions but rarely in other positions. The straight line paths usually make the tracker fail quickly at beginning, which is unsuited for the early tracker.

With the evolution of the tracker during training (from left to right), the target will gradually seek more difficult cases to defeat the tracker. In this regard, the *RPF*, *Ram*, and *Nav* target usually explores possible directions uniformly. The minor difference is that the *Nav* and *RPF* tend to explore the directional path, while the *Ram* explores randomly (see the right of Fig. 6). As for our AD-VAT method, however, the reinforced target could adapt to the capability of the tracker, resulting in different moving patterns at different stages. For example, the target tends to walk around the edge of observation at the second stage, but the center area at the fourth stage. Besides, it seems that the reinforced target could balance the two exploration modes of *Nav* and *Ram* naturally. Sometimes it tends to explore the map, and sometimes it duels with the tracker locally. By the dueling mechanism, the target could discover the weakness of the tracker more often, which seems to serve as a kind of importance sampling that enhances the tracker efficiently during training. Such a "weakness-finding" phase seems to be absent from the *Nav* and *Ram* algorithms. Even though the distributions of

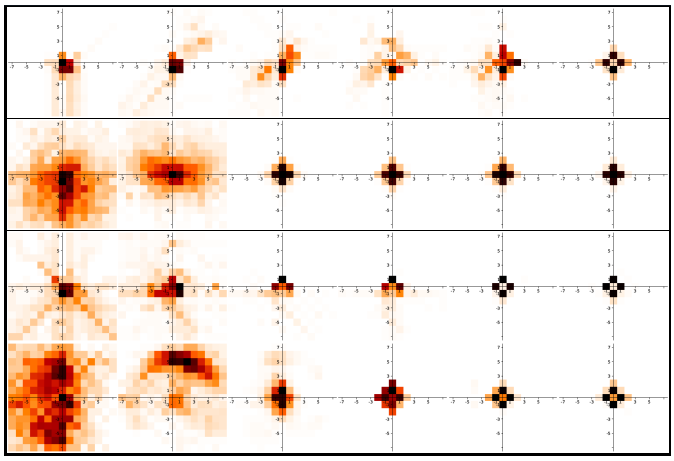

Fig. 6. Target position distributions in a *tracker*-centric coordinate system. Evolution from early training to late training is arranged from left to right. From top to bottom: *RPF*, *Ram*, *Nav*, *AD-VAT+*. The darker, the bigger the counts.

Nav and *Ram* seems impeccable at the end, the weakness is exposed during the adversarial testing in Sec. 5.1.4.

5.1.4 Evaluation with Adversarial Target

To verify the robustness of the tracker and the effectiveness of the adversarial target, we consider adversarial testing, using the learned adversarial target to attack each tracker. Even though the three rule-based targets have covered most of the common cases, they are still not the most difficult cases at all. The adversarial testing illustrates that, 1) the tracker learned by baseline method is fragile, and our tracker is more robust than them. 2) the target learned in AD-VAT+ is "smart" enough in downgrading the performance of trackers.

In details, the four trackers, trained in different ways, is dedicated to follow the adversarial target learned by AD-VAT+. The adversarial testing only uses the "block" map. Each tracker runs 100 episodes and reports the mean and standard deviation of AR and EL in Table.3. The AR and EL of all of trackers are much lower than the average result reported in Table. 2, which proves that the adversarial target is able to effectively expose the weakness of each tracker to make them get lower reward even fail. Compared with other trackers, the AD-VAT+ tracker could perform much better than other ones. Besides, we also notice that the AR decreases but EL is still comparable to the average results. It suggests that the target could enlarge the distance from the tracker but is difficult to get rid of it. It illustrates that the tracker is robust to make the target be observable in its view range, even though the target is aggressive.

