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

Fangwei Zhong[†], **Peng Sun**[‡], **Wenhan Luo**[‡], **Tingyun Yan**[†], **Yizhou Wang**[†]
[†] Peking University, [‡] Tencent AI Lab

Abstract

Visual active tracking (VAT) aims at following a target object by autonomously controlling the motion system of a tracker given visual observations. In this paper, we propose a novel method which adopts an asymmetric dueling mechanism for learning visual active tracking, namely AD-VAT. In AD-VAT, the target and the tracker are mutual opponents, *i.e.*, the tracker manages to lockup the target, and the target tries to escape from the tracker. In the implementation, both the tracker and the target are approximated by deep networks, and their policies that map environment observations to control actions can be learned via reinforcement learning in an end-to-end manner. The tracker and the target are asymmetric in observations, network structures and reward functions. Different from the tracker, the target is modeled with a tracker-aware network, *i.e.*, besides its own observation, the tracker's observations and actions are also fed as input to the network. In addition, it learns to predict the tracker's reward as an auxiliary task. We argue that such an asymmetric adversarial mechanism is able to learn a stronger target, which vice versa induces a more robust tracker. The experimental results, in both 2D and 3D environments, demonstrate that the proposed method leads to a faster convergence in training the tracker and more robust tracking behaviors in different testing scenarios.

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 (Babenko et al., 2009; Ross et al., 2008; Mei & Ling, 2009; Hu et al., 2012; Bolme et al., 2010; Kalal et al., 2012) 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 (Sutton & Barto, 1998; Mnih et al., 2015; Silver et al., 2016b; Mnih et al., 2016), training an end-to-end deep neural network via reinforcement learning for VAT is shown to be feasible (Luo et al., 2018). 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.

However, the performance of the deep reinforcement learning based tracker is still limited by the training methods. Due to the "trial-and-error" nature of reinforcement learning, it is hard to directly train the tracker in real world. Alternatively, virtual environments are always utilized to generate

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

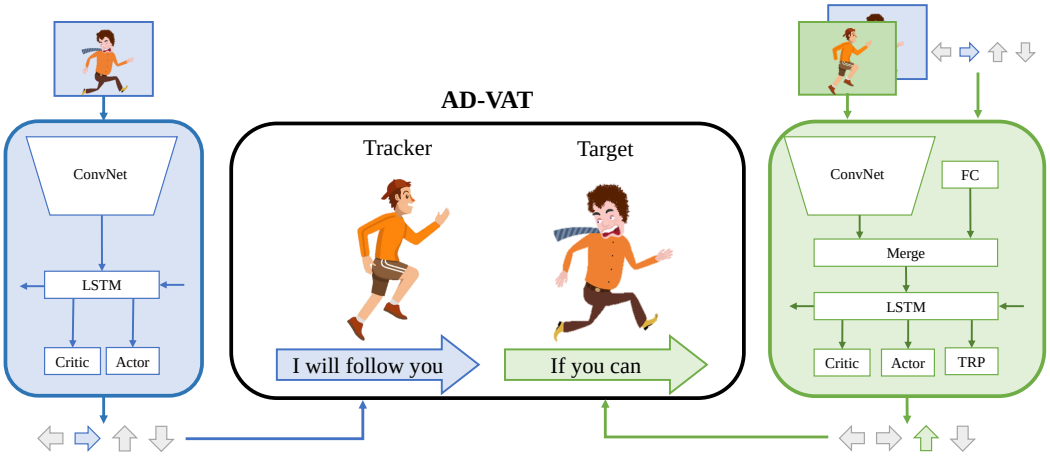

Figure 1: An overview of AD-VAT. Note that TRP is *Tracker Reward Predictor.*

sufficient data for training without tedious human labeling. Nevertheless, to deploy the trained tracker in real-world, one has to overcome the gap between virtual and real environments. One solution can be building numbers of high-fidelity environments (Zhu et al., 2017). 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 the 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. (2018) 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 trajectory of the target object 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 track other motion patterns, like a sharp turn.

In this work, we propose an *Asymmetric Dueling mechanism for learning Visual Active Tracking (AD-VAT)*(see Fig.1) where the tracker and the target object, viewed as two learnable agents, are opponents and can mutually enhance during training. The two agents are different in observation, network architecture and goal, yet are competitive, hence the name "asymmetric dueling". The tracker manages to follow the target. Meanwhile, the target tries to escape from the tracker. We shape a reward structure to guide the learning of the two agents. Different from the zero-sum reward structure (Bansal et al., 2017; Pinto et al., 2017), we adopt a novel reward referred to as "partial zero-sum", which only encourages the tracker-target competition when they are reasonably close and penalizes when they are too far-away. We find empirically that the partial zero-sum stabilizes the training. If the zero-sum were adopted, the training would become highly unstable in the sense that 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. When exploring the escape policy, the target consequently generates various trajectories of its actions. Such escape trajectories could be regarded as "hard examples" for the tracker, which should be beneficial for the learning of the tracker. As the training proceeds, the environments of AD-VAT naturally form a curriculum, because the tracker is more likely to compete with a target with the appropriate difficulty level when both agents are becoming stronger simultaneously. Thus, it improves the sample-efficiency of learning the tracker. To learn the optimal policy to escape, we model the target with a tracker-aware network, *i.e.*, besides its own observation, the observation and actions of the tracker's are also fed to the escaping network. In addition, it learns to predict the reward of the tracker as an auxiliary task, which could help the target better shape a representation about the tracker to learn the optimal policy to escape. We argue that such an "asymmetric dueling" mechanism is able to learn a stronger target, which vice versa yields a more robust tracker ultimately.

