End-to-end Active Object Tracking via Reinforcement Learning

Wenhan Luo [* 1]   **Peng Sun** [* 1]   **Fangwei Zhong** [2]   **Wei Liu** [1]   **Tong Zhang** [1]   **Yizhou Wang** [2]

Abstract

We study active object tracking, where a tracker takes as input the visual observation (*i.e.*, frame sequence) and produces the camera control signal (*e.g.*, move forward, turn left, *etc.*). Conventional methods tackle the tracking and the camera control separately, which is challenging to tune jointly. It also incurs many human efforts for labeling and many expensive trial-and-errors in real-world. To address these issues, we propose, in this paper, an end-to-end solution via deep reinforcement learning, where a ConvNet-LSTM function approximator is adopted for the direct frame-to-action prediction. We further propose an environment augmentation technique and a customized reward function, which are crucial for a successful training. The tracker trained in simulators (ViZ-Doom, Unreal Engine) shows good generalization in the case of unseen object moving path, unseen object appearance, unseen background, and distracting object. It can restore tracking when occasionally losing the target. With the experiments over the VOT dataset, we also find that the tracking ability, obtained solely from simulators, can potentially transfer to real-world scenarios.

1. Introduction

Object tracking has gained much attention in recent decades (Bertinetto et al., 2016a; Danelljan et al., 2017; Zhu et al., 2016; Cui et al., 2016). The aim of object tracking is to localize an object in continuous video frames given an initial annotation in the first frame. Much of the existing work is, however, on the *passive* tracker, where it is presumed that the object of interest is always in the image scene so

---
[*]Equal contribution   [1]Tencent AI Lab [2]Peking University. Correspondence to: Wenhan Luo <whluo.china@gmail.com>, Peng Sun <pengsun000@gmail.com>, Fangwei Zhong <zfw@pku.edu.cn>, Wei Liu <wl2223@columbia.edu>, Tong Zhang <tongzhang@tongzhang-ml.org>, Yizhou Wang <yizhou.Wang@pku.edu.cn>.
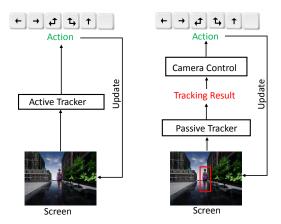
Figure 1. The pipeline of active tracking. Left: end-to-end approach. Right: passive tracking plus other modules.

that there is no need to handle the camera control during tracking. This fashion is inapplicable to some use-cases, *e.g.*, the tracking performed by a mobile robot with a camera mounted or by a drone. To this end, one should seek a solution of *active* tracking, which composes two sub-tasks, *i.e.*, the object tracking and the camera control (Fig. 1, Right).

Unfortunately, it is hard to jointly tune the pipeline with the two separate sub-tasks. The tracking task may also involve many human efforts for bounding box labeling. Moreover, the implementation of camera control is non-trivial and can incur many expensive trial-and-errors happening in real-world. To address these issues, we propose an end-to-end active tracking solution via deep reinforcement learning. To be specific, we adopt a ConvNet-LSTM network, taking as input raw video frames and outputting camera movement actions (*e.g.*, move forward, turn left, *etc.*).

We leverage virtual environments to conveniently simulate active tracking, saving the expensive human labeling or real-world trial-and-error. In a virtual environment, an agent (*i.e.*, the tracker) observes a state (a visual frame) from a first-person perspective and takes an action, and then the environment returns the updated state (next visual frame). We adopt the modern Reinforcement Learning (RL) algorithm A3C (Mnih et al., 2016) to train the agent, where a customized reward function is designed to encourage the

agent to be closely following the object.

We also adopt an environment augmentation technique to boost the tracker's generalization ability. For this purpose, much engineering is devoted to preparing various environments in different object appearances, different backgrounds, and different object trajectories. We manage this by either using a simulator's plug-in or developing specific APIs to communicate with a simulator engine. See Sec. 3.1.

To our slight surprise, the trained tracker shows good generalization capability. In testing, it performs robust active tracking in the case of unseen object movement path, unseen object appearance, unseen background, and distracting object. Additionally, the tracker can restore tracking when it occasionally loses the target due to, *e.g.*, abrupt object movement.

In our experiments, the proposed tracking approach also outperforms a few representative conventional passive trackers which are equipped with a hand-tuned camera-control module. While we are not pursuing a state-of-the-art passive tracker in this work, the experimental results do show that a passive tracker is not indispensable in active tracking. Alternatively, a direct end-to-end solution can be effective. As far as we know, there has not yet been any attempt to deal with active tracking in an end-to-end manner.

Finally, we perform qualitative evaluation on some video clips taken from the VOT dataset (Kristan et al., 2016). The results show that the tracking ability, obtained purely from simulators, can potentially transfer to real-world scenarios.

2. Related Work

Object Tracking. Roughly, object tracking (Wu et al., 2013) is conducted in both passive and active ways. As mentioned in Sec. 1, passive object tracking has gained more attention due to its relatively simpler problem settings. In recent decades, passive object tracking has achieved a great progress (Wu et al., 2013). Many approaches (Hu et al., 2012) have been proposed to overcome difficulties resulted from the issues such as occlusion and illumination variations. In (Ross et al., 2008) subspace learning was adopted to update the appearance model of an object and integrated into a particle filter framework for object tracking. Babenko *et al.* (Babenko et al., 2009) employed multiple instance learning to track an object. Correlation filter based object tracking (Valmadre et al., 2017; Choi et al., 2017b) has also achieved a success in real-time object tracking (Bolme et al., 2010; Henriques et al., 2015). In (Hare et al., 2016), structured output prediction was used to constrain object tracking, avoiding converting positions to labels of training samples. In (Kalal et al., 2012), Tracking, Learning and Detection (TLD) were integrated into one framework for long-term tracking, where a detection module

can re-initialize the tracker once a missing object reappears. Recent years have witnessed the success of deep learning in object tracking (Wang et al., 2016; Bertinetto et al., 2016b). For instance, a stacked autoencoder was trained to learn good representations for object tracking in (Wang & Yeung, 2013). Both low-level and high-level representations were adopted to gain both accuracy and robustness (Ma et al., 2015).