We also visualize the relative position distribution of each tracker respectively, when they are attacked by the adversarial target. Shown as Fig. 7, we can see that the distributions of three baselines diffuse to different directions, which indicates ways the target escapes. This suggests that the target is capable of finding a way to escape based on the policy of the tracker during dueling and effectively expose the tracker's "weakness". Differently, when dueling with the AD-VAT+ tracker, the distribution converge around the center without diffusion. This indicates that the the AD-VAT+

TABLE 3
Using the adversarial target to attack each tracker. The best results are shown in bold.

Target	Tracker	AR	EL
AD-VAT+	RPF	240 ± 120	432 ± 156
	Ram	186 ± 167	383 ± 192
	Nav	222 ± 152	373 ± 196
	AD-VAT+	**284**±64	**496**±36

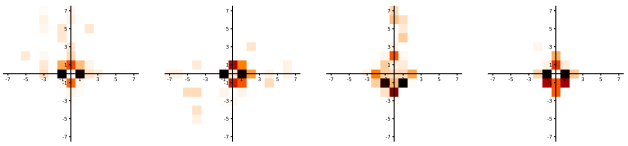

Fig. 7. The relative position distribution of the AD-VAT+ target while adversarial testing the trackers. From left to right: RPF, Ram, Nav, AD-VAT+.

tracker is robust to the adversarial target, and the target is tightly followed, hardly has chance to escape.

5.2 Comparing to Baselines in The 3D Environments

In the realistic 3D environment, we test the effectiveness of the model in unseen realistic scenarios, showing its potential for transferring to real-world scenario. In the EnvAug Room, we train four trackers with different targets (*RPF*, *Ram*, *Nav*, and *AD-VAT*) , respectively.

We evaluate the performance of our AD-VAT tracker compared to the three baselines in the validation and testing environments (EnvAug Room, Urban City, Snow Village, Parking Lot). Since the trajectories of the target used for testing are randomly generated, each tracker runs 100 episodes in each environment and we report the mean and standard variance of AR and EL in Table 4.

From the table, we can see that *AD-VAT* tracker outperforms the three baselines, and achieves a better policy in term of AR and EL in the four environments. Note that the *EnvAug Room* for validation is of the same settings as the training of *Nav* tracker. So the result on it demonstrates that the AD-VAT tracker is able to generalize to unseen motion of target, even outperform the tracker that trained with the testing target. The results on other environments illustrate that the most of trackers have the potential to transfer to realistic environments which are unseen while training. There is still a significant gap in their results, even though all of them are equipped with the Conv-LSTM network and trained with the environment augmentation. We believe that the environment augmentation method and the Conv-LSTM network are necessary to endow the trackers with the ability of transferring, but not sufficient. That is because we notice that the *RPF* tracker's performance is much lower than others. It illustrates that the diversity of the trajectories of the target is crucial for learning a robust tracker. The improvement achieved by *AD-VAT* verifies that the adversarial behavior of the *AD-VAT* target could significantly improve the capability of the tracker, especially in these challenging environments(Snow Village and Parking Lot). So we suspect that the two baseline methods(*Ram* and *Nav*) are not sufficiently effective when transferring to challenging environments, although they could diversify the trajectories

TABLE 4
Comparison with heuristic target. The best results are shown in bold.

Environment	Target	AR	EL
EnvAug Room	RPF	166 ± 118	360 ± 180
	Ram	273 ± 113	426 ± 155
	Nav	311 ± 122	426 ± 143
	AD-VAT	**345**±76	**474**±93
UrbanCity	RPF	76 ± 82	220 ± 137
	Ram	350 ± 22	**498 ± 25**
	Nav	345 ± 80	471 ± 97
	AD-VAT	**357**±60	483±73
SnowVillage	RPF	114 ± 104	283 ± 163
	Ram	203 ± 133	301 ± 178
	Nav	208 ± 133	316 ± 162
	AD-VAT	**256**±131	**356**±167
ParkingLot	RPF	35 ± 59	136 ± 98
	Ram	202 ± 127	317 ± 177
	Nav	188 ± 140	282 ± 186
	AD-VAT	**234**±129	**337**±174

