

# Visual Object Tracking in Drone Images with Deep Reinforcement Learning

Derya Gözen  
Bilkent University  
Email: derya.gozen@bilkent.edu.tr

Sedat Özer  
Bilkent University  
Email: sedat@cs.bilkent.edu.tr

**Abstract**—There is an increasing demand on utilizing camera equipped drones and their applications in many domains varying from agriculture to entertainment and from sports events to surveillance. In such drone applications, an essential and a common task is tracking an object of interest visually. Drone (or UAV) images have different properties when compared to the ground taken (natural) images and those differences introduce additional complexities to the existing object trackers to be directly applied on drone applications. Some important differences among those complexities include (i) smaller object sizes to be tracked and (ii) different orientations and viewing angles yielding different texture and features to be observed. Therefore, new algorithms trained on drone images are needed for the drone-based applications. In this paper, we introduce a deep reinforcement learning (RL) based single object tracker that tracks an object of interest in drone images by estimating a series of actions to find the location of the object in the next frame. This is the first work introducing a single object tracker using a deep RL-based technique for drone images. Our proposed solution introduces a novel reward function that aims to reduce the total number of actions taken to estimate the object's location in the next frame and also introduces a different backbone network to be used on low resolution images. Additionally, we introduce a set of new actions into the action library to better deal with the above-mentioned complexities. We compare our proposed solutions to a state of the art tracking algorithm from the recent literature and demonstrate up to 3.87% improvement in precision and 3.6% improvement in IoU values on the VisDrone2019 data set. We also provide additional results on OTB-100 data set and show up to 3.15% improvement in precision on the OTB-100 data set when compared to the same previous state of the art algorithm. Lastly, we analyze the ability to handle some of the challenges faced during tracking, including but not limited to occlusion, deformation, and scale variation for our proposed solutions.

**Index Terms**—Object Tracking, Visual Object Tracking, Deep Reinforcement Learning, Aerial Images, UAV videos

## I. INTRODUCTION

Drone or unmanned aerial vehicle (UAV) applications are on the rise with the recent advances in both robotics and computer vision, recently. The growing numbers of drone applications in many domains such as entertainment, aerial photography, meteorology, maintenance or delivery [21], and the widespread use of cameras on drones increased the demand and interest on computer vision based solutions. Tracking an object of interest visually in drone videos is a common and an essential problem in computer vision and an emerging component in many drone applications. While tracking an object in standard (ground taken) videos has been widely studied, it has been only a

recent interest in drone videos due to the lack of available large drone data sets until recently. The introduction of recent drone data sets such as VisDrone [32] is aimed to solve the data-related issues of visual object tracking in drone images.

In general, the problem of tracking a single object is widely known as “single object tracking” or “visual object tracking” in the relevant literature and this problem has been widely studied on standard videos including VOT [14]–[16] and OTB data sets [26], [27]. However, visual tracking is a recent topic in drone (aerial) images and many of the main challenges are yet to be addressed. Consequently, there are only a limited number of relevant papers available on the subject.

While there have been various techniques proposed for visual object tracking in computer vision, a common and a recent research focus has been on developing RL based solutions as in [6], [23], [29], [30], due to their potential on learning meaningful actions to define how to move a bounding box of an object of interest from the current frame to the next one in the exploratory manner of the RL framework.

Many RL-based trackers, as in [29], focus on providing a reward to the RL algorithm after collecting all the action sequences in a given video clip. In that approach, the algorithm has to first obtain all the action sequences for all the frames in the clip and creates a batch of those (action) sequences. That batch of action sequences is, then, used to reward the algorithm. However, we propose rewarding the algorithm after obtaining each action sequence individually, i.e. as soon as the action sequence reaches to the terminal action (end of the sequence) for a frame. That way, we can force our algorithm to learn taking more accurate actions for each frame in a video and that approach would yield an improved performance as our experiments demonstrate.

Furthermore, we emphasize on that it is important to choose the right action set for the picked data domain, when action-based RL networks are used such as [29]. For example, in drone data sets, since the objects typically appear smaller than they appear in natural (ground) videos, it is important to consider going along each direction including the diagonal directions, when taking an action. Therefore, we also study the performance of including (and deciding on) the right actions on drone data set. In this paper, to compare our results, we chose using ADNet from [29] as being the closest recent relevant work to ours and compare our results to that in our experiments. To the our best knowledge, this is the first work

that introduces a deep RL based visual object tracker for drone videos with reported results on VisDrone data set.

In this paper, our main contributions include:

- introduction of a novel reinforcement learning based deep single object tracker for drone videos,
- introduction of a novel reward function for an improved performance,
- introduction of new action types for drone data sets,
- testing our algorithm on two data sets: VisDrone2019 and OTB-100.

The rest of our paper is organized as follows: related work is reviewed in the next section (Section II), then we give a brief introduction to our baseline and backbone network (ADNet) in Section III. In Section IV, we introduce our multiple action-sequence-based RL trackers. Then, Section V gives the details of the training process of action-sequence based RL networks, which consists of three main stages: supervised learning, reinforcement learning and online adaptation in tracking. Then, the experiments are conducted on OTB-100 and VisDrone2019 data sets. Our experiments are presented in Section VI. Finally, we analyze various aspects of the models and discuss our results in Section VII.

## II. RELATED WORK

Object tracking is one of the core and necessary computer vision tasks in many vision applications. Its wide range of application areas includes but not limited to video surveillance, human-computer interaction and behaviour analysis. Earlier works focused on using hand-crafted features to track objects as in [9], [12], [13]. Recently, correlation filter based techniques are also widely used in object tracking because of their efficient and robust results [3], [5].