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 the two important components: "partial zero-sum reward" and "tracker-aware network". The 3D environments are built on Unreal Engine 4, a popular game engine for building high-fidelity environments. We choose a large room for training, where the texture of the background/players and the illumination are randomized. Three realistic scenarios built by artists are used for further evaluating the generalization. In the 3D environments, we further demonstrate that the tracker trained in AD-VAT is capable of generalizing to high-fidelity environments even it is trained in a simple environment.

The contributions of our work can be summarized as follows: (1)We propose an asymmetric dueling mechanism for learning visual active tracking (AD-VAT). Within the proposed AD-VAT, target generates diverse trajectories when competing with the tracker, ultimately driving the learning of the tracker; (2) We shape a partial zero-sum reward structure for AD-VAT. The reward structure makes the training efficiently; (3) We develop a tracker-aware network for building an effective target agent. This yields better escaping policy and consequently better tracking policy.

2 Related Work

Active Object Tracking. As described above that, 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) (Wu et al., 2013). In general, this problem could be addressed in a two-step manner or in a 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 (Babenko et al., 2009; Ross et al., 2008; Mei & Ling, 2009; Hu et al., 2012; Bolme et al., 2010; Kalal et al., 2012). Thus one can utilize mature visual tracking algorithms (Henriques et al., 2015; Ma et al., 2015; Choi et al., 2017) 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 (Denzler & Paulus, 1994) a two-stage method is proposed to handle robot control by motion detection and motion tracking. Kim et al. (Kim et al., 2005) detect moving objects and track them using an active camera with pan/tilt/zoom. In (Hong et al., 2018), two modules, a perception module and a control policy module, are learned separately to train an agent accomplishing both obstacle avoidance task and a target following task.

Admittedly, tracking algorithms has been successful while still not perfect. Camera control based on tracking results is challenging due to factors such as that the correspondence between image space and camera parameter space is unknown. Additionally, joint tuning of visual tracking and camera control is expensive and encounters 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 (Luo et al., 2018). This end-to-end solution verifies the effectiveness, while not efficient enough to solve this problem. To improve the generalization potential, environment augmentation is conducted in this work. However, the target itself, like the path, motion pattern is fixed in their augmented environments, which is believed to limit the performance. This inspires us to resort to the idea of dueling in this paper, *i.e.*, the target learns to get rid of the tracker by itself, guiding the learning of the tracker.

Self-play method for reinforcement learning. In terms of game theory, our setting is a 2-player non-cooperative extensive game with partial observation (Osborne & Rubinstein, 1994). When the reward is zero-sum, some self-play algorithm with a specific value function (*e.g.*, Counterfactual Regret, abbreviated as CFR (Zinkevich et al., 2008)) is proven to approximate the Nash Equilibrium, which means no player can obtain more rewards by unilaterally altering its policy, and henceforth produces very strong policy. CFR and its variants (*e.g.*, CFR+ (Tammelin, 2014), Monte Carlo CFR (Burch et al., 2012)) are shown to be successful in several applications. For instance, the poker AI Libratus defeats professional human players in 2-player unlimited Texas Hold'em (Brown & Sandholm, 2017), where CFR is utilized in combination of other careful devised techniques. In specific, the observation sequence is formalized as a game tree, where similar observations/actions are merged (called abstraction) and sub-games are solved efficiently. However, the abstraction and sub-game solving technique cannot be applied to our task in a straightforward way. In our VAT task, the observation is screen pixels, for which we need a more efficient visual feature representation, preferably learned automatically with, *e.g.*, a deep ConvNet. There are attempts to combine CFR with deep nets (Jin et al., 2017; Lanctot et al., 2017; Moravčík et al., 2017), however, they lose the

theoretical guarantee in regards of finding the Nash Equilibrium, despite they show encouraging empirical results.

