drones

MDPI

Article

Drones Chasing Drones: Reinforcement Learning and Deep Search Area Proposal

Moulay A. Akhloufi [1,*,†], Sebastien Arola [1,2,†] and Alexandre Bonnet [1,2,†]

[1] Perception, Robotics, and Intelligent Machines (PRIME), Department of Computer Science, Université de Moncton, Moncton, NB E1A3E9, Canada
[2] UPSSITECH, 31062 Toulouse, France
* Correspondence: moulay.akhloufi@umoncton.ca; Tel.: +1-506-858-4120
† These authors contributed equally to this work.

Received: 30 June 2019; Accepted: 13 July 2019; Published: 16 July 2019

Abstract: Unmanned aerial vehicles (UAVs) are very popular and increasingly used in different applications. Today, the use of multiple UAVs and UAV swarms are attracting more interest from the research community, leading to the exploration of topics such as UAV cooperation, multi-drone autonomous navigation, etc. In this work, we propose two approaches for UAV pursuit-evasion. The first approach uses deep reinforcement learning to predict the actions to apply to the follower UAV to keep track of the target UAV. The second approach uses a deep object detector and a search area proposal (SAP) to predict the position of the target UAV in the next frame for tracking purposes. The two approaches are promising and lead to a higher tracking accuracy with an intersection over union (IoU) above the selected threshold. We also show that the deep SAP-based approach improves the detection of distant objects that cover small areas in the image. The efficiency of the proposed algorithms is demonstrated in outdoor tracking scenarios using real UAVs.

Keywords: unmanned aerial vehicles; tracking; deep learning; reinforcement learning; drone pursuit-evasion

1. Introduction

In recent years, UAVs (unmanned aerial vehicles) have seen an increase in popularity and are now widely used in many applications. An area that was once mostly limited to the military is now expanding into commercial and industrial sectors, with hundreds of drone applications emerging every day. Today, UAVs have become central to the functions of various industries and organizations. Many of these applications use individual UAVs. However, the use of multiple UAVs and UAV swarms is attracting more interest from the research community, leading to the exploration of topics such as UAV cooperation, multi-drone autonomous navigation, etc. In this work, we are interested in multiple UAV pursuit-evasion. The strategy can be collaborative or competitive between multiple UAVs. The goal here is to use deep learning and the captured images from one of the UAVs to detect and keep track/follow the second moving UAV.

Recent work has been proposed in the area of collaborative UAVs. Schmuck et al. [1] propose the use of multiple small UAVs to act as agents and collaborate in a monocular simultaneous localization and mapping (SLAM) application. Vemprala et al. [2] propose a vision-based collaborative localization framework for multiple micro aerial vehicles (MAVs). They use the images captured by the cameras of the MAVs to estimate the pose. Individual and relative pose estimations are combined to localize against surrounding environments. The algorithm uses feature detection and tracking to estimate a six-degrees-of-freedom pose. They tested their algorithms using simulations within Microsoft AirSim [3]. Tian et al. [4] proposed the use of UAV fleets to search for lost persons. The UAVs collaboratively explore an area under thick forest canopies where the use of GPS is unreliable.

They propose the use of onboard computations and wireless communication. A laser-range finder equips each UAV and is used to estimate its position, location, and plan its path. A 3D map of the terrain is built, and all the individual maps are fused on a ground station. With the built maps, it is possible to recognize unexplored and already-searched spots. Chung et al. [5] review aerial swarm robotics approaches and techniques that allow each member of the swarm to communicate and allocate tasks among themselves, plan their trajectories, and coordinate their flight to efficiently achieve the overall objectives of the swarm. The paper reviews state-of-the-art theoretical tools, focusing on those developed and applied to aerial swarms. They present results covering trajectory generation, task allocation, adversarial control, distributed sensing, monitoring, and mapping. Other work deals more with pursuit-evasion involving unmanned aerial vehicles. Alexopoulos et al. [6] propose a simulation of a two-player pursuit-evasion game with UAVs in a three-dimensional environment. They present a game theoretic framework to solve the pursuit problem using two identical quad-rotors. Krishnamoorthy et al. [7] use a UAV to search a target moving on a road. Ground sensors are used to trigger the search by a UAV. In [8], Alexopoulos et al. propose a cooperative pursuit-evasion game with UAVs. They present a game-theoretical solution and show its applicability using multiple real outdoor flight experiments with a hex-rotor UAV. Fargeas et al. [9] analyze the problem of path planning for a team of UAVs performing surveillance and pursuit. The UAVs rely on communicating with ground sensors to detect potential intruders. They propose a heuristic algorithm to coordinate the UAVs during surveillance and pursuit and show its effectiveness through simulations. Fu et al. [10] propose a ground target pursuit algorithm using fixed-wing UAVs. They propose a method that generates waypoints and direct the UAV to the latest waypoint. They simulated three different scenarios to demonstrate the performance and efficiency of the proposed algorithm. In Camci et al. [11], the authors are interested in the control of quadcopters in pursuit-evasion scenarios. They propose the use of reinforcement learning to tune type-2 Takagi–Sugeno–Kang fuzzy logic controllers (TSK-FLCs). They benchmarked their approach in several scenarios and under different noisy conditions.

