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

[*]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>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

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.

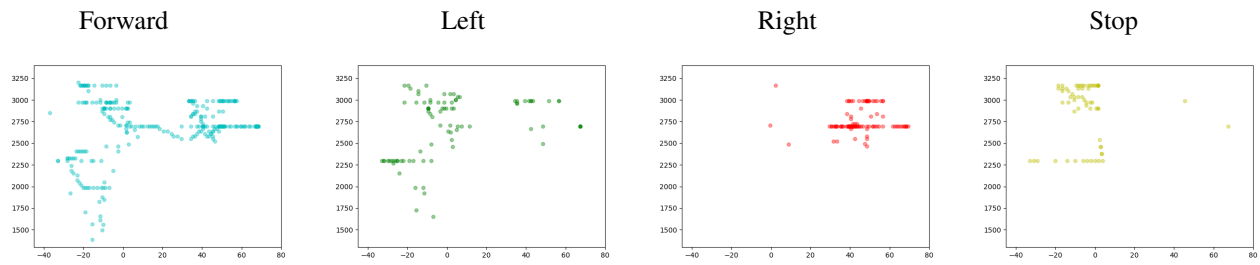

Figure 1. Visual results of individual actions of the proposed active tracker on the video clip of Woman. From left to right, they are actions of Forward (move-forward), Left (turn-left/turn-left-and-move-forward), Right (turn-right/turn-right-and-move-forward) and Stop (no-op).

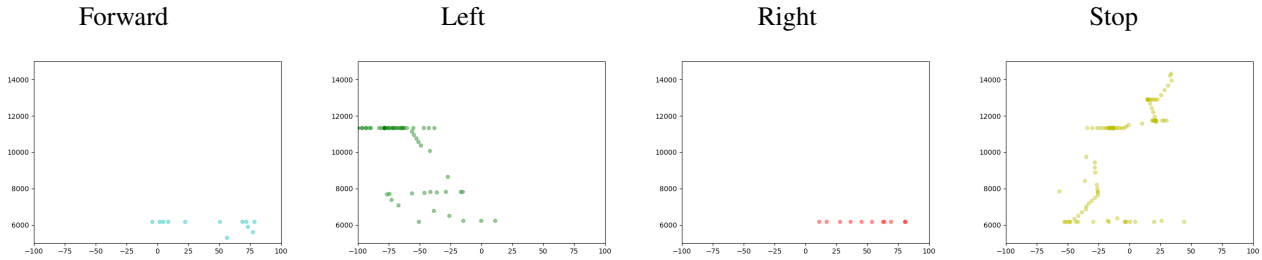

Figure 2. Visual results of individual actions of the proposed active tracker on the video clip of Sphere. From left to right, they are actions of Forward (move-forward), Left (turn-left/turn-left-and-move-forward), Right (turn-right/turn-right-and-move-forward) and Stop (no-op).

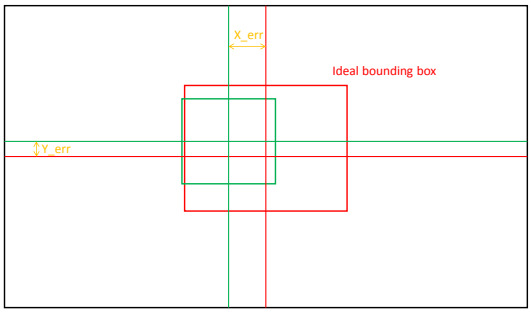

Figure 3. An example to illustrate errors.

We have the following parameters,

$$\begin{aligned} X_{ideal} &= W/2, \\ Y_{ideal} &= H/2, \\ A_{ideal} &= W * H * 20\%, \end{aligned} \tag{1}$$

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,

$$\begin{aligned} X_{err} &= X_{bounding} - X_{ideal}, \\ Y_{err} &= Y_{bounding} - Y_{ideal}, \\ Z_{err} &= A_{ideal} - (W_{bounding} * H_{bounding}). \end{aligned} \tag{2}$$