Active object tracking additionally considers camera control compared with traditional object tracking. There exists not much research attention in the area of active tracking. Conventional solutions dealt with object tracking and camera control in separate components (Denzler & Paulus, 1994; Murray & Basu, 1994; Kim et al., 2005), but these solutions are difficult to tune. Our proposal is completely different from them as it tackles object tracking and camera control simultaneously in an end-to-end manner.

Reinforcement Learning. Reinforcement Learning (RL) (Sutton & Barto, 1998) intends for a principled approach to temporal decision making problems. In a typical RL framework, an *agent* learns from the *environment* a *policy* function that maps *state* to *action* at each discrete time step, where the objective is to maximize the accumulated *rewards* returned by the environment. Historically, RL has been successfully applied to inventory management, path planning, game playing, *etc.*

On the other hand, the past half decade has witnessed a breakthrough in deep learning applied to computer vision tasks, including image classification (Krizhevsky et al., 2012), segmentation (Long et al., 2015), object detection and localization (Girshick et al., 2014), and so on. In particular, researchers believe that deep Convolutional Neural Networks (ConvNets) can learn good features from raw image pixels, which is able to benefit higher-level tasks.

Equipped with deep ConvNets, RL also shows impressive successes on those tasks involving image (-like) raw states, *e.g.*, playing board game GO (Silver et al., 2016) and video game (Mnih et al., 2015; Wu & Tian, 2017). Recently, in the computer vision community there are also preliminary attempts of applying deep RL to traditional tasks, *e.g.*, object localization (Caicedo & Lazebnik, 2015) and region proposal (Jie et al., 2016). There are also methods of visual tracking relying on RL (Choi et al., 2017a; Huang et al., 2017; Supancic & Ramanan, 2017; Yun et al., 2017). However, they are distinct from our work, as they formulate passive tracking with RL but have nothing to do with camera control. While our focus in this work is active tracking.

3. Our Approach

In our approach, virtual tracking scenes are generated for both training and testing. To train the tracker, we employ

Figure 2. The architecture of the ConvNet-LSTM network.

a state-of-the-art reinforcement learning algorithm, A3C (Mnih et al., 2016). For the sake of robust and effective training, we also propose data augmentation techniques and a customized reward function, which are elaborated later.

Although various types of states are available, for a research purpose we let the state be only an RGB screen frame of the first-person perspective in this study. To be more specific, the tracker observes the raw visual state and takes one action from the action set $\mathcal{A} = \{$*turn-left, turn-right, turn-left-and-move-forward, turn-right-and-move-forward, move-forward, no-op*$\}$. The action is processed by the environment, which returns to the agent the updated screen frame as well as the current reward.

3.1. Tracking Scenarios

It is impossible to train the desired end-to-end active tracker in real-world scenarios. Thus, we adopt two types of virtual environments for simulated training.

ViZDoom. ViZDoom (Kempka et al., 2016; ViZ) is an RL research platform based on a 3D FPS video game called Doom. In ViZDoom, the game engine corresponds to the environment, while the video game player corresponds to the agent. The agent receives from the environment a state and a reward at each time step. In this study, we make customized ViZDoom maps (see Fig. 4) composed of an object (a monster) and background (ceiling, floor, and wall). The monster walks along a pre-specified path programmed by the ACS script (Kempka et al., 2016), and our goal is to train the agent, *i.e.*, the tracker, to follow closely the object.

Unreal Engine. Though convenient for research, ViZDoom does not provide realistic scenarios. To this end, we adopt Unreal Engine (UE) (unr) to construct nearly real-world environments. UE is a popular game engine and has a broad influence in the game industry. It provides realistic scenarios which can mimic real-world scenes (please see exemplar images in Fig. 5 and videos in our supplementary materials). 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 environments constructed based on UE.

3.2. A3C Algorithm

Following (Mnih et al., 2016), we adopt a popular RL algorithm called Actor-Critic. At time step t, we denote by s_t the observed state, which corresponds to a raw RGB frame. The action set is denoted by $\mathcal{A}$ of size $K = |\mathcal{A}|$. An action, $a_t \in \mathcal{A}$, is drawn from a policy function distribution: $a_t \backsim \pi(\cdot|s_t) \in \mathbb{R}^K$, referred to as an *Actor*. The environment then returns a reward $r_t \in \mathbb{R}$ according to a *reward function* $r_t = g(s_t)$, which will be characterized in Sec. 3.4. The updated state s_{t+1} at next time step $t + 1$ is subject to a certain but unknown state transition function $s_{t+1} = f(s_t, a_t)$, governed by the environment. In this way, we can observe a *trace* consisting of a sequence of tuplets $\tau = \{\ldots, (s_t, a_t, r_t), (s_{t+1}, a_{t+1}, r_{t+1}), \ldots\}$. Meanwhile, we denote by $V(s_t) \in \mathbb{R}$ the expected accumulated reward in the future given state s_t (referred to as *Critic*).

The policy function $\pi(\cdot)$ and the value function $V(\cdot)$ are then jointly modeled by a neural network, as will be discussed in Sec. 3.3. Rewriting them as $\pi(\cdot|s_t; \theta)$ and $V(s_t; \theta')$ with parameters θ and θ', respectively, we can learn θ and θ' over the trace τ with simultaneous stochastic policy gradient and value function regression:

$$\theta \leftarrow \theta + \alpha\big(R_t - V(s_t)\big)\nabla_\theta \log \pi(a_t|s_t) + \beta\nabla_\theta H\big(\pi(\cdot|s_t)\big), \tag{1}$$

$$\theta' \leftarrow \theta' - \alpha\nabla_{\theta'}\frac{1}{2}\big(R_t - V(s_t)\big)^2, \tag{2}$$

where $R_t = \sum_{t'=t}^{t+T-1} \gamma^{t'-t} r_{t'}$ is a discounted sum of future rewards up to T time steps with factor $0 < \gamma \leq 1$, α is the learning rate, $H(\cdot)$ is an entropy regularizer, and β is the regularizer factor.