In this work, we propose the use of vision-based deep learning object detection and reinforcement learning for detecting and tracking a UAV (target or leader) by another UAV (tracker or follower). The proposed framework uses vision data captured by a UAV and deep learning to detect and follow another UAV. The algorithm is divided into two parts, the detection of the target UAV and the control of UAV navigation (follower UAV). The deep reinforcement learning approach [12] uses a deep convolutional neural network (CNN) to extract the target pose based on the previous pose and the current frame. The network works like a Q-learning algorithm [13]. The output is a probabilistic distribution between a set of possible actions such as translation, resizing (e.g., when a target is moving away), or stopping. For each frame from the captured sequence, the algorithm iterates over its predictions until the network predicts a "stop" when the target is within the desired position. The second approach uses deep learning object detection for the detection and tracking of a UAV in a pursuit-evasion scenario. The deep object detection approach uses images captured by a UAV to detect and follow another UAV [14,15]. This approach uses historical detection data from a set of image sequences and inputs this data to a search area proposal (SAP) algorithm in order to locate the area with a high probability UAV presence. The following sections present the details of the proposed approaches.

2. Proposed Framework

The proposed framework uses images captured by a UAV and a deep learning network to detect and follow another UAV in a pursuit-evasion scenario. The position of the detected target UAV (detected bounding box) is sent to a high-level controller that decides on the controls to send to the follower UAV to keep the target close to the center of its image frame. The proposed framework is presented in Figure 1.

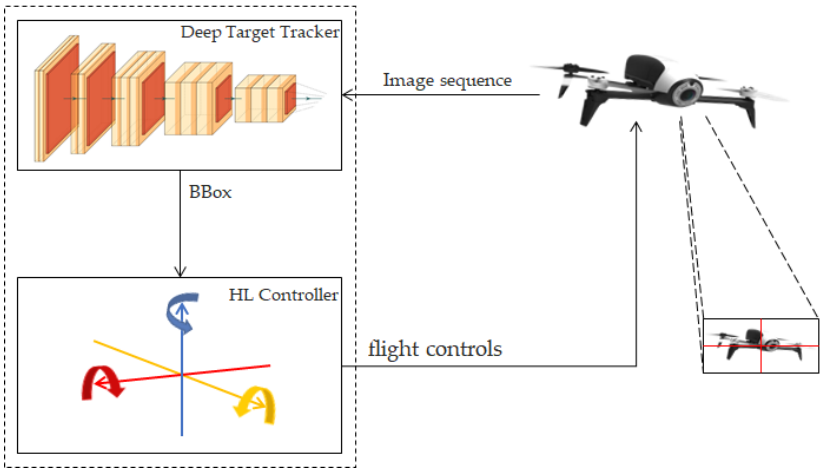

Figure 1. Proposed deep unmanned aerial vehicle (UAV) tracking framework.

2.1. Reinforcement Learning for Tracking

In this work, we want to develop an architecture capable of tracking moving targets using predictions over time from a sequence of previously captured frames. The proposed algorithm tracks the target and moves a bounding box according to each movement prediction. The architecture is based on an ADNet (action-decision network) [16]. The core network used in this architecture is based on the first layers of VGG-M [17] and is comprised of three convolutional layers (Conv) and two fully connected layers (FCs). The three convolutional layers use pretrained VGG-M. Each layer has a local response normalization and max-pooling. Fully connected layers use dropout for regularization. ReLU is used as activation. The fully connected layer FC5 is concatenated with actions history. Softmax is the last layer (action layer) used to get the probability for each predicted action. The number of layers is kept small to enable faster predictions and the associated controls. Figure 2 shows an overview of the proposed architecture.