On the other hand, there is a body of work that combines self-play with deep net to train a policy in either partial or perfect observation, *e.g.*, playing board game (Tesauro, 1995; Silver et al., 2016a; 2017b;a), playing video game (ope; Jaderberg et al., 2018), robot controlling (Bansal et al., 2017; Pinto et al., 2017), navigation in maze (Sukhbaatar et al., 2017), collision avoidance in autonomous driving (Behzadan & Munir, 2018), *etc.*The methods just utilize the conventional value functions in the RL literature, and also shown good empirical results. Despite lacking theoretical guarantee of Nash Equilibrium finding, several techniques, in particular the domain randomization and the opponent pool sampling, are suggested to ensure the policy diversity and the training stabilization. Noting this, we stick to the conventional value function when performing the self-play training in this work.

3 Method

In this section, we introduce our proposed method: *Asymmetric Dueling mechanism for learning Visual Active Tracking (AD-VAT)*. At first the proposed is formulated as multi-agent system. Then we illustrate the two key components: the reward structure and a tracker-aware model for the target.

3.1 Formulation

For the notations of our two-agent settings, let superscript α denote the tracker and superscript β denote the target. The policy of the tracker, $\pi^{\alpha}(a_t^{\alpha}|o_t^{\alpha})$, is a distribution over tracker action a_t^{α}, and conditioned on tracker observation o_t^{α}, where the subscript t denotes time step. When the policy takes as function approximator a Neural Network with parameter θ^{α}, we further write it as

$$\pi^{\alpha}(a_t^{\alpha}|o_t^{\alpha};\theta^{\alpha}). \tag{1}$$

Likewise, the policy of the target can be written as

$$\pi^{\beta}(a_t^{\beta}|o^{\beta_t};\theta^{\beta}). \tag{2}$$

The tracker intends to maximize its expected return

$$\mathbb{E}_{\pi^{\alpha},\pi^{\beta}}\left[\sum_{t=1}^{T} r_t^{\alpha}\right] \tag{3}$$

by learning the parameter θ^{α}, where T denotes the horizon length of an episode and r_t^{α} is the immediate reward of the tracker at time step t. In contrast, the target tries to maximize

$$\mathbb{E}_{\pi^{\alpha},\pi^{\beta}}\left[\sum_{t=1}^{T} r_t^{\beta}\right] \tag{4}$$

by learning θ^{β}.

3.2 Reward Structure

The conventional adversarial methods (Bansal et al., 2017; Pinto et al., 2017) 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, $r_t^{\alpha} + r_t^{\beta} = 0$. However, such kind of formulation is not suitable for AD-VAT. 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 latent MDP transition is usually meaningless and ineffective for improving the skill level of the agent. 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 utilize the zero-sum reward only when the target is observed by tracker, but gives penalties to each agent 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 (Luo et al., 2018), composing of a positive constant and an error penalty term. However, we do not take the orientation difference between the target and the tracker into consideration. Moreover, we measure the relative position

error based on a polar coordinate system in the 3D environment, considering the model of the camera observation. With a slight abuse of notation, we can now write the reward function as

$$r^{\alpha} = A - \zeta dist(P^{\beta}, P^{exp}), \tag{5}$$

here $A > 0$, $\zeta > 0$ are tuning parameters, specially $A = 1.0$, $\zeta = 2.0$. P^{β} and P^{exp} represents the target object's position and the expected position. Note that P is the relative coordinates in the tracker-centric coordinate system. Each term of the coordinate is normalized regarding the range of the observation R. Specifically, P is plane position (x, y) in the 2D environment, but is under a polar coordinates (ρ, θ) in 3D environment. ρ is the distance between tracker and target, θ is the relative angle to the front of the camera. 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 reward of the tracker, written as below:

$$r^{\beta} = \begin{cases} -r^{\alpha}, & \text{if } r^{\alpha} > -A \\ -r^{\alpha} - \xi dist(P^{\beta}, P^{far}), & \text{otherwise} \end{cases} \tag{6}$$

where r^{α} is the reward of the tracker as defined in Eq. 5, $\xi > 0$ is a tuning parameter controlling the factor of the penalty term, and $P^{far} = (1, 1)$, representing the farthest point to tracker within the observation. The reward is also clipped r_t is smaller than $-A$. The insight behind this formulation is that, we do not encourage the target to escape from the tracker as far as possible. Alternatively, by applying this reward function, the optimal policy for the target we expect should be escaping and disappearing from the observation range of the tracker but keeping close to the tracker. In Appendix A we visualize the reward structure given in Eq. (5) and (6) for a better understanding.

3.3 Tracker-aware Target

The "tracker-aware" idea is inspired by 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. The conventional adversary usually uses only its own observation Bansal et al. (2017) or shares the same observation as the opponent Pinto et al. (2017). Recall the target policy $\pi^{\beta}(a_t^{\beta}|o_t^{\beta}; \theta^{\beta})$ written as in Eq. (2). However, the imperfect/partial o^{β} 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 seen as a kind of "opponent modeling", and alleviate the difficulty in its own policy learning. By doing so, we can write the output heads of such a "tracker-aware" policy network as $\pi^{\beta}\left(a_t^{\beta}, \hat{r}_t^{\alpha}|o_t^{\alpha}, a_t^{\alpha}, o_t^{\beta}\right)$ in a slight abuse of notations, where $\hat{r}_t^{\alpha}$ is the predicted immediate reward for the tracker and o_t^{α}, a_t^{α} are respectively the observation and the action of the tracker.