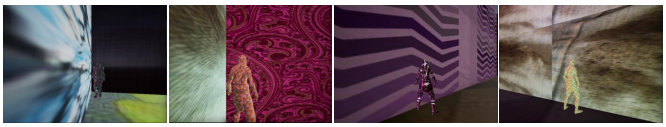

Fig. 8. The adversarial target in AD-VAT attempts to to fool the tracker by moving to the wall that is of similar texture to itself.

based on some rules. We infer that the target in *AD-VAT* could explore the environment more actively to discover more difficult cases. For example, in *EnvAug Room*, the target would prefer to move close to the wall which is of the similar texture to itself to fool the tracker (see Fig. 8). By competing with the target, the tracker consequently becomes stronger. Thanks to such hard example mining behavior, the tracker learned an appearance-invariant feature(See Sec.5.3.3).

5.3 Evaluation in The 3D Environments with Obstacles

In this section, we pay special attention to how to improve the ability of trackers to handle challenging cases, especially when the target is occluded by obstacles. Based on AD-VAT, we explore two ways to help the tracker further improve capabilities: using more advanced environment augmentation method and a two-stage training strategy. For training, we employ the *EnvAug Room + Obstacle* , which further augments the environment via randomly placing randomized obstacles. We also adopt the two-stage training strategy. At the first stage, we train the tracker and target, with the same hyper-parameters as in *EnvAug Room*. Differently, it takes 300K iterations, which is much higher than the previous experiment for the increased complexity of environment. At the second stage, we eliminate the penalty term in the target's reward ($\mu' = 0$, $\nu' = 0$), to encourage the target to perform competitive behavior in any case. We also adjust the learning rate δ' to 0.00003. The models are fine-tuned by 200K iterations.

5.3.1 Quantitative Evaluation

To evaluate the ability of overcoming occlusion, three realistic environments with obstacles are used for testing. We compare the performance of trackers trained by three protocols of AD-VAT. The quantitative results are reported in Table. 5. We use

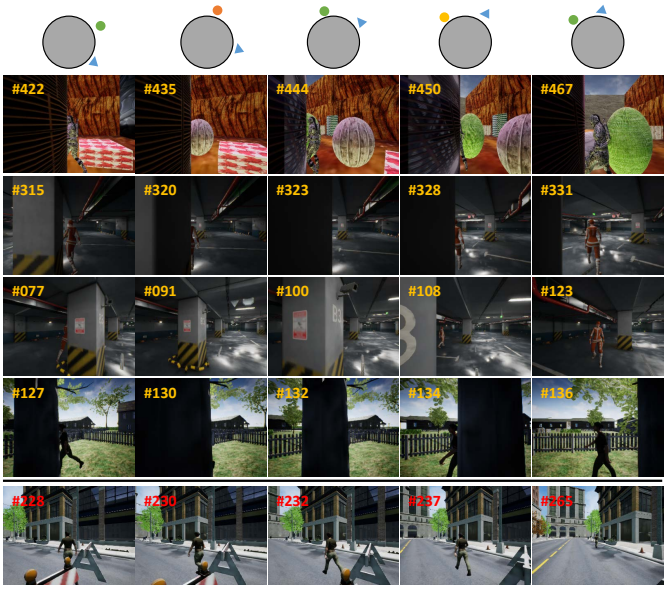

Fig. 9. Exemplar cases. The four sequences with yellow number are successful cases. The one with the red number is a failure case. The top row is a simple map of the first sequence to show the location of the obstacle, target, and tracker at each keyframe. The gray circle is the obstacle, the small circle is the target, and the small triangle is the tracker. The green, yellow and orange represent the occlusion state of the target: no occlusion, locally occluded, and fully occluded, respectively.

$*$ to mark the one that trained in *EnvAug Room*. The others that without $*$ are trained in EnvAug+Obstacle Room.