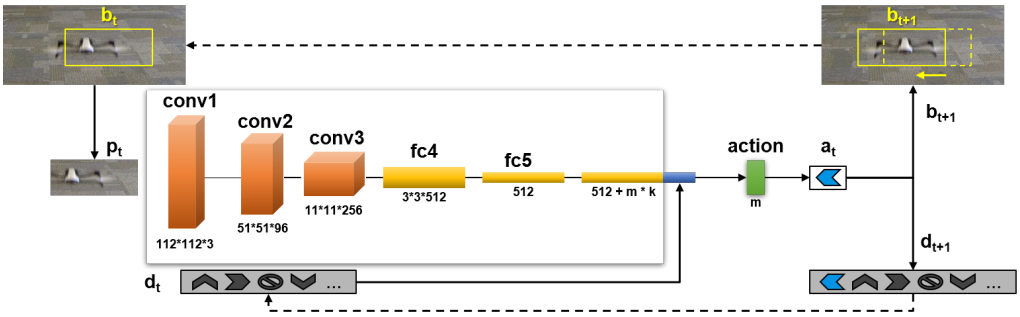

Figure 2. Deep reinforcement learning architecture overview. Enclosed in the box is the core convolutional neural network (CNN) based on VGG-M. Dashed arrows depict state transitions.

The proposed algorithm uses reinforcement learning [18,19], where we define actions, states, and rewards. Actions are the possible target's movements. We define three types of actions: translation, resizing, and stopping. Each action is encoded as a one-hot vector. States are pairs formed of a target's bounding box and actions history. Rewards are based on the computation of the intersection area between the bounding box and the ground truth. The reward is set to zero during iterations. When the predicted action is "stop" or when we fall in an oscillating pattern such as "left, right, left", the reward is updated and computed by Equation (1) as follows:

$$r(s_t) = \begin{cases} 1 & if\ IoU(b_t, G) > 0.7, \\ -1 & otherwise, \end{cases} \tag{1}$$

where s_t is the current state, b_t is the current detected bounding box, and G is the ground truth. *IoU* is the intersection over union, also known as Jaccard Index [20], a performance metric used in the evaluation of object detection algorithms. A threshold is used to decide if a detection is positive or negative. In our case, we set this threshold to 0.7, meaning that only a detection with an overlap above 70% is considered as a good detection (positive).

2.1.1. Supervised Learning

We need a dataset of UAV images for our supervised learning. We captured multiple video sequences and labeled the position of our target (ground-truth bounding box). To train our network to predict the right actions and to move a bounding box closer to the corresponding ground-truth bounding box, we added Gaussian noise to this data to get a set of pairs (noisy bounding box, action). Figure 3 shows an example from our dataset with the ground-truth bounding box and noisy boxes generated using Gaussian noise.

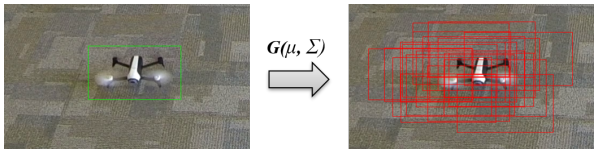

Figure 3. Ground-truth- (**left**) and Gaussian-noise-generated boxes (**right**).

Training the network consists of estimating the best action(s) to predict using the noisy boxes. We optimized an objective function that aims to maximize the IoU between the detected (noisy) bounding box and the corresponding ground-truth bounding box. This optimization problem is given by Equation (2):

$$action_i = \underset{a}{\operatorname{argmax}}\, IoU(f_b(b_i, a), G), \tag{2}$$

where $action_i$ is the predicted action, and $f_b(b_i, a)$ are the set of action(s) applied to the bounding box to position it closer to the ground-truth bounding box G.