4 Experiments

4.1 Environments

2D Environments. In the 2D Environment, maps are represented by a 80×80 matrix, where 0 denotes free space, 1 denotes an obstacle, 2 denotes the tracker, and 4 denotes the target. We generate the maps of two patterns, "random maze" and "random block" (see examples in the top row of Fig.2). We use maps of block for training and maps of maze for testing. For both the tracker and the target, the observation is a matrix of size 13×13 around the agent. The tracker aims to place the target in the center of the observation. 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. The 2D environments provide ideal conditions due to lack of observation or motion noise. The experiments in the 2D environments are dedicated to evaluate and quantify the effectiveness of our approach in ideal conditions.

3D Environments. The 3D environments are more similar to the real-world setting in observation and action. In the 3D environments, the observation is an image of the first-person view of the world as seen by the agent. The possible actions are *move-forward, move-backward, turn-left, turn-right, turn-left-and-move-forward, turn-right-and-move-forward*, and *no-op*.

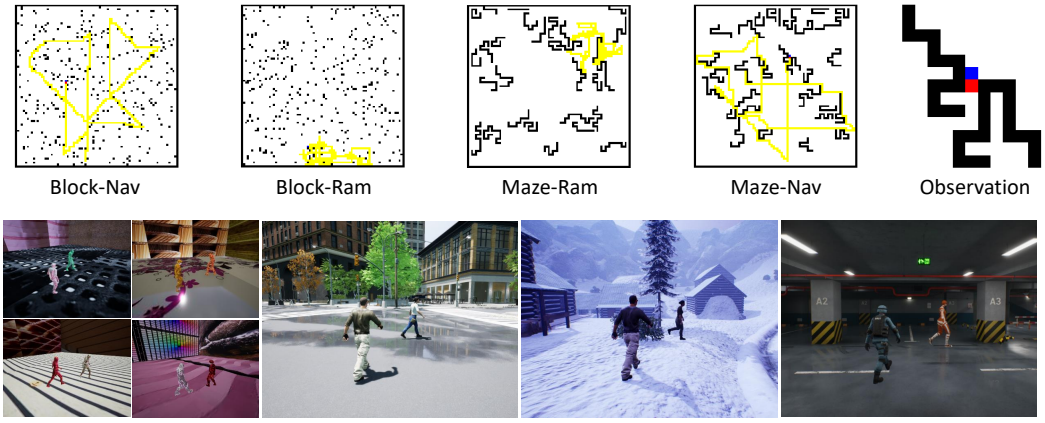

Figure 2: The environments used for training and testing.

UnrealEnv is a photo-realistic and flexible simulation environment built on the Unreal Engine. We employ UnrealCV (Qiu et al., 2017), which provides convenient APIs, along with a wrapper (Zhong et al., 2017) compatible with OpenAI Gym (Brockman et al., 2016), for interactions between RL algorithms and the UnrealEnv. For training, we built a *Domain Randomized Room(DR Room)* with two controllable players (target and tracker). The domain randomization techniques used could help agents learn better feature representation in terms of visual observation. In testing, we focus on the transferring ability of the tracker to different unseen environments. We use three realistic scenarios, *Urban City, Snow Village* and *Parking Lot*, to mimic real-world scenes for evaluating. The bottom row in Fig.2 shows the snapshots of the four 3D environments used.

Details of the four environments are: (1) *DR Room* is a plain room of floor and walls only, but the textures and illumination conditions are randomized. For the textures, we randomly choose pictures from a texture dataset (Kylberg, 2011) 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, orientation. (2) *Urban City* is a street view of urban city, including well modelled buildings, streets, trees and transportation facilities with high fidelity. Besides, there are some puddles on the road, reflecting the buildings. (3) *Snow Village* consists of bumpy snowfields with several trees, bushes and some cabins. Specifically, the bumpy snowfield is challenging for tracking, because it frequently causes the situation that agent observation is not in the same height. (4) *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.

4.2 Baselines

We provide two kinds of base target agent to randomly generate trajectories as baselines to compare, *Rambler (Ram)* and *Navigator (Nav)*. Agent *Ram* walks randomly without any purpose like a man. 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 $(1, 10)$. Agent *Nav* is a navigator, planning the shortest path to a specific goal in use of the Astar algorithm (Hart et al., 1968). Thus, it could navigate to most of free space in the map. To randomize the trajectories, the goal coordinate and the initial coordinate are randomly sampled from the free space. In most of case, *Ram* prefers to walk around a local area repeatedly. In contrast, *Nav* would like to explore the map globally, shown as the yellow trajectories in Fig. 2. Thus we regard trajectories from the *Ram* as easier cases, and trajectories from *Nav* as more difficult cases for tracker.