When compared to the deep learning based techniques, typically, the conventional tracking methods using hand-crafted features are more likely to have difficulties in finding the location of the object of interest due to various tracking-related obstacles. Those obstacles include appearance or disappearance of objects, occlusion, background clutter, appearance similarity, motion blur and illumination change. With the rise of deep learning techniques and their ability to provide a stronger functional approximation and (deep) feature representations, deep learning based techniques have recently drawn attention of the researchers working on visual object tracking. Some of the earlier studies use Convolutional Neural Networks (CNNs) for feature extraction to replace the hand-crafted features of the correlation filter methodologies [7], [8], [18], [19]. MDNet [19] uses a pre-trained CNN with ground truths that are annotated manually to generate target representations and assess the candidate bounding boxes sampled randomly around the previous state in order to find the optimal location. A different deep learning technique, VITAL algorithm [22] is proposed to handle high spatial overlapping and extreme class imbalance between positive and negative samples via adversarial learning.

Moreover, Siamese networks are proposed for object tracking in [2], [10], [24]. SiamFC [2] and Siamese INstance search

Tracker (SINT) [24] both use Siamese deep neural networks to learn the function bridging the initial patch position of the target, and the candidate in the following frames.

Deep learning techniques in object tracking has also encouraged deep reinforcement learning to make quite a rapid progress in the recent years. For instance, some studies focus on sequential decision-making processes, which learn the policies with deep reinforcement learning algorithms [23], [29]–[31]. Those approaches mainly aim to find the optimal decision-making policy with reinforcement learning by maximizing the performance of long-term tracking. Action-Decision Network (ADNet) [29] introduces a novel tracker controlled by sequential actions obtained from deep reinforcement learning stage [29]. Additionally, ADNet combines supervised learning and reinforcement learning in its training. Another model that relies on sequential actions is Deep Q-Network for Visual Tracking (TrackDQN) [28], which is able to make decisions in order to move the target from the current state. TrackDQN leverages Q-Network to obtain the set of actions to move the current bounding box to the next frame. Although ADNet has 11 actions in its action set under the assumption that the scale of the bounding box remains the same throughout tracking, TrackDQN has 9 actions including aspect ratio changes.

Although the literature mentioned above tend to use standard and ground-taken video sequences, visual tracking has become an emerging topic of research in aerial videos recorded by drones or UAVs. In the last decade, there is a number of studies that propose state-of-the-art methods to track objects in UAV scenarios [1], [17], [25]. During the development of tracking methods on specifically drone videos, the researchers face many additional challenges and those challenges include limited availability of aerial data sets, smaller object size, a large variance in objects' scale, orientation, different viewing angle yielding different shape and features, limited hardware capacity and constant movement of both the UAV camera and the targets. As a result, studies focusing on visual tracking on drone videos remain very limited in number. Besides, most of them focus on detection based tracking as in [11], [20], [21]. Within the scope of drone data sets, tracking by detection appears to be a widely used approach considering the fact that object detection has proved a rapid progress in the recent years. However, there are not many work introducing RL based trackers for visual object tracking in drone videos (as of the submission date of this paper, we could not find any work). That is where our work differs from the other relevant visual object trackers for drone videos the most.

## III. OVERVIEW OF OUR BASELINE NETWORK: ADNET

We choose an existing and a recent RL-based network, ADNet [29], from the literature as our baseline network in our experiments and also use it as a backbone for our proposed trackers. Therefore, in this section, we first provide an overview of our chosen baseline network.

The action-decision network, ADNet, determines the future position of the object of interest in terms of a predicted action

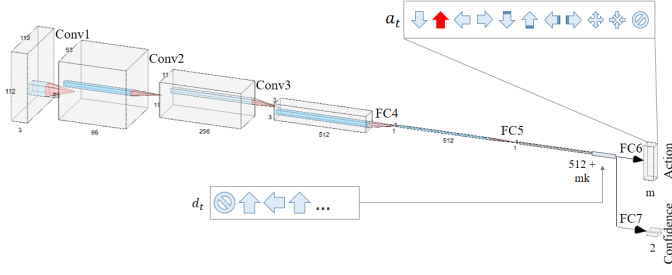


Fig. 1: Architecture of our base-network (ADNet). In this representation, “up” translation action (displayed in red) is predicted to capture the position of the object.

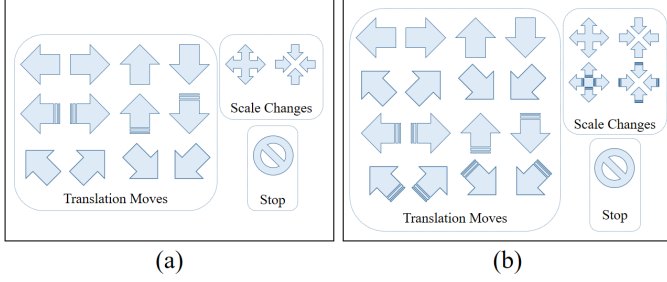


Fig. 2: The set of actions defined for (a) our base-network, and for (b) *Model-A*.

sequence, and each action in the sequence is predicted from the current state (where the state represents the current location of the bounding box). ADNet is trained by both supervised and reinforcement learning techniques, and an online adaptation process in tracking is adopted during the testing process, if the confidence score drops below a certain threshold. The network’s architecture is presented in Figure 1. We use the same architecture as base network in our approach. However, we also explain and show the changes that we made in the architecture in the following sections.

ADNet follows a Markov Decision Process strategy for tracking, which is defined by a tuple of states  $s \in S$ , actions  $a \in A$ , a state transition function  $s' = f(s, a)$ , and a reward function  $r(s, a)$ .