We can see that AD-VAT+ achieves the best performance in all of the environments. Comparing to both one-stage models trained in different environments (AD-VAT* and AD-VAT), it verifies the effectiveness of adding random obstacles to the environments while training. The improvements from AD-VAT to AD-VAT+ validate the necessity of the two-stage training strategy. These empirical results prove that AD-VAT not only works well in simple environments, but also applies to more complex/challenging environments. In addition, by refining the training strategy, the capability of the tracker can be further improved.

5.3.2 Exemplar Cases

Shown as Fig. 9, we select five exemplar cases to demonstrate how the tracker overcomes the occlusion and what kind of cases would fail the tracker.

The first sequence is an example from the EnvAug+Obstacle Room. On the top, We also plot the map of the obstacle, target, and tracker, corresponding to each keyframe. In this example, the target goes around a large cylinder. The tracker closely follows the target, even though the target is frequently occluded by the cylinder. A similar case also appears in a realistic environment, where the pillar makes the target disappeared from the view of the tracker. Meanwhile, the tracker also follows the path of target to recover from occlusion. Besides the behavior of path following, the tracker also tries to explore a new route to re-find the target, demonstrated in the third and fourth sequence. When the target goes through the back of a thin obstacle, the tracker intends to retreat to observe both sides of the obstacle, instead of bypassing the obstacle from behind. In the third sequence, the tracker moves backward and rotates to view both sides of the pillar when the target is occluded by the pillar. After a while, when it still does not see the target, it decides to actively go to another side of the pillar to search the target. In the fourth sequence, the tracker also moves backward and rotates to observe both sides of the tree and successfully recovers from the loss when the target appears from one side of the tree.

From the four successful examples, we suspect that our tracker learns to handle these challenging cases due to the following three aspects. 1) The tracker learns to predict the location of the target for making a better decision, thanks to the Conv-LSMT network structure. 2) The tracker tries to avoid the occurrence of occlusion by actively moving the camera to explore a short path to recover from the loss, benefited from the natural advantage of the active vision and reinforcement learning. 3) The EnvAug+Obstacle Room covers most of occlusion cases that occur in the realistic scenarios for learning, thanks to the enhanced environment augmentation technique.

The sequence at bottom is a typical failure case. The tracker fails to catch up the target but still looks at the target continuously. The main reason for the failure is that the tracker does not notice the existence of the short obstacles in the front. These obstacles do not occlude the target directly but obstruct the movement of the tracker. Potentially, we could extend the vision system or memory of the network to help the tracker be aware of these short obstacles, to overcome the case to further improve the system. It is an interesting future work but beyond the scope of this work.

TABLE 5
Results on the 3D environments with obstacles.

Environment	Protocol	AR	EL
EnvAug Room + Obstacle	AD-VAT*	151 ± 145	286 ± 162
	AD-VAT	281 ± 138	409 ± 142
	AD-VAT+	**318** ± 119	**441** ± 113
ParkingLot	AD-VAT*	183 ± 116	344 ± 153
	AD-VAT	284 ± 115	429 ± 128
	AD-VAT+	**319** ± 98	**454** ± 105
UrbanRoad	AD-VAT*	202 ± 122	350 ± 146
	AD-VAT	269 ± 112	426 ± 123
	AD-VAT+	**302** ± 102	**448** ± 102
Garden	AD-VAT*	38 ± 48	136 ± 72
	AD-VAT	193 ± 115	352 ± 146
	AD-VAT+	**231** ± 121	**377** ± 149

5.3.3 What is learned by AD-VAT?

To understand what the model has learned, we analyse the latent space learned by the Conv-LSTM of the tracker. We collect a set of videos and visualize the t-SNE embeddings of the latent feature. To better control and analyse the potential factors, we collect synthetic videos from EnvAug Room, instead of real-world videos. Specifically, we fix the camera to record the moving target, which walks from the same location (the image center) to a specific direction with the same speed. We select 8 typical directions, *e.g.*, forward, backward, left, right, forward-left, forward-right, backward-left, and backward-right. The target moves to each direction 10 times repeatedly, but with different appearance of itself and the background. Fig. 10 shows the t-SNE visualization [48] of high-dimensional latent vectors computed from observations