4.3 Implementation Details

Each agent is trained by A3C (Mnih et al., 2016), a commonly used reinforcement learning algorithm. Multiple workers are running in parallel when training. Specifically, 16 workers are used in the 2D experiment, and 4 workers are used in the 3D experiment.

Network Architecture. For the tracker, we follow the end-to-end Conv-LSTM network architecture as (Luo et al., 2018). Differently, the Conv-Net in this paper is deeper, and the input color images are pre-processed, transformed to gray image and the pixel values are scaled to $[-1, 1]$. we also develop

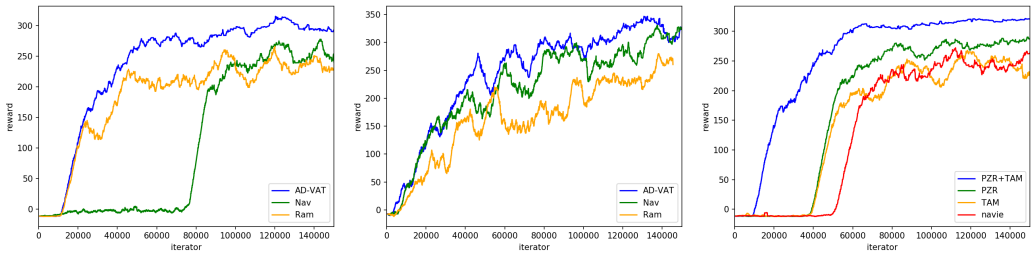

Figure 3: Cumulative reward curves for tracker trained in AD-VAT versus the baseline methods when tested in the validation environment. For both 2D (left) and 3D (middle) settings, AD-VAT improves the sample-efficiency and achieves higher reward. The right figure is the learning curve of the ablation study in 2D environment.

Table 1: Results on the 2D environments.

Environment		**Accumulated Reward(AR)**			**Episode Length(EL)**		
Map	Target	Ram	Nav	AD-VAT	Ram	Nav	AD-VAT
Maze	Ram	350±23	287±99	**353±22**	500±0	439±131	**500±0**
	Nav	243±128	257±126	**264±108**	412±175	409±173	**431±143**
Block	Ram	352±53	303±67	**362±16**	491±64	462±83	**500±0**
	Nav	265±134	246±144	**308±60**	414±178	386±194	**488±69**

the Conv-LSTM network architecture for the target, but different in the input and output, shown as Fig.1. The network parameters are updated with a shared Adam optimizer.

Hyper parameters. For the tracker, the learning rates δ_0^α and $\delta_1^{\alpha\prime}$ 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 $\lambda^\alpha = 0.01$. The parameter updating frequency n is 20, and the maximum global iteration for training is 150K. Comparing to the tracker, a higher regularizer factor is used for encouraging the target to explore, $\lambda_0^\beta = 0.2$ in 2D and $\lambda_1^\beta = 0.05$ in 3D. The more exploration taken by target, the more diverse the generated trajectories are. It is useful for the learning of the tracker. 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* agents, the *Nav* agent is more challenging, thus is more suitable for validation.

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, and EL roughly measures the duration of good tracking. Because the episode is done when the tracker lost for continuous 10 steps or reach the max episode length.

4.4 Results on the 2D environment

We quantitatively evaluate the performance of our approach, comparing to the two baselines. Furthermore, we conduct an ablation study to show the effect of the tracker-aware model and partial zero-sum reward.

Quantitative Evaluation. We test the active tracker trained with different target agents in four testing settings, showing the effectiveness of *AD-VAT*. 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 1. 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 perfectly, without losing the target.

We note that, at the beginning of the learning, the adversarial target usually walks randomly around the start point, performing similar policy as *Ram*. Such 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 pattern, which could further reinforce the tracker. Such a learning process is close to the curriculum learning, but the curriculum is automatically produced by the target via adversarial reinforcement learning. We also report the learning curve as the mean of cumulative rewards in the validation environment, shown as the left sub-figure in Fig 3. It consistently shows the advantage of the proposed *AD-VAT*.

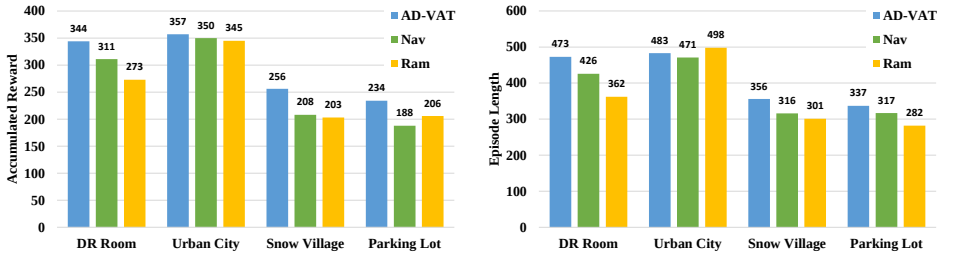

Figure 4: Results on the 3D environment. The higher is the better.