During training, several threads are launched, each maintaining an independent environment-agent interaction. However, the network parameters are shared across the threads and updated every T time steps asynchronously in a lock-free manner using Eq. (1) in each thread. This kind of many-thread training is reported to be fast yet stable, leading to improved generalization (Mnih et al., 2016). Later in Sec. 3.5, we will introduce environment augmentation techniques to further improve the generalization ability.
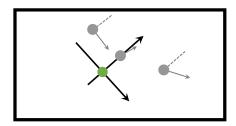
Figure 3. A top view of a map with the local coordinate system. The green dot indicates the agent (tracker). The gray dot indicates the initial position and orientation of an object to be tracked. Three gray dots mean three possible initial configurations. Arrow indicates the orientation of an object. Dashed gray lines are parallel to the y-axis. The outer thick black rectangle represents the boundary. Best viewed in color.

3.3. Network Architecture

The tracker is a ConvNet-LSTM neural network as shown in Fig. 2, where the architecture specification is given in the following table. The FC6 and FC1 correspond to the 6-action policy $\pi\left(\cdot|s_t\right)$ and the value $V(s_t)$, respectively. The screen is resized to $84 \times 84 \times 3$ RGB image as the network input.

Layer#	1	2	3	4	5
Parameters	C8×8-16S4	C4×4-32S2	FC256	LSTM256	FC6 FC1

3.4. Reward Function

To perform active tracking, it is a natural intuition that the reward function should encourage the agent to closely follow the object. In this line of thought, firstly we define a two-dimensional local coordinate system, denoted by $\mathcal{S}$ (see Fig. 3). The x-axis points from the agent's left shoulder to right shoulder, and the y-axis is perpendicular to the x-axis and points to the agent's front. The origin is where the agent is. System $\mathcal{S}$ is parallel to the floor. Secondly, we manage to obtain object's local coordinate (x, y) and orientation a (in radius) with regard to system $\mathcal{S}$.

With a slight abuse of notation, we can now write the reward function as

$$r = A - \left(\frac{\sqrt{x^2 + (y-d)^2}}{c} + \lambda|a| \right), \qquad (3)$$

where $A > 0$, $c > 0$, $d > 0$, $\lambda > 0$ are tuning parameters. In plain English, Eq. (3) says that the maximum reward A is achieved when the object stands perfectly in front of the agent with a distance d and exhibits no rotation (see Fig. 3).

In Eq. (3) we have omitted the time step subscript t without loss of clarity. Also note that the reward function defined in this way does not explicitly depend on the raw visual state. Instead, it depends on certain internal states. Thanks to the APIs provided by virtual environments, we are able to access the interested internal states and develop the desired reward function.
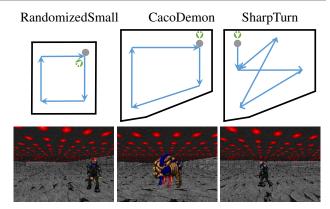

Figure 4. Maps and screenshots of ViZDoom environments. In all maps, the green dot (with white arrow indicating orientation) represents the agent. The gray dot indicates the object. Blue lines are planned paths and black lines are walls. Best viewed in color.

3.5. Environment Augmentation

To make the tracker generalize well, we propose simple yet effective techniques for environment augmentation during training.

For ViZDoom, recall the object's local position and orientation (x, y, a) in system $\mathcal{S}$ described in Sec. 3.4. For a given environment (*i.e.*, a ViZDoom map) with initial (x, y, a), we randomly perturb it N times by editing the map with the ACS script (Kempka et al., 2016), yielding a set of environments with varied initial positions and orientations $\{x_i, y_i, a_i\}_{i=1}^N$. We further allow flipping left-right the screen frame (and accordingly the left-right action). As a result, we obtain $2N$ environments out of one environment. See Fig. 3 for an illustration of several possible initial positions and orientations in the local system $\mathcal{S}$. During the A3C training, we uniformly randomly sample one of the $2N$ environments at the beginning of every episode. As will be seen in Sec. 4.2, this technique significantly improves the generalization ability of the tracker.

For UE, we construct an environment with a character/target walking following a fixed path. To augment the environment, we randomly choose some background objects (*e.g.*, tree or building) in the environment and make them invisible. At the same time, every episode starts from the position, where the agent fails at the last episode. This makes the environment and the starting point different from episode to episode, so the variations of the environment during training are augmented.

4. Experimental Results

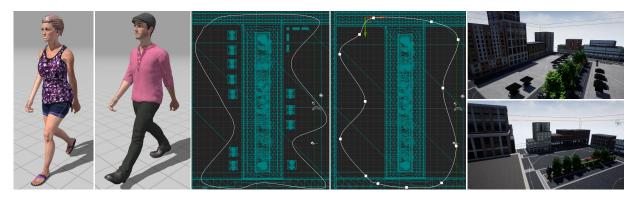The settings are described in Sec. 4.1. The experimental results are reported for the virtual environments ViZDoom

Figure 5. Screenshots of UE environments. From left to right, there are *Stefani*, *Malcom*, *Path1*, *Path2*, *Square1* and *Square2*. Best viewed in color.

(Sec. 4.2) and UE (Sec. 4.3). Qualitative evaluation is performed for real-world scenarios taken from the VOT dataset (Sec. 4.4). To investigate what the tracker has learned, we conduct ablation analysis using a saliency visualization technique (Simonyan et al., 2013) in Sec. 4.5.

4.1. Settings

Environment. A set of environments are produced for both training and testing. For ViZDoom, we adopt a training map as in Fig. 4, left column. This map is then augmented as described in Sec. 3.5 with $N = 21$, leading to 42 environments that we can sample from during training. For testing, we make other 9 maps, some of which are shown in Fig. 4, middle and right columns. In all maps, the path of the target is pre-specified, indicated by the blue lines. However, it is worth noting that the object does not strictly follow the planned path. Instead, it sometimes randomly moves in a "zig-zag" way during the course, which is a built-in game engine behavior. This poses an additional difficulty to the tracking problem.

For UE, we generate an environment named *Square* with random invisible background objects and a target named *Stefani* walking along a fixed path for training. For testing, we make another four environments named as *Square1StefaniPath1 (S1SP1)*, *Square1MalcomPath1 (S1MP1)*, *Square1StefaniPath2 (S1SP2)*, and *Square2MalcomPath2 (S2MP2)*. As shown in Fig. 5, *Square1* and *Square2* are two different maps, *Stefani* and *Malcom* are two characters/targets, and *Path1* and *Path2* are different paths. Note that, the training environment *Square* is generated by hiding some background objects in *Square1*.