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

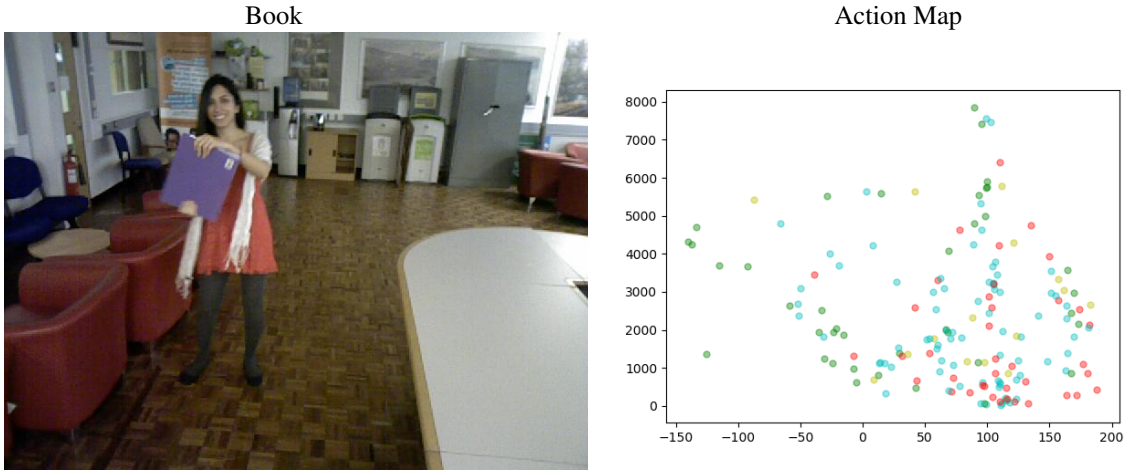

Figure 4. Left: an exemplar image of video "Book". Right: actions output by our tracker.

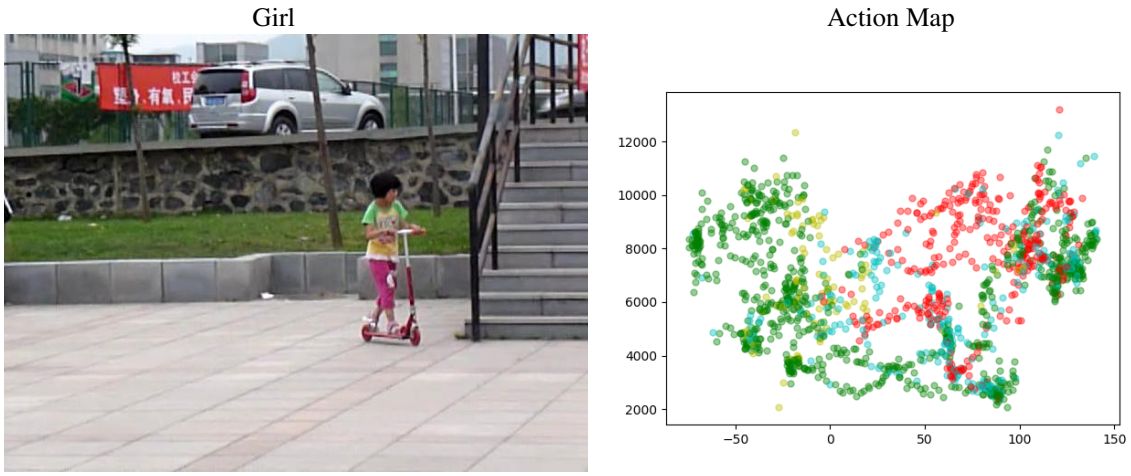

Figure 5. Left: an exemplar image of video "Girl". Right: actions output by our tracker.

discrete actions and send actions to the environment (ViZDoom 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

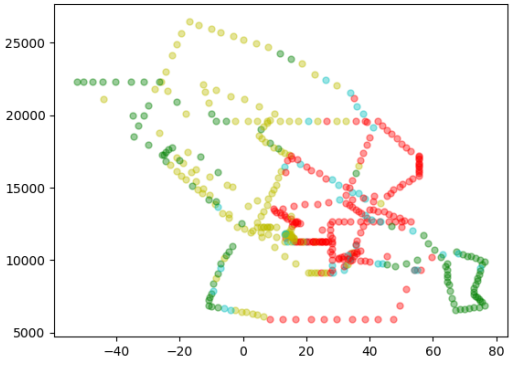

Figure 6. Left: an exemplar image of video "Iceskater". Right: actions output by our tracker.

Iceskater1

Action Map

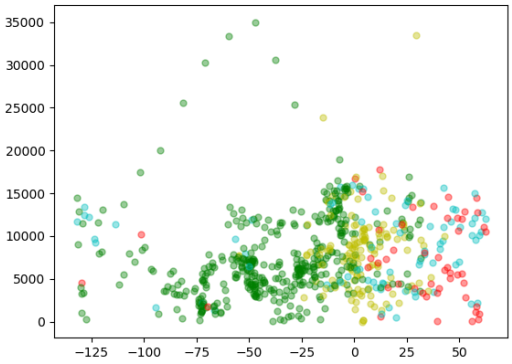

Figure 7. Left: an exemplar image of video "Iceskater1". Right: actions output by our tracker.