In a video that has  $L$  frames, an action and a state are represented as  $a_{t,l}$  and  $s_{t,l}$ , respectively, for  $t = 1, \dots, T_l$  and  $l = 1, \dots, L$ , where  $T_l$  denotes the terminal state at frame  $l$ . The terminal state occurs when the action is *stop*, and is transferred to the next frame as following:  $s_{t,l+1} = s_{T_l,l}$ .

The action set,  $A$ , contains a total number of 11 actions including translation movements, scale changes, and *stop* action. The complete action set can be seen in Figure 2(a).

The state,  $s_t$ , is defined as a tuple consisting of the image patch:  $p_t$ , and the action dynamics vector:  $(p_t, d_t)$ . The bounding box:  $b_t$  is represented by a vector,  $b_t = [x^{(t)}, y^{(t)}, w^{(t)}, h^{(t)}]$ . The first two elements of  $b_t$  correspond to the  $x$  and  $y$  coordinates of the center; whereas the last two elements denote the width and height of the bounding box, respectively. The patch:  $p_t$  in frame  $F$  at iteration  $t$  is defined as:  $p_t = \phi(b_t, F)$ , where  $\phi$  is a function to resize the input

image to fit the input size of the network. Action dynamics vector stores the history of the past  $k$  actions.

Each action  $a_t$  associated with state  $s_t$  determines the next state  $s_{t+1}$  by state transition functions. State transition functions are patch transition function  $b_{t+1} = f_p(b_t, a_t)$ , which moves the patch according to the action; and action dynamics function  $d_{t+1} = f_d(d_t, a_t)$ , which stores the action history transition. Correspondingly, the amount of change in the patch is calculated by  $\Delta x^{(t)} = \alpha w^{(t)}$  and  $\Delta y^{(t)} = \alpha h^{(t)}$ , where  $\alpha$  is a constant.

When the *stop* action (the terminal action for a sequence of actions) is reached as a result of state transitions, the patch is finalized. The tracker obtains the reward after the tracking is performed on all the frames in each video sample. Then, the reward function,  $r(s)$ , is activated for the action sets resulting from these tracking simulations. ADNet uses the following reward function, where  $IoU(b_T, G)$  is the overlap ratio of the ground truth and the terminal patch:

$$r(s_T) = \begin{cases} +1, & \text{if } IoU(b_T, G) > 0.7 \\ -1, & \text{otherwise} \end{cases} \quad (1)$$

The reward at the termination state,  $z_t = r(s_T)$  is interpreted as the tracking score (target value) while updating the network in reinforcement learning stage.

#### IV. OUR ACTION-SEQUENCE-BASED RL TRACKERS

In our action-sequence-based RL tracker, we use ADNet as the base network. In this paper, we introduce four different models and we name them: *Model-A*, *Model-B*, *Model-C* and *Model-D*. Below, we explain each of those models in details.

##### A. Model-A

Our first model focuses on utilizing and choosing proper action sets for the drone domain. For that task, we utilized and tested different setups. However, we report only the best working action set for us here. We noticed that utilizing the action set as shown in Figure 2(b) showed best for us. There, we have 12 actions representing the directional movements, two actions for scale changes and one action representing the terminal action (*stop*), yielding total of 15 actions.

##### B. Model-B

Our base network (ADNet) uses a pre-trained network to extract the visual features. In particular, the ADNet architecture uses the first three convolutional layers of VGG-M [4] for initialization of the filters, biases and weights. In order to analyze the effect of backbone network on model performance, our second model, *Model-B*, uses VGG-F [4] as backbone network. Since VGG-F has relatively larger filter sizes compared to VGG-M, we choose the first two convolutional layers of VGG-F so that the filter size does not exceed the input dimensions through convolutional layers. Figure 3 displays the detailed network structure of *Model-B*.

The first convolutional layer of VGG-F consists of 64 filters of size  $11 \times 11$ , with stride 4. It is followed by ReLu and batch normalization layers. Then, a max-pooling operation is applied

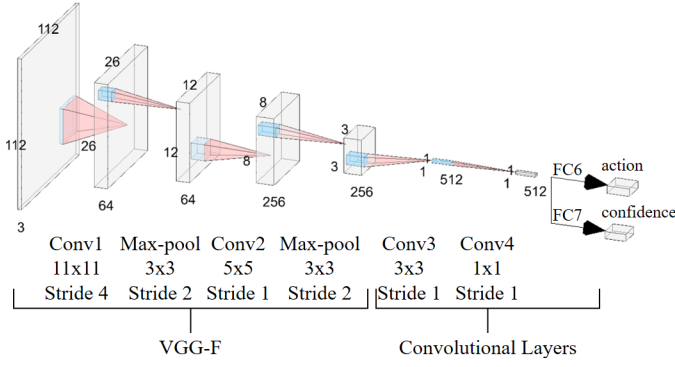


Fig. 3: Architecture of *Model-B*.

with filter size  $3 \times 3$  and stride 2. The second convolutional layer has 256 filters of size  $5 \times 5$  with stride 1. Similarly, this layer is also followed by ReLu activation, batch normalization, and an identical max-pooling layer.