For both ViZDoom and UE, we terminate an ep[is]doe when either the accumulated reward drops below a threshold or the episode length reaches a maximum number. In our experiments, we let the reward threshold be -450 and the maximum length be 3000, respectively.

Metric. Two metrics are employed for the experiments. Specifically, *Accumulated Reward* (AR) and *Episode Length* (EL) of each episode are calculated for quantitative evaluation. Note that, the immediate reward defined in Eq. (3) measures the goodness of tracking at some time step, so the metric AR is conceptually much like *Precision* in the conventional tracking literature. Also note that too small AR means a failure of tracking and leads to a termination of the current episode. As such, the metric EL roughly measures the duration of good tracking, which shares the same spirit of the metric *Successfully Tracked Frames* in conventional tracking applications. When letting $A = 1.0$ in Eq. (3), we have that the theoretically maximum AR and EL are both 3000 due to our episode termination criterion. In all the following experiments, 100 episodes are run to report the mean and standard deviation, unless specified otherwise.

Implementation details. We include the implementation details in the supplementary material due to the space constraint.

4.2. Active Tracking in The ViZDoom Environment

We firstly test the active tracker in a testing environment named *Standard*, showing the effectiveness of the proposed environment augmentation technique. The second part is contributed to the experiments in more challenging testing environments which vary from the *Standard* environment with regard to object appearance, background, path, and object distraction. Comparison with a set of traditional trackers is conducted in the last part.

Standard Testing Environment. In Tab. 1, we report the results in an independent testing environment named *Standard* (see supplementary materials for its detailed description), where we compare two training protocols: with (called *RandomizedEnv*) or without (called *SingleEnv*) the augmentation technique as in Sec. 3.5. As can be seen, *RandomizedEnv* performs significantly better than *SingleEnv*.

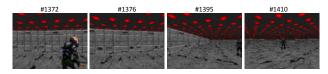We discover that the *SingleEnv* protocol quickly exhausts

#1372 #1376 #1395 #1410

Figure 6. Recovering tracking when the target disappears in the *SharpTurn* environment.

Table 1. Performance of different protocols in the *Standard* testing environment.

Protocol	AR	EL
RandomizedEnv	2547±58	2959±32
SingleEnv	840±602	2404±287

the training capability and obtains the best validation result at about 9×10^6 training iterations. On the contrary, the best validation result of *RandomizedEnv* protocol occurs at 48×10^6, showing that the capacity of the network is exploited better despite the longer training time. In the following experiments, we only report experimental results with the *RandomizedEnv* protocol.

Various Testing Environments. To evaluate the generalization ability of our active tracker, we test it in 8 more challenging environments as in Tab. 2. Comparing to the training environment, they present different target appearances, different backgrounds, more varying paths, and distracting targets. See supplementary materials for the detailed description.

From the 4 categories in Tab. 2 we have findings below.
1) The tracker generalizes well in the case of target appearance changing (*Zombie*, *Cacodemon*).
2) The tracker is insensitive to background variations such as changing the ceiling and floor (*FloorCeiling*) or placing additional walls in the map (*Corridor*).
3) The tracker does not lose a target even when the target takes several sharp turns (*SharpTurn*). Note that in conventional tracking, the target is commonly assumed to move smoothly.
We also observe that the tracker can recover tracking when it accidentally loses the target. As shown in Fig. 6, the target turns right suddenly and the tracker loses it (frame #1372). Although the target completely disappears in the image, the tracker takes a series of *turn-right* actions (frame #1376 to #1394). It rediscovers the target (frame #1410), and continues to track steadily afterwards. We believe that this capability attributes to the LSTM unit which takes into account historical states when producing current outputs.
Our tracker performs well when the target walks counterclockwise (*Counterclockwise*), indicating that the tracker does not work by simply memorizing the turning pattern.
4) The tracker is insensitive to a distracting object (*Noise1*), even when the "bait" is very close to the path (*Noise2*).

The proposed tracker shows satisfactory generalization in various unseen environments. Readers are encouraged to

Table 2. Performance of the proposed active tracker in different testing environments.

Environment	AR	EL
CacoDemon	2415±71	2981±10
Zombie	2386±86	2904±40
FloorCeiling	1504 ± 158	2581 ± 84
Corridor	2636 ± 34	2983 ± 17
SharpTurn	2560±34	2987±12
Counterclockwise	2537±58	2964±23
Noise1	2493±72	2977±14
Noise2	2313±103	2855±56

Table 3. Comparison with traditional trackers. The best results are shown in bold.

Environment	Tracker	AR	EL
Standard	MIL	-454.2 ± 0.3	743.1 ± 21.4
	Meanshift	-452.5 ± 0.2	553.4 ± 2.2
	KCF	-454.1 ± 0.2	228.4 ± 5.5
	Correlation	-453.6 ± 0.2	252.7 ± 16.6
	Active	**2457±58**	**2959±32**
SharpTurn	MIL	-453.3 ± 0.2	388.3 ± 15.5
	Meanshift	-454.4 ± 0.3	250.1 ± 1.9
	KCF	-452.4 ± 0.2	199.2 ± 5.7
	Correlation	-453.0 ± 0.2	186.3 ± 6.0
	Active	**2560±34**	**2987±12**
Cacodemon	MIL	-453.5 ± 0.2	540.6 ± 18.2
	Meanshift	-452.9 ± 0.2	484.3 ± 9.4
	KCF	-454.5 ± 0.3	263.1 ± 6.2
	Correlation	-453.3 ± 0.2	155.8 ± 1.9
	Active	**2451±71**	**2981±10**

watch more result videos provided in our supplementary materials.

Comparison with Simulated Active Trackers. In a more extensive experiment we compare the proposed tracker with a few traditional trackers. These trackers are originally developed for passive tracking applications. Particularly, the MIL (Babenko et al., 2009), Meanshift (Comaniciu et al., 2000), KCF (Henriques et al., 2015), and Correlation (Danelljan et al., 2014) trackers are employed for comparison. We implement them by directly invoking the interface from OpenCV (Ope) (MIL, KCF and Meanshift trackers) and Dlib (Dli) (Correlation tracker).