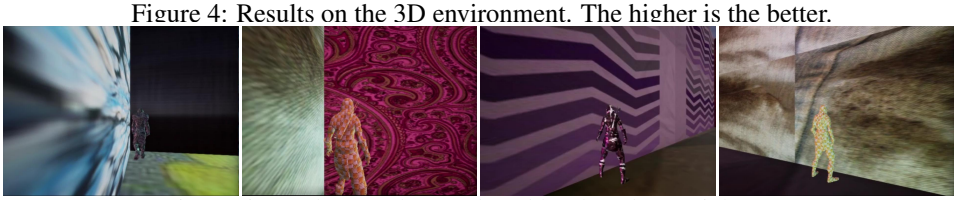

Figure 5: Hard examples produced by the adversarial target.

Ablation Study. In Section 3, we introduced two components to implement *AD-VAT*: "partial zero-sum" reward structure (PZR) and tracker-aware model (TAM) for target. These two components are important as they influence the natural curriculum for the tracker. Thus, we report an ablation study result to show the effectiveness of these two components, shown in Fig. 3. The naive method is an intuitive idea that target only uses its own observation with auxiliary task, guided by a zero-sum reward. As shown in the right sub-figure of Fig. 3, using each component separately could improve the sample-efficiency, comparing to the naive method. Besides, PZR contributes to the improvement of the tracking performance more significant than TAM. Moreover, when combining PZR and TAM, both sample-efficiency and the tracking performance are significantly boosted, comparing to the other three settings.

4.5 Results on the 3D environment

In the 3D UnrealEnv, we test the effectiveness of the model in unseen realistic scenarios, showing the transfer potential in real-world scenario. We train the three models with different target models (*Ram, Nav*, and *AD-VAT*) in the DR Room. And then, we directly run the three trackers in the validation and testing environments, 100 episodes for each, and report AR and EL in Fig. 4

The result in DR Room demonstrates again that even though the target model is unseen for our *AD-VAT* tracker, it still outperforms the others. Because the DR Room for validation is of the same settings as the training of *Nav* tracker. The results in UrbanCity show that the three models are able to transfer to realistic environment.

We believe that the domain randomization method and the Conv-LSTM network endow the trackers the ability of transferring. However, the performance of the two baselines degrades more compared with our method, in other environments. So we think that the two baseline methods are not sufficiently effective when transferring to challenging environments. Training with *AD-VAT* could significantly improve the capability of the tracker in challenging environments. We infer that the target in *AD-VAT* could explore the environment more actively to discover more challenging cases. For example, in DR Room, the target would prefer to move close to the wall that is similar to itself to fool the tracker(see Fig. 5). By competing with the target, the tracker consequently becomes stronger.

5 Conclusion

In this paper, we have proposed a mechanism AD-VAT for visual active tracking. 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. Experiments including ablation study in both 2D and 3D environments verify the effectiveness of the proposed mechanism.

References

Openai five. https://blog.openai.com/openai-five/. Accessed August 30, 2018.

Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Visual tracking with online multiple instance learning. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pp. 983–990, 2009.

Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*, 2017.

Vahid Behzadan and Arslan Munir. Adversarial reinforcement learning framework for benchmarking collision avoidance mechanisms in autonomous vehicles. *arXiv preprint arXiv:1806.01368*, 2018.

David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2544–2550, 2010.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Noam Brown and Tuomas Sandholm. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, pp. eaao1733, 2017.

Neil Burch, Marc Lanctot, Duane Szafron, and Richard G Gibson. Efficient monte carlo counterfactual regret minimization in games with many player actions. In *Advances in Neural Information Processing Systems*, pp. 1880–1888, 2012.

Janghoon Choi, Junseok Kwon, and Kyoung Mu Lee. Visual tracking by reinforced decision making. *arXiv preprint arXiv:1702.06291*, 2017.

Joachim Denzler and Dietrich WR Paulus. Active motion detection and object tracking. In *International Conference on Image Processing*, volume 3, pp. 635–639, 1994.

Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015.

Zhang-Wei Hong, Chen Yu-Ming, Shih-Yang Su, Tzu-Yun Shann, Yi-Hsiang Chang, Hsuan-Kung Yang, Brian Hsi-Lin Ho, Chih-Chieh Tu, Yueh-Chuan Chang, Tsu-Ching Hsiao, et al. Virtual-to-real: Learning to control in visual semantic segmentation. *arXiv preprint arXiv:1802.00285*, 2018.