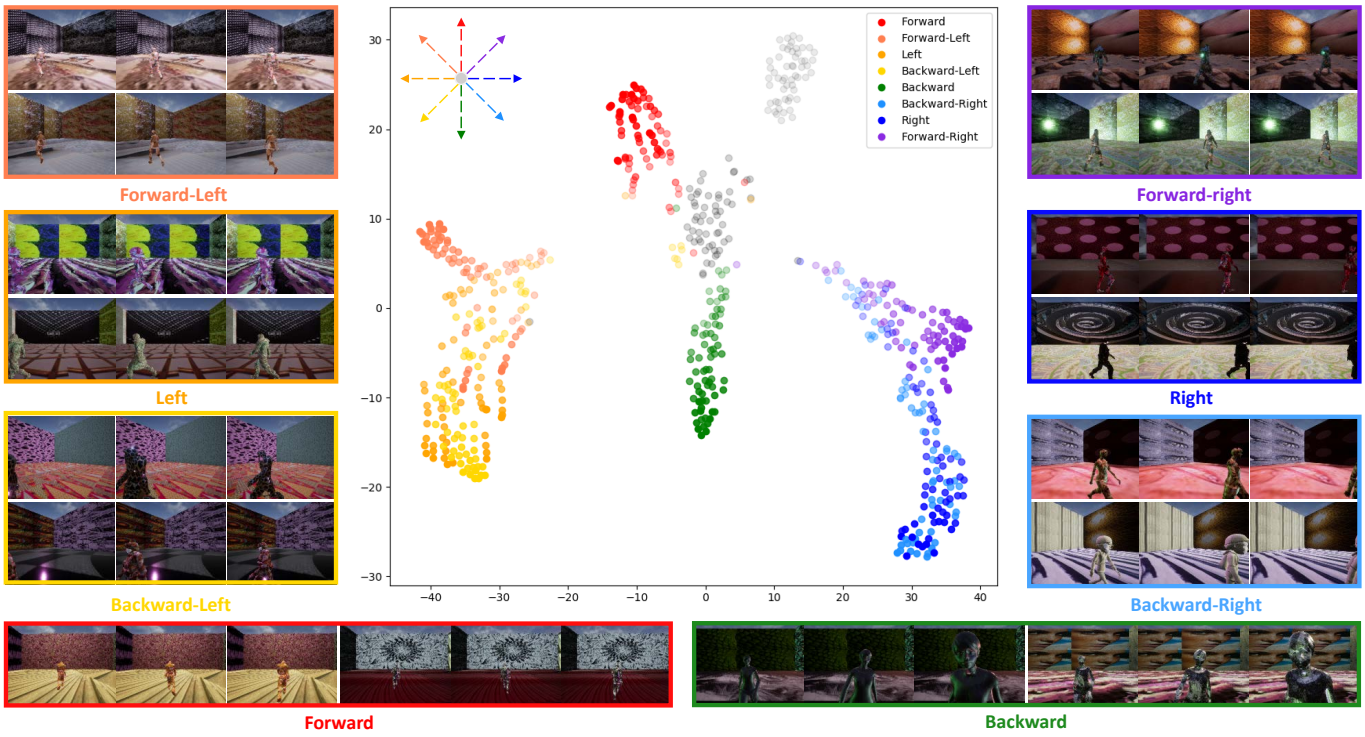

Fig. 10. t-SNE visualization of the latent vectors. The gray dot is the feature of the first four frames of each sequence. The colors represents the eight directions, respectively. The darkness of the color corresponds to the distance between the target and the start point. The instances of each direction are shown around the center figure.

of these videos. As it indicates, the embedding vectors are clearly classed into four clusters, which correspond to four directions: forward, backward, left, and right.