To make the comparison feasible, we add to the passive tracker an additional PID-like module for the camera control, enabling it to interact with the environment (see Fig. 1, Right). In the first frame, a manual bounding box must be given to indicate the object to be tracked. For each subsequent frame, the passive tracker then predicts a bounding box, which is passed to the "Camera Control" module. Finally, the action is produced by "pulling back" the target to its position in a previous frame (see supplementary materials for the details of the implementation). For a fair comparison
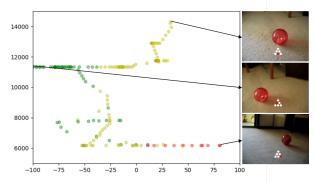
Figure 7. Actions output from the proposed active tracker of the Woman (top) and Sphere (bottom) sequences.

with the proposed active tracker, we employ the same action set $\mathcal{A}$ as described in Sec. 3.

Armed with this camera-control module, the performance of traditional trackers is compared with the active tracker in *Standard*, *SharpTurn* and *Cacodemon*. The results in Tab. 3 show that the end-to-end active tracker beats the simulated "active" trackers by a significant gap. We investigate the tracking process of these trackers and find that they lose the target soon. The Meanshift tracker works well when there is no camera shift between continuous frames, while in the active tracking scenario it loses the target soon. Both KCF and Correlation trackers seem not capable of handling such a large camera shift, so they do not work as well as the case in passive tracking. The MIL tracker works reasonably in the active case, while it easily drifts when the object turns suddenly.

Recalling Fig. 6, another reason of our tracker beating the traditional trackers is that our tracker can quickly discover the target again in the case that it is missed. While the simulated active trackers can hardly recover from failure cases.

4.3. Active Tracking in The UE Environment

We firstly compare models trained with randomized environment and single environment. Then we test our active tracker in four environments and also compare it against

Table 4. Performance of different protocols in *S2MP2*.

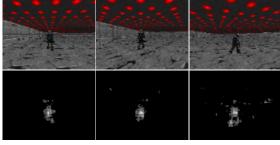| Protocol	AR	EL
RandomizedEnv	1203.6±1428.4	2603.6±503.0
SingleEnv	-453.4±1.5	461.9±180.0

Figure 8. Saliency maps learned by the tracker. The top row shows input observations, and the bottom row shows their corresponding saliency maps. The corresponding actions of these input images are *turn-right-and-move-forward, turn-left-and-move-forward* and *turn-left-and-move-forward*, respectively. These saliency maps clearly show the focus of the tracker.

traditional trackers.

RandomizedEnv versus SingleEnv. Based on the *Square* environment, we train two models individually by the *RandomizedEnv* protocol (random number of invisible background objects and starting point) and *SingleEnv* protocol (fixed environment). They are tested in the *S2MP2* environment, where the map, target, and the path are unseen during training. As shown in Tab. 4, similar results are obtained as those in Tab. 1. We believe that the improvement benefits from the environment randomness brought by the proposed environment augmentation techniques. In the following, we only report the results of *RandomizedEnv* protocol.

Various Testing Environments. To intensively investigate the generalization ability of the active tracker, we test it in four different environments and present the results in Tab. 5. We compare it with the simulated active trackers described in Sec. 4.2, as well as one based on the long-term TLD tracker (Kalal et al., 2012).

According to the results in Tab. 5 we conduct the following analysis: 1) Comparison between *S1SP1* and *S1MP1* shows that the tracker generalizes well even when the model is trained with target Stefani, revealing that it does not overfit to a specialized appearance. 2) The active tracker performs well when changing the path (*S1SP1* versus *S1SP2*), demonstrating that it does not act by memorizing specialized path. 3) When we change the map, target, and path at the same time (*S2MP2*), though the tracker could not seize the target as accurately as in previous environments (the AR value drops), it can still track objects robustly (comparable EL value as in previous environments), proving its superior generalization potential. 4) In most cases, the proposed tracker outperforms the simulated active tracker, or achieves compa-

Table 5. Comparison with traditional trackers. The best results are shown in bold.

Environment	Tracker	AR	EL
	MIL	-453.8 ±0.8	601.4 ± 300.9
	Meanshift	-454.1±1.3	628.6±111.2
S1SP1	KCF	-453.6±2.5	782.4±136.1
	Correlation	-454.9±0.9	1710.4±417.0
	TLD	-453.6±1.3	376.0±70.9
	Active	**2495.7±12.4**	**3000.0±0.0**
	MIL	-358.7±189.4	1430.0±825.3
	Meanshift	708.3±3.3	**3000.0±0.0**
S1MP1	KCF	-453.6±2.6	797.4±37.5
	Correlation	-453.5±1.3	1404.4±131.1
	TLD	-453.4±2.0	651.0±54.5
	Active	**2106.0±29.3**	**3000.0±0.0**
	MIL	-452.4±0.7	420.2±104.9
	Meanshift	-453.0±1.8	630.2±223.8
S1SP2	KCF	-453.9±1.5	594.0±378.8
	Correlation	-452.4±0.4	293.8±97.4
	TLD	-454.7±1.8	218.0±26.0
	Active	**2162.5±48.5**	**3000.0±0.0**
	MIL	-453.1±0.9	749.0±301.0
	Meanshift	726.5±10.8	**3000.0±0.0**
S2MP2	KCF	-452.4±1.0	247.8±18.8
	Correlation	-215.0±475.3	1571.6±919.1
	TLD	-453.1±1.8	208.8±33.1
	Active	**740.0±577.4**	2565.3±339.3

rable results if it is not the best. The results of the simulated active tracker also suggest that it is difficult to tune a unified camera-control module for them, even when a long term tracker is adopted (see the results of TLD). However, our work exactly sidesteps this issue by training an end-to-end active tracker.

4.4. Active Tracking in Real-world Scenarios

To evaluate how the active tracker performs in real-world scenarios, we take the network trained in a UE environment and test it on a few video clips from the VOT dataset (Kristan et al., 2016). Obviously, we can by no means control the camera action for a recorded video. However, we can feed in the video frame sequentially and observe the output action predicted by the network, checking whether it is consistent with the actual situation.