### C. Model-C

In an attempt to increase the performance of reinforcement learning stage, we introduce *Model-C* by making alterations on two areas of the reinforcement learning algorithm of our base-network: reward function and RL algorithm. First of all, we introduce a hybrid reward function in the reinforcement learning stage, where the length of action set and the overlap ratio are both included during the rewarding process. It is important to recognize that *Model-C* generates rewards in the terminal patches, which indicates that the sequence of actions is rewarded, rather than individual actions. Thus, the new reward function lets the model utilize the amount of actions required to reach the target state. Moreover, the overlap ratio,  $IoU(b_T, G)$ , is included so that the reward is also proportional to the success of the tracking sequence. The reward function of *Model-C* is defined as following:

$$r(s_T) = \begin{cases} (10 - \text{length}(\{a_{t,l}\})) * IoU(b_T, G), & \text{if } IoU(b_T, G) > 0.7 \\ -1, & \text{otherwise} \end{cases} \quad (2)$$

where  $\{a_{t,l}\}$  is the set of actions resulting from tracking simulation.

Secondly, our base model updates the network parameters,  $W_{RL}$ , after the calculation of tracking scores on all the frames in each video clip sampled from test videos. In contrast, we successively calculate target values and rewards, then update the network parameters. Hence, we let reinforcement learning algorithm give reward to the set of actions each time the tracking is applied on a sample video clip to update the model (Algorithm 1). In this way, the rewards and punishments are given right after the tracking operation of each sample video clip is terminated in order to obtain robustness and increase the performance during reinforcement learning.

### D. Model-D

*Model-D* is very similar to *Model-C* in terms of reinforcement learning algorithm, that is to say *Model-D* also updates

**Data:** Pre-trained network ( $W_{SL}$ ), training sequences  $\{F_l\}$  and ground truths  $\{G_l\}$

**Result:** Trained network weights ( $W_{RL}$ )

Initialize  $W_{RL}$  with  $W_{SL}$ ;

**while**  $W_{RL}$  does not converge **do**

Randomly select  $\{F_l\}_{l=1}^L$  and  $\{G_l\}_{l=1}^L$

Set initial  $b_{1,1} \leftarrow G_1$

Set initial  $d_{1,1}$  as zero vector

$T_1 \leftarrow 1$

**for**  $l \leftarrow 2$  **to**  $L$  **do**

$\{a_{t,l}\}, \{b_{t,l}\}, \{d_{t,l}\}, T_l$

$\leftarrow \text{TRACKING}(b_{T_{l-1},l-1}, d_{T_{l-1},l-1}, F_l)$

Compute tracking scores  $\{z_{t,l}\}$  with  $\{b_{t,l}\}$  and  $\{G_l\}$

Calculate  $\Delta W_{RL}$

Update  $W_{RL}$  using  $\Delta W_{RL}$

**end**

**end**

**Algorithm 1:** Action-Sequence-Based Tracker (*Model-C*)

the parameters of the network using Algorithm 1. However, its reward function is identical to that of our base-network, ADNet, which is described in Eq. 1.

## V. TRAINING OUR ACTION-SEQUENCE-BASED RL TRACKER

Training of an action-sequence-based RL trackers consists of three individual training stages: (i) supervised learning, (ii) reinforcement learning and (iii) online adaptation in tracking. Supervised learning trains the network to learn and predict action labels with respect to the patch positions. In the reinforcement learning stage, the network that resulted from supervised learning is updated by training sequences with tracking simulation. The main goal of reinforcement learning is to utilize and improve action dynamics. Finally, while tracking the test sequences, the challenging cases such as changes in appearance and deformations of the objects are captured by applying an online adaptation stage.

### A. Supervised learning stage

The supervised learning stage aims to train the network parameters,  $\{w_1, w_2, \dots, w_7\}$ , to associate the states with the corresponding actions. Hence, the action dynamics vector is not taken into consideration in this stage.

The training data set consists of patches and size of ground truth for each frame in the videos. Therefore, sample patches to be used during supervised learning are generated by applying Gaussian noise to the ground truth bounding boxes. Action label,  $o_j^{act}$ , for a patch is predicted to be the action that results in maximum overlap ratio of ground truth and the patch associated with this action, i.e. Intersection-Over-Union ( $IoU$ ), as following:  $o_j^{act} = \arg \max_a IoU(\hat{f}(p_j, a), G)$ .

Moreover, it is assumed that the class label,  $o_j^{cls}$ , of a patch is predicted correctly if the overlap ratio of patch and ground truth is higher than 0.70, as shown in Eq. (3).

TABLE I: Comparison of our proposed methods to the baseline algorithm on OTB-100 and VisDrone2019 data sets.

Experiment Type	Model	OTB-100			VisDrone2019		
		Precision (20 pixels)	FPS	IoU	Precision (20 pixels)	FPS	IoU
Baseline model	<i>ADNet</i>	78.47%	4.89	0.603	89.15%	6.33	0.579
Action set	<i>Model-A</i>	79.45%	4.58	0.612	91.94%	6.08	0.557
Backbone network	<i>Model-B</i>	77.15%	8.11	0.574	89.67%	6.53	0.553
Reward function	<i>Model-C</i>	80.61%	6.25	0.589	93.02%	5.61	0.611
	<i>Model-D</i>	81.62%	7.02	0.616	91.74%	6.13	0.615

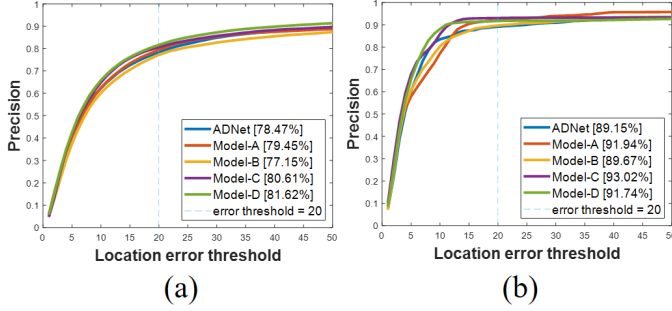


Fig. 4: Precision vs. location error threshold plots for experiments on (a) OTB-100 data set, (b) VisDrone2019 data set. The percentages in the legends correspond to the mean precision of the model performance on the corresponding data set when the location error threshold is 20 pixels.