We argue that the learned feature is location-sensitive and appearance-invariant. The distance between the projected features are highly correlative to the distance of their corresponding tracker-centric coordinates. It indicates that the feature is location-sensitive. Even though the appearance is different, once the target is close in geometric location, the projected features are also of very short distance. This suggests that the feature is appearance-invariant. The appearance-invariant feature means that the model learns robust features that could adapt to different appearances of the target and backgrounds, even in real-world scenarios (see Sec. 5.4). Such appearance-invariant feature mainly benefits from environment augmentation techniques. From the behavior we observed in Fig. 8, we can see that the target's behavior also contributes to effectively learning the appearance-invariant feature for tracking. The location-sensitive property shows that the relative location of the target is one of the key cues for a good tracking policy. Learning the location-sensitive feature requires experience that the target appearing in different locations. In AD-VAT, the target can naturally generate a wide variety of data with different relative positions for learning, when exploring the escape strategies.

In the left cluster, the feature of the forward-left forms a separate branch at the end. However, the features of two distinct directions (right and backward-right) are still mixed into one branch without being split. A similar case also happens to the right cluster. This result is not ideal but reasonable. First, their visual observations, viewed by a modular camera, are ambiguous. Shown as the instances in Fig. 8, they are even confusing to human eyes. Second, it can be partly attributed to the limited action space. Because the two moving directions of backward-right/left are unavailable in our action space. So the tracker also outputs action "turn right/left" as the right/left case. The same output action finally causes that the model maps the observations closely into a latent space.

5.4 Transfer The Tracker to Real-World Video Clips

we conduct a qualitative evaluation as [11], to verify the capability of our tracker in real-world scenarios. In this evaluation, we feed the video clips from VOT dataset [49] to the tracker and observe the network output actions. Note that the tracker could not really control the camera movement, hence we regard the test as "passively". However, the tracker's output action is expected to be sensitive to the position and the scale of the target in the image. For example, when the target appears in the right (left) side, the tracker tends to turn right (left), trying to fictitiously move the camera to "place" the target in the image center. By visualizing whether the output action is consistent with the position and scale of the target at each frame, we are able to demonstrate the potential of transferring the tracking ability to real-world.

We plot three "Action" maps, shown in Fig. 11. Note that the meaning of each axis is the same as [11] except that we normalize the values for better understanding. In details, the horizontal axis indicates the normalized x-axis position of the target in the image, with a positive (negative) value

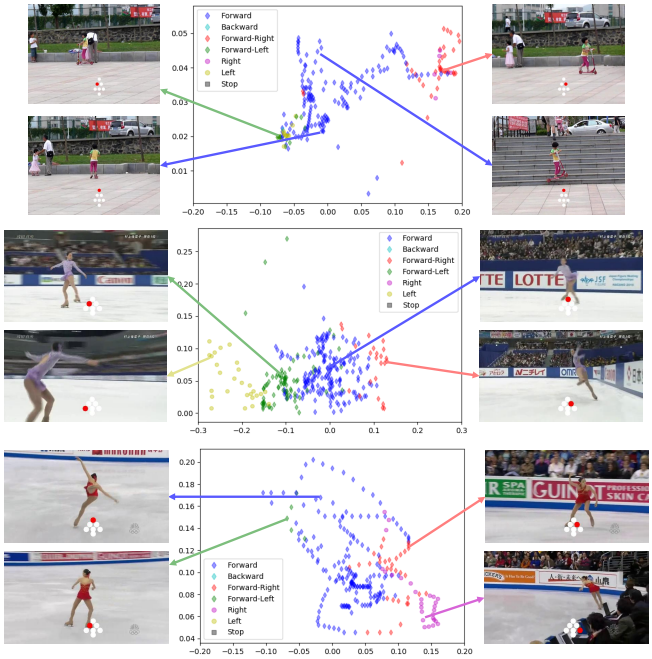

Fig. 11. The Action map from AD-VAT tracker fed with the real-world video clips. From top to bottom is three sequences from VOT dataset: VOT2015-girl, VOT2015-iceskater, VOT2013-iceskater1.

meaning that a target is in the right (left) side. The vertical axis indicates the normalized size of the target, *i.e.*, the area of the ground truth bounding box. We use seven marks to represent the seven actions respectively, as shown in the legend.