Fig. 7 shows the output actions for two video clips named Woman and Sphere, respectively. The horizontal axis indicates the position of the target in the image, with a positive (negative) value meaning that a target in the right (left) part. The vertical axis indicates the size of the target, *i.e.*, the area of the ground truth bounding box. Green and red dots indicate turn-left/turn-left-and-move-forward and turn-right/turn-right-and-move-forward actions, respectively. Yellow dots represent No-op action. As the figure

show, 1) When the target resides in the right (left) side, the tracker tends to turn right (left), trying to move the camera to "pull" the target to the center. 2) When the target size becomes bigger, which probably indicates that the tracker is too close to the target, the tracker outputs no-op actions more often, intending to stop and wait the target to move farther.

We believe that the qualitative evaluation shows evidence that the active tracker, learned from purely the virtual environment, is able to output correct actions for camera control in real-world scenarios. Due to the constraint of space, we include more results of the real-world scenarios in the supplementary materials.

4.5. Action Saliency Map

We are curious about what the tracker has learned so that it leads to good performance. To this end, we follow the method in (Simonyan et al., 2013) to generate a saliency map of the input image with regard to a specific action. Making it more specific, an input frame s_i is fed into the tracker and forwarded to output the policy function. An action a_i will be sampled subsequently. Then the gradient of a_i with regard to s_i is propagated backwards to the input layer, and a saliency map is generated. This process calculates exactly which part of the original input image influences the corresponding action with the greatest magnitude.

Note that the saliency map is image specific, *i.e.*, for each input image a corresponding saliency map can be derived. Consequently, we can observe how the input images influence the tracker's actions. Fig. 8 shows a few pairs of input image and corresponding saliency map. The saliency maps consistently show that the pixels corresponding to the object dominate the importance to actions of the tracker. It indicates that the tracker indeed learns how to find the target.

5. Conclusion

We proposed an end-to-end active tracker via deep reinforcement learning. Unlike conventional passive trackers, the proposed tracker is trained in simulators, saving the efforts of human labeling or trail-and-errors in real-world. It shows good generalization to unseen environments. The tracking ability can potentially transfer to real-world scenarios.

References

Dlib. http://dlib.net/.

Opencv. http://opencv.org/.

Vizdoom. http://vizdoom.cs.put.edu.pl/.

Unreal engine. https://www.unrealengine.com/en-US/blog.

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

Bertinetto, Luca, Valmadre, Jack, Golodetz, Stuart, Miksik, Ondrej, and Torr, Philip HS. Staple: Complementary learners for real-time tracking. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1401–1409, 2016a.

Bertinetto, Luca, Valmadre, Jack, Henriques, Joao F, Vedaldi, Andrea, and Torr, Philip HS. Fully-convolutional siamese networks for object tracking. In *European Conference on Computer Vision*, pp. 850–865. Springer, 2016b.

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

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

Caicedo, Juan C and Lazebnik, Svetlana. Active object localization with deep reinforcement learning. In *International Conference on Computer Vision*, pp. 2488–2496, 2015.

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

Choi, Jongwon, Jin Chang, Hyung, Yun, Sangdoo, Fischer, Tobias, Demiris, Yiannis, and Young Choi, Jin. Attentional correlation filter network for adaptive visual tracking. In *The IEEE Conference on Computer Vision and Pattern Recognition*, July 2017b.

Comaniciu, Dorin, Ramesh, Visvanathan, and Meer, Peter. Real-time tracking of non-rigid objects using mean shift. In *The IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pp. 142–149, 2000.

Cui, Zhen, Xiao, Shengtao, Feng, Jiashi, and Yan, Shuicheng. Recurrently target-attending tracking. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1449–1458, 2016.

Danelljan, Martin, Häger, Gustav, Khan, Fahad, and Felsberg, Michael. Accurate scale estimation for robust visual tracking. In *British Machine Vision Conference*, 2014.

Danelljan, Martin, Bhat, Goutam, Shahbaz Khan, Fahad, and Felsberg, Michael. Eco: Efficient convolution operators for tracking. In *The IEEE Conference on Computer Vision and Pattern Recognition*, July 2017.

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

Girshick, Ross, Donahue, Jeff, Darrell, Trevor, and Malik, Jitendra. Rich feature hierarchies for accurate object detection and semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

Hare, Sam, Golodetz, Stuart, Saffari, Amir, Vineet, Vibhav, Cheng, Ming-Ming, Hicks, Stephen L, and Torr, Philip HS. Struck: Structured output tracking with kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):2096–2109, 2016.

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

Hu, Weiming, Li, Xi, Luo, Wenhan, Zhang, Xiaoqin, Maybank, Stephen, and Zhang, Zhongfei. 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.

Huang, Chen, Lucey, Simon, and Ramanan, Deva. Learning policies for adaptive tracking with deep feature cascades. In *International Conference on Computer Vision*, 2017.

Jie, Zequn, Liang, Xiaodan, Feng, Jiashi, Jin, Xiaojie, Lu, Wen, and Yan, Shuicheng. Tree-structured reinforcement learning for sequential object localization. In *Advances in Neural Information Processing Systems*, pp. 127–135, 2016.

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

Kempka, Michał, Wydmuch, Marek, Runc, Grzegorz, Toczek, Jakub, and Jaśkowski, Wojciech. Vizdoom: A doom-based ai research platform for visual reinforcement learning. *arXiv preprint arXiv:1605.02097*, 2016.

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

Kristan, Matej, Matas, Jiri, Leonardis, Aleš, Vojir, Tomas, Pflugfelder, Roman, Fernandez, Gustavo, Nebehay, Georg, Porikli, Fatih, and Čehovin, Luka. A novel performance evaluation methodology for single-target trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2137–2155, Nov 2016.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems*, pp. 1097–1105. 2012.

Long, Jonathan, Shelhamer, Evan, and Darrell, Trevor. Fully convolutional networks for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

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

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

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

Murray, Don and Basu, Anup. Motion tracking with an active camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):449–459, 1994.

Qiu, Weichao, Zhong, Fangwei, Zhang, Yi, Qiao, Siyuan, Xiao, Zihao, Kim, Tae Soo, Wang, Yizhou Wang, and Yuille, Alan. Unrealcv: Virtual worlds for computer vision. *ACM Multimedia Open Source Software Competition*, 2017.