$$o_j^{cls} = \begin{cases} 1, & \text{if } IoU(p_j, G) > 0.7 \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

The loss function  $L_{SL}$  is defined as a multi-task function to train parameters in our action-sequence-based RL tracker.  $L_{SL}$  combines the cross-entropy losses of both action label prediction,  $\hat{o}_j^{act}$ , and class label prediction,  $\hat{o}_j^{cls}$ .  $L_{SL}$  is defined as

$$L_{SL} = \frac{1}{m} \sum_{j=1}^m L(o_j^{act}, \hat{o}_j^{act}) + \frac{1}{m} \sum_{j=1}^m L(o_j^{cls}, \hat{o}_j^{cls}) \quad (4)$$

where  $m$  is the batch size, and  $L$  is the cross-entropy loss. The network parameters,  $W_{SL}$ , are updated by minimizing the loss function.

The steps of supervised learning stage is identical for all of our action-sequence-based RL trackers: *Model-A*, *Model-B*, *Model-C* and *Model-D*.

### B. Reinforcement learning stage

In reinforcement learning, the network parameters  $\{w_1, w_2, \dots, w_6\}$  are trained in order to capture the state-action policies. Thus, *FC7* layer, which predicts the confidence score is neglected.

Reinforcement learning initially uses the network parameters resulting from the supervised learning stage. Through the iterations of reinforcement learning stage, a training sequence is randomly selected with their ground truths. Then, tracking simulation is performed by training this sequence labelled by ground truths. During tracking simulation, state set  $\{s_{t,l}\}$ ,

action set  $\{a_{t,l}\}$ , and reward set  $\{r(s_{t,l})\}$  are produced for  $t = 1, \dots, T_l$ , and  $l = 1, \dots, L$ . The action  $a_{t,l}$  is defined as the action that maximizes the conditional action probability  $p(a_{t,l}|s_{t,l})$ , under network parameters  $W_{RL}$ ,  $\{w_1, w_2, \dots, w_6\}$ , as shown:  $a_{t,l} = \arg \max_a p(a|s_{t,l}; W_{RL})$ .

The tracking score of the simulation corresponds to the reward given at the terminal state. In *Model-A*, *Model-B* and *Model-D*, successful tracking with *IoU* greater than 0.70 is rewarded with +1, whereas failures are punished with -1, as defined in Eq. (1). On the other hand, *Model-C* uses a reward function that is a hybrid approach combining the *IoU* value and the number of actions in the action set resulting from tracking simulation, as in Eq. (2).

Furthermore, in reinforcement learning stage, the network parameters ( $W_{RL}$ ) are updated using Stochastic Gradient Ascend in order to maximize the tracking scores, as shown in Eq. (5).

$$\Delta W_{RL} \propto \sum_l^L \sum_t^{T_L} \frac{\partial \log p(a_{t,l}|s_{t,l}; W_{RL})}{\partial W_{RL}} z_{t,l} \quad (5)$$

With a different approach than *Model-A* and *Model-B*, which update the network parameters after applying simulation tracking on all the frames of each video clip, *Model-C* and *Model-D* successively calculates and updates network for each frame in a sampled video clip, as shown in Algorithm 1. Namely, the RL algorithms vary across our proposed models.

### C. Online adaptation in tracking

During the implementation of tracking, the pre-trained network is updated to be more robust to challenges such as sudden changes in the appearance of the object, or deformation. The convolutional layers of ADNet  $\{w_1, w_2, w_3\}$  possess the information regarding tracking. On the other hand, fully connected layers  $\{w_4, w_5, w_6, w_7\}$  are expected to learn the information that is specific to the videos. Therefore, during online tracking, the convolutional layers remain stable, whereas fully connected layers are fine-tuned.

Fine-tuning the parameters requires a supervised learning approach. The patch position predicted by the network is considered as the temporal ground truth. During online adaptation, the training samples contain patches that are randomly sampled around this ground  $\{p_i\}$ , action labels  $\{o_j^{act}\}$ , and class labels  $\{o_j^{cls}\}$ , with a very similar approach adopted in the supervised learning stage. Online adaptation is conducted in every  $I$  frames using these training samples.