Weiming Hu, Xi Li, Wenhan Luo, Xiaoqin Zhang, Stephen Maybank, and Zhongfei Zhang. Single and multiple object tracking using log-euclidean riemannian subspace and block-division appearance model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(12):2420–2440, 2012.

Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *arXiv preprint arXiv:1807.01281*, 2018.

Peter H Jin, Sergey Levine, and Kurt Keutzer. Regret minimization for partially observable deep reinforcement learning. *arXiv preprint arXiv:1710.11424*, 2017.

Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422, 2012.

Kye Kyung Kim, Soo Hyun Cho, Hae Jin Kim, and Jae Yeon Lee. Detecting and tracking moving object using an active camera. In *International Conference on Advanced Communication Technology*, volume 2, pp. 817–820, 2005.

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

Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Julien Perolat, David Silver, Thore Graepel, et al. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 4190–4203, 2017.

Wenhan Luo, Peng Sun, Fangwei Zhong, Wei Liu, Tong Zhang, and Yizhou Wang. End-to-end active object tracking via reinforcement learning. In *International Conference on Machine Learning*, pp. 3286–3295, 2018.

Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. Hierarchical convolutional features for visual tracking. In *International Conference on Computer Vision*, pp. 3074–3082, 2015.

Xue Mei and Haibin Ling. Robust visual tracking using l1 minimization. In *International Conference on Computer Vision*, pp. 1436–1443, 2009.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1928–1937, 2016.

Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

Martin J Osborne and Ariel Rubinstein. *A course in game theory*. 1994.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. *arXiv preprint arXiv:1703.02702*, 2017.

Weichao Qiu, Fangwei Zhong, Yi Zhang, Siyuan Qiao, Zihao Xiao, Wang Yizhou Kim, Tae Soo, and Alan Yuille. Unrealcv: Virtual worlds for computer vision. In *Proceedings of the 2017 ACM on Multimedia Conference*, pp. 1221–1224. ACM, 2017.

David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1-3):125–141, 2008.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016a.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016b.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017a.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017b.

Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*, 2017.

Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.

Oskari Tammelin. Solving large imperfect information games using cfr+. *arXiv preprint arXiv:1407.5042*, 2014.

Gerald Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3): 58–68, 1995.

Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2411–2418, 2013.

Fangwei Zhong, Weichao Qiu, Tingyun Yan, Alan Yuille, and Yizhou Wang. Gym-unrealcv: Realistic virtual worlds for visual reinforcement learning. Web Page, 2017. URL https://github.com/unrealcv/gym-unrealcv.

Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *International Conference on Robotics and Automation*, 2017.

Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems*, pp. 1729–1736, 2008.

A Visualizing Partial Zero-sum Reward

To help better understand the reward structure given in Eq. (5) and (6), we visualize the sum $r^{\alpha} + r^{\beta}$ as heatmap in $x - y$ plane. See Fig. 6.

For the 2D experiment, the observations for both the tracker and target are bird-views. We want to penalize that the target gets too far away from the tracker. Therefore, the zero-sum area is a circle (Fig. 6, 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 $r^{\beta} = -A$.

For the 3D experiment, 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. 6, 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.

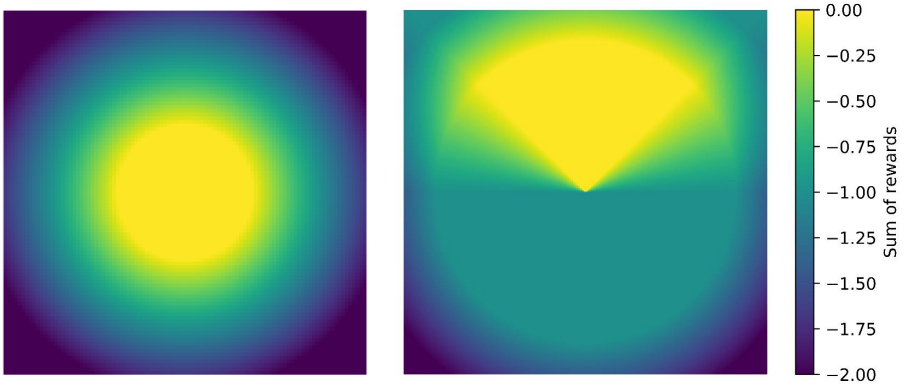

Figure 6: Visualizing $r^{\alpha} + r^{\beta}$ as heatmap in $x - y$ plane with the tracker in the image center. Left: The reward adopted in our 2D environment experiment. Right: The reward adopted in our 3D environment experiment. See the Appendix text for more explanations.

B 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. In Fig. 7, we plot the target position histograms/heatmaps, instead of the trajectory itself. For ease of visualization, we adopt a relative coordinate system for the target position when drawing the heatmaps, because the start locations for both the tracker and the target are random upon each episode. In Fig. 7, the heatmpas are in a *start point*-centric coordinate system, while in Fig. 8 the heatmpas are in a *tracker*-centric coordinate system.