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

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

Simonyan, Karen, Vedaldi, Andrea, and Zisserman, Andrew. Deep inside convolutional networks: Visualising image classification models and saliency maps. *International Conference on Learning Representations*, 2013.

Supancic, III, James and Ramanan, Deva. Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning. In *International Conference on Computer Vision*, 2017.

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

Valmadre, Jack, Bertinetto, Luca, Henriques, Joao, Vedaldi, Andrea, and Torr, Philip H. S. End-to-end representation learning for correlation filter based tracking. In *The IEEE Conference on Computer Vision and Pattern Recognition*, July 2017.

Wang, Lijun, Ouyang, Wanli, Wang, Xiaogang, and Lu, Huchuan. Stct: Sequentially training convolutional networks for visual tracking. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1373–1381, 2016.

Wang, Naiyan and Yeung, Dit-Yan. Learning a deep compact image representation for visual tracking. In *Advances in Neural Information Processing Systems*, pp. 809–817, 2013.

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

Wu, Yuxin and Tian, Yuandong. Training agent for first-person shooter game with actor-critic curriculum learning. In *International Conference on Learning Representations*, 2017.

Yun, Sangdoo, Choi, Jongwon, Yoo, Youngjoon, Yun, Kimin, and Young Choi, Jin. Action-decision networks for visual tracking with deep reinforcement learning. In *The IEEE Conference on Computer Vision and Pattern Recognition*, July 2017.

Zhong, Fangwei, Qiu, Weichao, Yan, Tingyun, Alan, Yuille, and Wang, Yizhou. Gym-unrealcv: Realistic virtual worlds for visual reinforcement learning, 2017.

Zhu, Gao, Porikli, Fatih, and Li, Hongdong. Beyond local search: Tracking objects everywhere with instance-specific proposals. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pp. 943–951, 2016.

Supplementary Materials for "End-to-end Active Object Tracking via Reinforcement Learning"

Wenhan Luo [* 1]   Peng Sun [* 1]   Fangwei Zhong [2]   Wei Liu [1]   Tong Zhang [1]   Yizhou Wang [2]

1. Introduction

In this supplementary material, we first give the implementation details of the proposed method. We then present the experimental results which are not included in the main texts due to space limitation. Particularly, more visual results with regard to video clips of Woman and Sphere will be provided in Sec. 3. Sec. 4 provides results of the proposed tracker on other video clips from the VOT dataset. The PID-like camera control module we developed for the simulated active tracker with the traditional trackers is illustrated in Sec. 5. Additional notes of ViZDoom experiments are presented in Sec. 6.

2. Implementation Details

The network parameters are updated with Adam optimization, with the initial learning rate $\alpha = 0.0001$. The regularizer factor $\beta = 0.01$ and the reward discount factor $\gamma = 0.99$. The parameter updating frequency is $T = 20$, and the maximum global iteration for training is 100×10^6. Validation is performed every 70 seconds and the best validation network model is applied to report performance in testing environments.

3. More Results of Woman and Sphere

Figures 1 and 2 show actions individually according to our discrete actions. The actions are grouped as Forward (Move-forward), Left (including both Turn-left and Turn-left-and-move-forward actions in our action space), Right (including both Turn-right and Turn-right-and-move-forward actions in our action space), and Stop (No-op). By doing so, the results can better indicate the potential of our tracker in real-world scenarios. Though trained in pure virtual environments, it

---

*Equal contribution [1]Tencent AI Lab [2]Peking University. Correspondence to: Wenhan Luo <whluo.china@gmail.com>, Peng Sun <pengsun000@gmail.com>, Fangwei Zhong <zfw@pku.edu.cn>, Wei Liu <wl2223@columbia.edu>, Tong Zhang <tongzhang@tongzhang-ml.org>, Yizhou Wang <yizhou.Wang@pku.edu.cn>.

can predict the correct actions to control the camera, further "pulling" the target to the center of the image.

4. Results of Other Video Clips

Showing its potential even trained in virtual UE4 environment, we again test it intensively on other video clips from the VOT dataset. Figures from 4 to 7 show the actions output by the active tracker on videos named *Book*, *Girl*, *Iceskater* and *Iceskater1*.

As the same as the main submission file, green dots indicate actions of Turn-left or Turn-left-and-move-forward, and red dots represent actions of Turn-right or Turn-right-and-move-forward. Yellow dots indicate the No-op action. Blue dots respond to the Move-forward action. All the left and right actions are predicted correctly regarding the position of the target in the image (horizontal axis) except that the tracker predicts to turn left while the target is in the right part of the image (horizontally ranging from 50 to 150 in Fig. 5 and horizontally ranging from 60 to 80 in Fig. 6). In the case of *Iceskater* the incorrect predictions are made when the size of the target is very small. At that moment the camera is actually shooting a long-range perspective. Thus the target is not so salient considering the audiences, which may lead to the incorrect predictions. The gap between real-world scenarios and photo-realistic virtual environments can also result in such failures.

5. PID-like Camera Control

The PID camera controller we developed is similar to a Proportional-integral-derivative controller. It decides an action sequence based on the specific error.

As shown in Fig. 3, we assume that the optimal tracking means that the rectangle is in the center of the image and takes about 20% of the pixel space (the red bounding box in the figure). This means that we should control the camera to meet these conditions.

Formally, we denote the image width and height by W and H, and the ideal position by (X_{ideal}, Y_{ideal}), respectively.
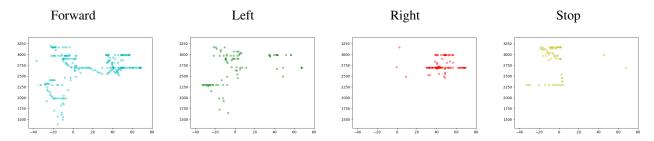
Figure 1. Visual results of individual actions of the proposed active tracker on the video clip of Woman. From left to right, they are actions of Forward (move-forward), Left (turn-left/turn-left-and-move-forward), Right (turn-right/turn-right-and-move-forward) and Stop (no-op).
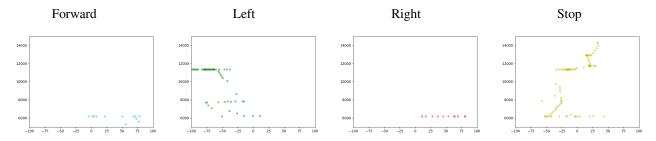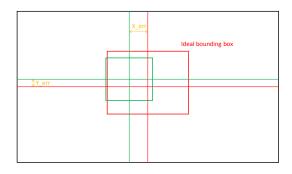

Figure 2. Visual results of individual actions of the proposed active tracker on the video clip of Sphere. From left to right, they are actions of Forward (move-forward), Left (turn-left/turn-left-and-move-forward), Right (turn-right/turn-right-and-move-forward) and Stop (no-op).