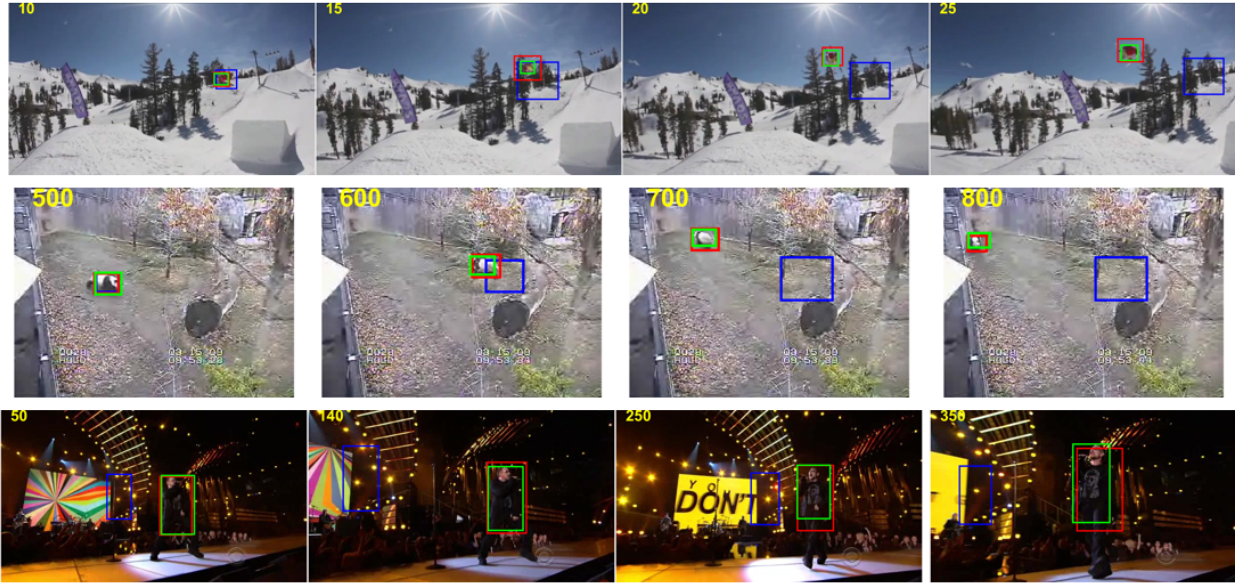


Fig. 5: Sample successful cases of our *Model-D* over the baseline network (ADNet) on OTB-100 data set. The tracking results of *Skiing*, *Panda* and *Singer2* from OTB-100 data set are displayed. The blue, red and green bounding boxes represent the bounding boxes of ADNet, *Model-D* and the ground truths, respectively.

The confidence score,  $c(s_{t,l})$ , of state  $s_{t,l}$ , is the target probability,  $p(\text{target}|s_{t,l}; W)$ , that result from the confidence layer, *FC7*. If the confidence score of the predicted target is greater than 0.50, training samples are generated using the tracked patch positions. The tracker is said to miss the target if the confidence score is less than  $-0.50$ . In that case, the target is re-detected by selecting the target position candidate with the highest confidence score, as  $b^* = \arg \max_{\tilde{b}_i} c(\tilde{b}_i)$ , where  $\tilde{b}_i$  is the set of target position candidates randomly generated by applying Gaussian noise to the current target position.

Our action-sequence-based RL trackers all adopt the same approach in online adaptation in tracking.

## VI. EXPERIMENTS

In this section, we evaluate the performance of our proposed models and compare them to ADNet on two data sets. We first train our models on VOT data sets and then test and compare their performances on OTB-100 [27] data set. Then, we train and test our proposed models on recently proposed VisDrone2019 [32] data set.

First, our models are trained on 58 videos collected from VOT2013 [16], VOT2014 [15] and VOT2015 [14] data sets. We test those trained models on OTB, which includes a total number of 100 videos from both OTB-50 [26] and OTB-100 [27] data sets. Then, we train our RL based deep trackers on VisDrone2019 (*VisDrone2019-SOT trainset\_part1*) [32] data set which includes 43 aerial videos for training, and 11 aerial videos (from *VisDrone2019-SOT valset*) for testing.

The performance of the models are evaluated based on two metrics: center location error (*CLE*) and *IoU*. *CLE* corresponds to the distance between the center of the bounding box resulting from tracking, and the center of the ground truth,

whereas *IoU* is the ratio of the intersection of the predicted bounding box and its ground truth over their union. For all the experiments, the precision values as a one-pass evaluation is analyzed with a center location error threshold of 20 pixels.

## VII. ANALYSIS

The performance analysis of the models is conducted adopting the following two perspectives: overall performance with respect to the defined metrics, and the model performance against challenging aspects in visual tracking data set, OTB.

### A. Overall performance

In this part, we analyze the results of our RL-based trackers introduced in Section IV. Table I presents the precision, *IoU* and frames-per-second (*FPS*) values of the corresponding models on OTB-100 and VisDrone2019 data sets.

The results show the precision averaged across all test samples with a center location error threshold of 20 pixels. Figure 4 displays the precision against location error threshold for each of our trackers.

The additional directions in action set in *Model-A* seem to improve performance slightly for OTB-100 and VisDrone2019. Furthermore, *Model-B* can achieve faster performance in terms of *FPS* value when VGG-F is used as backbone network. *Model-B* appears to perform faster than ADNet while maintaining the precision, even though there is a negligible decrease in its value. Therefore, *Model-B* seems to be practical in cases where speed is an important parameter.

Ultimately, when we consider the precision metric, our proposed RL method introduced in *Model-C* and *Model-D* appears to improve the performance on both data sets. In particular, *Model-D* increases the precision by 3.15% on OTB-100, and *Model-C* increases the precision on VisDrone2019