More results on other VOT videos are available in the supplementary material.

6 Conclusion

In this paper, we have proposed an asymmetric dueling mechanism for visual active tracking (AD-VAT) and its enhancement AD-VAT+. Within AD-VAT, agents of tracker and target are learned in an adversarial manner. With the design of the "partial zero-sum reward" structure and the tracker-aware target model, the reinforced active tracker outperforms baseline methods. The AD-VAT+, with a developed two-stage training method and the more advanced environment augmentation technique, further improves the robustness of the tracker in more challenging scenarios. Experiments including ablation study in both 2D and 3D environments verify the effectiveness of the proposed mechanism.

As future work, we would like to: 1) investigate the theoretical justification of applying modern Multi-Agent RL methods [34], [39] to solving Partially Observable Markov Game and finding Nash Equilibrium; 2) further develop the model for active tracking in more complex environment; 3) adapt the mechanism to other tasks (*e.g.*, learning to control a robotarm to grab a moving object).

Acknowledgments

The authors would like to thank Weichao Qiu for his help in extending UnrealCV. Fangwei Zhong, Tingyun Yan and Yizhou Wang were supported in part by the following grants: NSFC-61625201, NSFC-61527804, Tencent AI Lab Rhino-Bird Focused Research Program, Qualcomm University Collaborative Research Program.

REFERENCES

[1] B. Babenko, M.-H. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," in *The IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 983–990.

[2] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 125–141, 2008.

[3] X. Mei and H. Ling, "Robust visual tracking using l1 minimization," in *International Conference on Computer Vision*, 2009, pp. 1436–1443.

[4] W. Hu, X. Li, W. Luo, X. Zhang, S. Maybank, and Z. Zhang, "Single and multiple object tracking using log-euclidean riemannian subspace and block-division appearance model," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 12, pp. 2420–2440, 2012.

[5] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *The IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2544–2550.

[6] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012.

[7] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[9] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[10] 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 *International Conference on Machine Learning*, 2016, pp. 1928–1937.

[11] W. Luo, P. Sun, F. Zhong, W. Liu, T. Zhang, and Y. Wang, "End-to-end active object tracking via reinforcement learning," in *International Conference on Machine Learning*, 2018, pp. 3286–3295.

[12] G. Zhu, F. Porikli, and H. Li, "Beyond local search: Tracking objects everywhere with instance-specific proposals," in *The IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 943–951.

[13] F. Zhong, P. Sun, W. Luo, T. Yan, and Y. Wang, "AD-VAT: An asymmetric dueling mechanism for learning visual active tracking," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=HkgYmhR9KX

[14] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *The IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2411–2418.

[15] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[16] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *International Conference on Computer Vision*, 2015, pp. 3074–3082.

[17] J. Choi, J. Kwon, and K. M. Lee, "Visual tracking by reinforced decision making," *arXiv preprint arXiv:1702.06291*, 2017.

[18] J. Denzler and D. W. Paulus, "Active motion detection and object tracking," in *International Conference on Image Processing*, vol. 3, 1994, pp. 635–639.

[19] K. K. Kim, S. H. Cho, H. J. Kim, and J. Y. Lee, "Detecting and tracking moving object using an active camera," in *International Conference on Advanced Communication Technology*, vol. 2, 2005, pp. 817–820.

[20] Z.-W. Hong, C. Yu-Ming, S.-Y. Su, T.-Y. Shann, Y.-H. Chang, H.-K. Yang, B. H.-L. Ho, C.-C. Tu, Y.-C. Chang, T.-C. Hsiao *et al.*, "Virtual-to-real: Learning to control in visual semantic segmentation," *arXiv preprint arXiv:1802.00285*, 2018.