Figure 3. An example to illustrate errors.

We have the following parameters,

$$X_{ideal} = W/2,$$
$$Y_{ideal} = H/2, \qquad (1)$$
$$A_{ideal} = W * H * 20\%,$$

where A_{ideal} is the ideal area the object bounding box takes.

Let us assume that the result of a tracker is characterized by four parameters $\{X_{bounding}, Y_{bounding}, W_{bounding}, H_{bounding}\}$. Then we have the following errors,

$$X_{err} = X_{bounding} - X_{ideal},$$
$$Y_{err} = Y_{bounding} - Y_{ideal}, \qquad (2)$$
$$Z_{err} = A_{ideal} - (W_{bounding} * H_{bounding}).$$

Note that, Z_{err} (depth) is tied to the size of the object. Intuitively, X_{err} measures how far the bounding box is horizontally away from the desired position. In the case of Fig. 3, X_{err} is negative, meaning that the camera should move left so the object is closer to the center of the image.

In general we have the following commands,

$X_{err} < 0$ means that the tracker is to the right and needs to turn left (decreasing X).

$X_{err} > 0$ means that the tracker is to the left and needs to turn right (increasing X).

$Y_{err} < 0$ means that the tracker is too far away and needs to speed up (decreasing Y).

$Y_{err} > 0$ means that the tracker is too close and needs to slow down (increasing Y).

$Z_{err} < 0$ means that the tracker is too close and needs to slow down.

$Z_{err} > 0$ means that the tracker is too far away and needs to speed up.

Note that, the action space we adopted does not include actions like "look up" or "look down". At the same time, in the view of an ideal tracker, the moving forward of the target will lead to the decrease of Y value, so we intuitively map the change of Y_{err} to the status of distance between the target and the tracker.

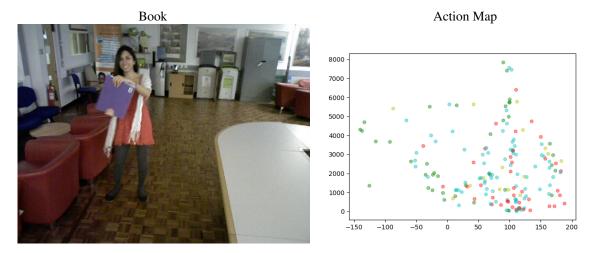Observing this, we then map these intuitive commands to

Book

Action Map

Figure 4. Left: an exemplar image of video "Book". Right: actions output by our tracker.
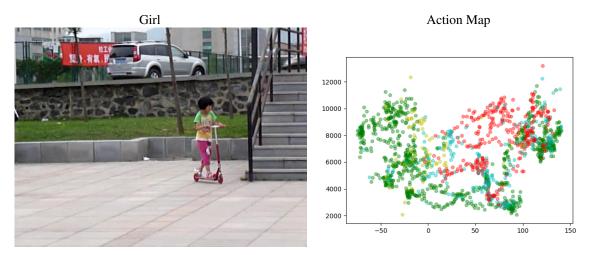
Girl

Action Map

Figure 5. Left: an exemplar image of video "Girl". Right: actions output by our tracker.

discrete actions and send actions to the environment (ViZ-Doom or UnrealCV). The quantity associated with a specific action depends on the quantity of a specific error. For example, if $X_{err} > 0$ and it is too large, then the quantity of turning right is also large.

This creates a procedure which sounds a lot like a PID controller: observe a state, figure out an error and create a control sequence to reduce the error, and then repeat the process over and over again.

6. Additional Notes

6.1. Description of Various Testing Environments of ViZDoom

Detailed description of the various ViZDoom testing environments are as follows. Specifically, they are obtained by modifying the *Standard* environment in the following

aspects:

1) Change the appearance of the target. Specifically, we have two environments named *CacoDemon* and *Zombie* with targets *CacoDemon* and *Zombie*, respectively.

2) Revise the background in the environment. We have an environment named *FloorCeiling* with different textures in ceiling and floor, and an environment named *Corridor* with a corridor structure.

3) Modify the path. The path in *SharpTurn* is composed of several sharp acute angles while clockwise path is changed to a counterclockwise one in *Counterclockwise*.

4) Add distractions. *Noise1* is formed by placing a same monster (stationary) near the path along which the target walks. *Noise2* is almost the same as *Noise1*, except that the distracting monster is closer to the path.
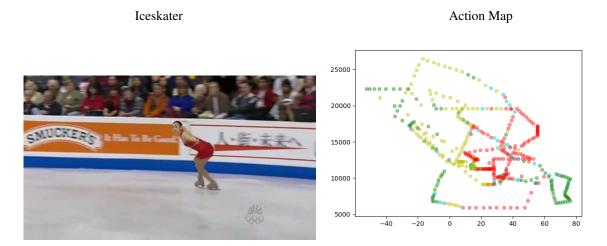
Iceskater

Action Map

Figure 6. Left: an exemplar image of video "Iceskater". Right: actions output by our tracker.
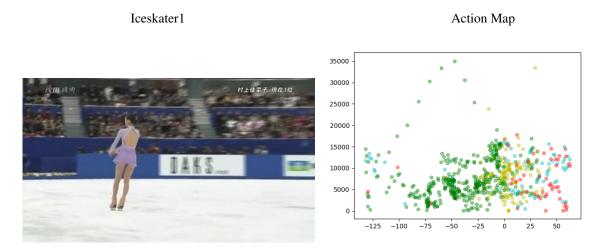
Iceskater1

Action Map

Figure 7. Left: an exemplar image of video "Iceskater1". Right: actions output by our tracker.