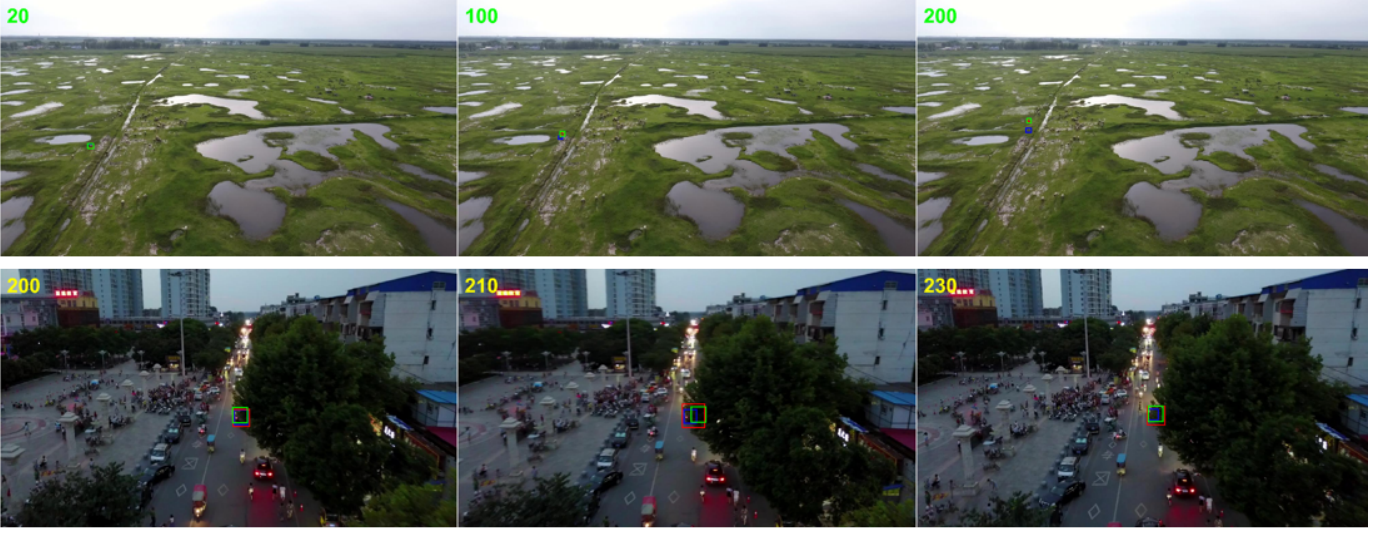


Fig. 6: Sample successful cases of our *Model-C* over the baseline network (ADNet) on VisDrone2019 data set. The tracking results of *uav0000092\_00575\_s* and *uav0000115\_00606\_s* from VisDrone2019 data set are displayed. The blue, red and green bounding boxes represent the bounding boxes of ADNet, *Model-C* and the ground truths, respectively.

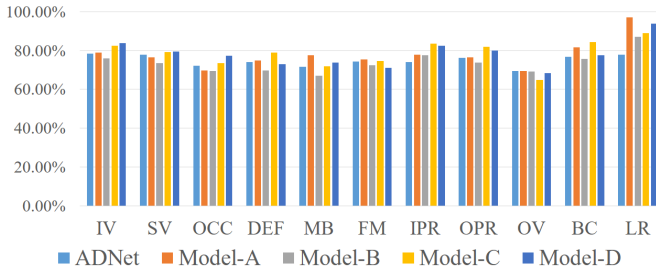


Fig. 7: Average precision results of *ADNet*, *Model-A*, *Model-B*, *Model-C*, and *Model-D* across the set of videos from OTB-100, grouped by challenging aspects.

data set by 3.87%. It can be interpreted that experimenting with the definition of reward function proves a considerable improvement on the overall model performance.

### B. Challenging aspects

Next, we take a closer look at how models handle some of the challenges in tracking task. The test sequences in OTB-100 are manually labeled with 11 different attributes, which reflects the challenging aspects including but not limited to scale variation, deformation, background clutter, and blur [27]. The number of videos bearing the challenging aspects and their definitions are presented in Table II.

Next, we discuss the model performance under various groups of challenges for tracking task on OTB-100. Figure 7 shows the average precision values of each model against challenging aspects.

In Figure 7, we observe that *Model-A*, which has a larger action set, seems to perform best for *Low Resolution* videos. The model exhibits the best performance with a significantly high precision under *Low Resolution* videos. On the other

TABLE II: Challenging aspects in OTB-100.

Name	Description	Number of videos
IV	Illumination Variation	38
SV	Scale Variation	64
OCC	Occlusion	49
DEF	Deformation	44
MB	Motion Blur	29
FM	Fast Motion	39
IPR	In-Plane Rotation	51
OPR	Out-of-Plane Rotation	63
OV	Out-of-View	14
BC	Background Clutters	31
LR	Low Resolution	9

hand, it has the lowest performance on *Occlusion* set. Also, with their proposed reward functions, *Model-C* and *Model-D* seems to significantly improve precision under *Illumination Variation*, *In-Plane Rotation* and *Low Resolution*.

Sample successful cases of *Model-D* on OTB-100, and *Model-C* on VisDrone2019 can be seen in Figure 5 and 6.

## VIII. CONCLUSION

Recently, drone-based applications receive an increasing demand in a wide range of domains such as video surveillance and entertainment. As camera equipped drones are widely used in such domains, more and more computer vision based solutions are needed & utilized on aerial videos recorded by drones. Although one of the most common vision tasks is object tracking, when applied on aerial images, it still suffers due to additional challenges such as smaller target size, significant change in orientation & scale and the movements of both UAV's camera and targets.

In this paper, we introduce a set of *action-sequence-based deep reinforcement learning trackers* for visual object tracking in aerial videos and demonstrate how RL based tracker can

be adopted on drone (UAV) images. We tested our trackers on both ground-taken videos (OTB-100), and drone videos (VisDrone2019). In our trackers, we discuss the action types, backbone network structure, and reward function in RL stage. We compare our proposed solutions to a recent model and demonstrate improvements in both precision and *IoU* values. Our *Model-D* demonstrates an improvement on precision up to 3.15% on OTB-100 dataset, while *Model-C* improves precision by 3.87% on VisDrone2019 dataset.

#### ACKNOWLEDGMENT

This paper has been produced benefiting from the 2232 International Fellowship for Outstanding Researchers Program of TÜBİTAK (Project No:118C356). However, the entire responsibility of the paper belongs to the owner of the paper. The financial support received from TÜBİTAK does not mean that the content of the publication is approved in a scientific sense by TÜBİTAK.