[21] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[22] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[23] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *arXiv preprint arXiv:1702.02284*, 2017.

[24] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," *arXiv preprint arXiv:1703.02702*, 2017.

[25] A. Mandlekar, Y. Zhu, A. Garg, L. Fei-Fei, and S. Savarese, "Adversarially robust policy learning: Active construction of physically-plausible perturbations," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3932–3939.

[26] D. Held, X. Geng, C. Florensa, and P. Abbeel, "Automatic goal generation for reinforcement learning agents," *arXiv preprint arXiv:1705.06366*, 2017.

[27] S. Sukhbaatar, Z. Lin, I. Kostrikov, G. Synnaeve, A. Szlam, and R. Fergus, "Intrinsic motivation and automatic curricula via asymmetric self-play," *arXiv preprint arXiv:1703.05407*, 2017.

[28] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[29] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch, "Emergent complexity via multi-agent competition," *arXiv preprint arXiv:1710.03748*, 2017.

[30] M. J. Osborne and A. Rubinstein, *A course in game theory*. MIT press, 1994.

[31] N. Vlassis, "A concise introduction to multiagent systems and distributed artificial intelligence," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 1, no. 1, pp. 1–71, 2007.

[32] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine Learning Proceedings 1994*. Elsevier, 1994, pp. 157–163.

[33] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, p. 484, 2016.

[34] M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, J. Perolat, D. Silver, T. Graepel *et al.*, "A unified game-theoretic approach to multiagent reinforcement learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4190–4203.

[35] G. W. Brown, "Iterative solution of games by fictitious play," *Activity analysis of production and allocation*, vol. 13, no. 1, pp. 374–376, 1951.

[36] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, "Regret minimization in games with incomplete information," in *Advances in Neural Information Processing Systems*, 2008, pp. 1729–1736.

[37] N. Brown, A. Lerer, S. Gross, and T. Sandholm, "Deep counterfactual regret minimization," *arXiv preprint arXiv:1811.00164*, vol. abs/1811.00164, 2018.

[38] J. Heinrich and D. Silver, "Deep reinforcement learning from self-play in imperfect-information games," *arXiv preprint arXiv:1603.01121*, 2016.

[39] S. Srinivasan, M. Lanctot, V. Zambaldi, J. Pérolat, K. Tuyls, R. Munos, and M. Bowling, "Actor-critic policy optimization in partially observable multiagent environments," in *Advances in Neural Information Processing Systems*, 2018, pp. 3426–3439.

[40] T. Sun, "The art of war," in *Strategic Studies*. Routledge, 2008, pp. 63–91.

[41] W. Qiu, F. Zhong, Y. Zhang, S. Qiao, Z. Xiao, W. Y. Kim, Tae Soo, and A. Yuille, "Unrealcv: Virtual worlds for computer vision," in *Proceedings of the 2017 ACM on Multimedia Conference*. ACM, 2017, pp. 1221–1224.

[42] F. Zhong, W. Qiu, T. Yan, A. Yuille, and Y. Wang, "Gym-unrealcv: Realistic virtual worlds for visual reinforcement learning," Web Page, 2017. [Online]. Available: https://github.com/unrealcv/gym-unrealcv

[43] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[44] 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 Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[45] G. Kylberg, "The kylberg texture dataset v. 1.0," Centre for Image Analysis, Swedish University of Agricultural Sciences and Uppsala University, Uppsala, Sweden, External report (Blue series) 35, September 2011. [Online]. Available: http://www.cb.uu.se/~gustaf/texture/

[46] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.
[47] D. Griffis, "A3c lstm atari with pytorch plus a3g design," Web Page. [Online]. Available: https://github.com/dgriff777/rl_a3c_pytorch
[48] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.
[49] M. Kristan, J. Matas, A. Leonardis, T. Vojir, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Čehovin, "A novel performance evaluation methodology for single-target trackers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 11, pp. 2137–2155, Nov 2016.