At early training stages (See the left of Fig. 7), *AD-VAT* and *Ram* generate similar trajectories, which are randomly walking around the start point. In contrast, for *Nav* method the target usually goes along a straight line to the goal position, causing the tracker to get lost quickly at beginning. The randomly walking trajectories help the tracker observe the target appearing in various positions, and henceforth sample more diverse experiences. As a result, a better exploration is achieved during the early training stage, which is not the case for the *Nav* method.

With the evolution of the tracker during training, the target will gradually seek more difficult cases to defeat the tracker. In this regard, the target for both *Ram* and *Nav* usually explore possible directions uniformly. The minor difference is that the *Nav* tends to explore the map globally, while the *Ram* is in local (See the right of Fig. 7). As for our AD-VAT method, however, the reinforced target could adapt to the capability of the tracker, resulting to different direction choosing patters at different stage. For

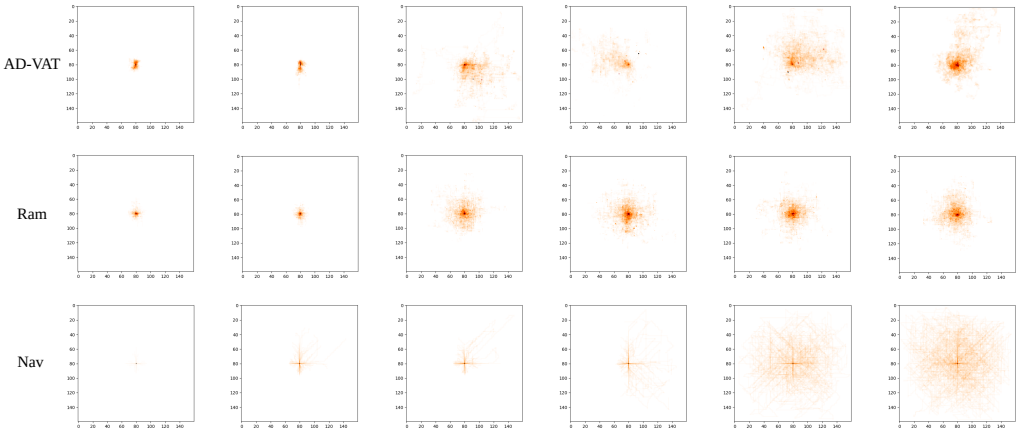

Figure 7: Target position heatmaps in a *start point*-centric coordinate system. Evolution from early training to late training is arranged from left to right. The darker, the bigger the counts.

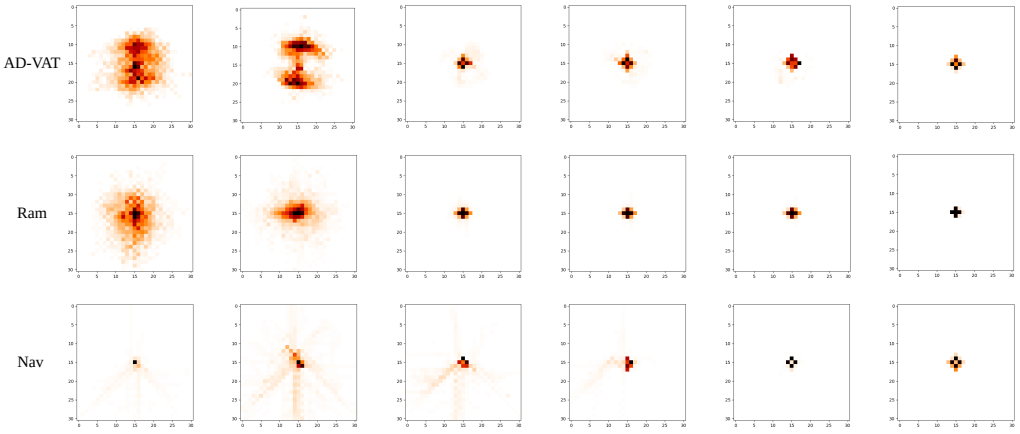

Figure 8: Target position heatmaps in a *tracker*-centric coordinate system. Evolution from early training to late training is arranged from left to right. The darker, the bigger the counts.

example, the target tends to move bottom-right at the third stage, but top-left at the fourth stage. See Fig. 7. Besides, it seems that the reinforced target could balance the two exploration modes of *Nav* and *Ram* naturally. Sometimes it explores the map, and sometimes it duels with the tracker locally. By the dueling mechanism, the target could find the weakness of the tracker more often (See the right of Fig. 8), which seems to serve as a kind of importance sampling that enhances the tracker efficiently during training. Such a "weakness-finding" seems absent for the *Nav* and *Ram* algorithms.