#### REFERENCES

- [1] Sebastiano Battiato, Luciano Cantelli, Fabio D'Urso, Giovanni Maria Farinella, Luca Guarnera, Dario Guastella, Carmelo Donato Melita, Giovanni Muscato, Alessandro Ortis, Francesco Ragusa, et al. A system for autonomous landing of a uav on a moving vehicle. In *International Conference on Image Analysis and Processing*, pages 129–139. Springer, 2017.
- [2] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.
- [3] David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 2544–2550. IEEE, 2010.
- [4] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.
- [5] Jongwon Choi, Hyung Jin Chang, Sangdoo Yun, Tobias Fischer, Yiannis Demiris, and Jin Young Choi. Attentional correlation filter network for adaptive visual tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4807–4816, 2017.
- [6] Andre Cohen and Vladimir Pavlovic. Reinforcement learning for robust and efficient real-world tracking. In *2010 20th International Conference on Pattern Recognition*, pages 2989–2992. IEEE, 2010.
- [7] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Eco: Efficient convolution operators for tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6638–6646, 2017.
- [8] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *European conference on computer vision*, pages 472–488. Springer, 2016.
- [9] Martin Danelljan, Fahad Shahbaz Khan, Michael Felsberg, and Joost Van de Weijer. Adaptive color attributes for real-time visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1090–1097, 2014.
- [10] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. A twofold siamese network for real-time object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4834–4843, 2018.
- [11] Ajit Jadhav, Prerana Mukherjee, Vinay Kaushik, and Brejesh Lall. Aerial multi-object tracking by detection using deep association networks. In *2020 National Conference on Communications (NCC)*, pages 1–6. IEEE, 2020.
- [12] Xu Jia, Huchuan Lu, and Ming-Hsuan Yang. Visual tracking via adaptive structural local sparse appearance model. In *2012 IEEE Conference on computer vision and pattern recognition*, pages 1822–1829. IEEE, 2012.
- [13] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, 34(7):1409–1422, 2011.
- [14] Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Luka Cehovin, Gustavo Fernandez, Tomas Vojir, Gustav Hager, Georg Nebehay, and Roman Pflugfelder. The visual object tracking vot2015 challenge results. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 1–23, 2015.
- [15] Matej Kristan, Jiri Matas, Aleš Leonardis, Tomáš Vojř, Roman Pflugfelder, Gustavo Fernandez, Georg Nebehay, Fatih Porikli, and Luka Čehovin. A novel performance evaluation methodology for single-target trackers. *IEEE transactions on pattern analysis and machine intelligence*, 38(11):2137–2155, 2016.
- [16] Matej Kristan, Roman Pflugfelder, Ales Leonardis, Jiri Matas, Fatih Porikli, Luka Cehovin, Georg Nebehay, Gustavo Fernandez, Tomas Vojir, et al. The vot2013 challenge: overview and additional results. 2014.
- [17] Tian Li, Feifei Ding, and Wenyan Yang. Uav object tracking by background cues and aberrances response suppression mechanism. *Neural Computing and Applications*, pages 1–15, 2020.
- [18] Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. Hierarchical convolutional features for visual tracking. In *Proceedings of the IEEE international conference on computer vision*, pages 3074–3082, 2015.
- [19] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4293–4302, 2016.
- [20] Paraskevi Nousi, Ioannis Mademlis, Iason Karakostas, Anastasios Tefas, and Ioannis Pitas. Embedded uav real-time visual object detection and tracking. In *2019 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 708–713. IEEE, 2019.
- [21] Siyang Pan, Zhihang Tong, Yanyun Zhao, Zhicheng Zhao, Fei Su, and Bojin Zhuang. Multi-object tracking hierarchically in visual data taken from drones. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.
- [22] Yibing Song, Chao Ma, Xiaohe Wu, Lijun Gong, Linchao Bao, Wangmeng Zuo, Chunhua Shen, Rynson WH Lau, and Ming-Hsuan Yang. Vital: Visual tracking via adversarial learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8990–8999, 2018.
- [23] James Supancic III and Deva Ramanan. Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 322–331, 2017.
- [24] Ran Tao, Efstratios Gavves, and Arnold WM Smeulders. Siamese instance search for tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1420–1429, 2016.
- [25] Michael Teutsch and Wolfgang Krüger. Detection, segmentation, and tracking of moving objects in uav videos. In *2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance*, pages 313–318. IEEE, 2012.
- [26] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2411–2418, 2013.
- [27] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015.
- [28] Pei Yang and Jiyue Huang. Trackdqn: Visual tracking via deep reinforcement learning. In *2019 IEEE 1st International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*, pages 277–282. IEEE, 2019.
- [29] Sangdoo Yun, Jongwon Choi, Youngjoon Yoo, Kimin Yun, and Jin Young Choi. Action-decision networks for visual tracking with deep reinforcement learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2711–2720, 2017.
- [30] Da Zhang, Hamid Maei, Xin Wang, and Yuan-Fang Wang. Deep reinforcement learning for visual object tracking in videos. *arXiv preprint arXiv:1701.08936*, 2017.
- [31] Zhao Zhong, Zichen Yang, Weitao Feng, Wei Wu, Yangyang Hu, and Cheng-Lin Liu. Decision controller for object tracking with deep reinforcement learning. *IEEE Access*, 7:28069–28079, 2019.
- [32] Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Qinghua Hu, and Haibin Ling. Vision meets drones: Past, present and future. *arXiv preprint arXiv:2001.06303*, 2020.