2.1.2. Reinforcement Learning

Training for tracking was conducted using reinforcement learning. Reinforcement learning [18,21] learns using an agent experience and the associated rewards. Given an environment with a state space and an action space, and giving possible actions in the different states, each action a_t at a specific state s_t gives a reward r_t based on a defined reward function [18,21]. In reinforcement learning, the most important part is the formulation of an optimal policy that maximizes reward. It allows taking the actions history into account. In this work, we use the REINFORCE algorithm [21], a reinforcement learning method that directly updates the policy weights based on the policy gradient to optimize the obtained rewards. This policy rule is given by Equation (3):

$$\Delta\theta_t = \alpha \bigtriangledown_\theta \log(\pi_\theta(s_t, a_t))r_t, \tag{3}$$

where θ is the weight, α is the learning rate, and $\pi_\theta(s_t, a_t)$ is the policy that maps actions to probabilities. Our policy is based on the use of a deep convolutional neural network (CNN). The REINFORCE algorithm details are given in [21].

2.2. Deep Object Detection and Tracking

To detect the target UAV, we developed an approach based on the deep YOLO algorithm [22,23]. The detection is performed with YOLO v2 [23], a deep convolutional neural network for object detection. To follow the drone, we developed a high-level control algorithm based on the use of the detected bounding box coordinates. The bounding box size and position help estimate the position and

orientation of the leader UAV and compute the command to send to the follower UAV. This approach is able to process videos at a rate of 30 frames per second (fps) while keeping a mean average precision (mAP) of object detection above 50%.

We use a UAV equipped with a camera to detect the target to follow. YOLO v2 [23] is used to detect the object of interest. It is a popular 24-layer deep convolutional neural network built for object detection. YOLO v2 starts with images of a size of 224 × 224 pixels for the classifier training and then tunes this classifier with 448 × 448 images. Anchor boxes are used, and the object detector outputs a tensor with our class confidence. In this case, we train it to recognize UAVs and non-UAVs. This deep CNN shows very good results for our detection and recognition tasks. It is capable of a detection an mAP score greater than 50% on a 30 fps video. Its ability to process video streams in real time is very interesting for our objective since we need a very fast detection to be able to estimate the target UAV position, compute the controls, and send them to the follower UAV.

Prediction and Tracking Using a Search Area Proposal (SAP)

In this part of the work, we introduce the use of a search area proposal (SAP) based on particle filters. Particle filters [24,25] are a set of algorithms used to solve a probabilistic filtering problem. The goal is to estimate the state of a dynamic system with only partial information and a noise applied to the sensor that captures this information. A set of particles is generated with a weight for each of them. The best particles are the ones that are the closest to the next state of the system. This way, we can easily predict the next state of the system given previous states. These algorithms are very popular in object tracking applications when motion can undergo nonlinear trajectories. The term particle filter comes from statistics. The technique is also known as Monte Carlo localization (MCL) in robotics [26] and as a condensation (conditional density propagation) algorithm in computer vision [27]. A theoretical overview of particle filters can be found in [28].

The objective here is to use a search area proposal with a deep CNN detection to improve the detection performance and limit the candidates to specific areas of the image. The particle filters algorithm predicts the next positions of the target UAV. The output of YOLO (bounding box coordinates) is used as input to a particle filters algorithm, and the next target positions are predicted. With these predictions, we can reduce the search area (the region of interest around the predicted position). Figure 4 shows the proposed approach. With this deep SAP approach, the detection of the target when it is located far from the follower (more than 5 m) was improved. Without SAP, the deep CNN struggles to detect the target when it is located far away because the UAV size in the image is very small.

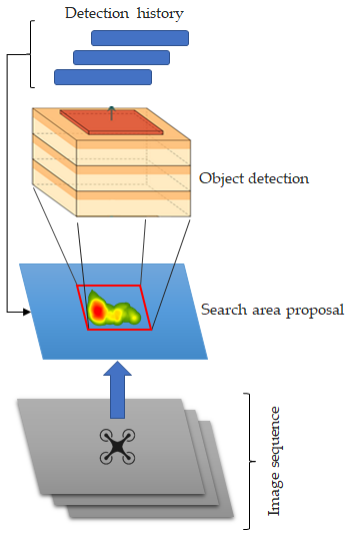

Figure 4. Deep object detector with a search area proposal. Image sequence and detection history are used to predict the area of interest (search area).

The extraction of the search area using SAP and its scaling to the input size of the deep CNN improves the detection accuracy. In Figure 5, we can see the predicted area (small image on the left) where particle filters estimated the UAV position. The green bounding box is the result of the detection in this area. The white bounding box is the box corresponding to the green box projected onto the whole frame.

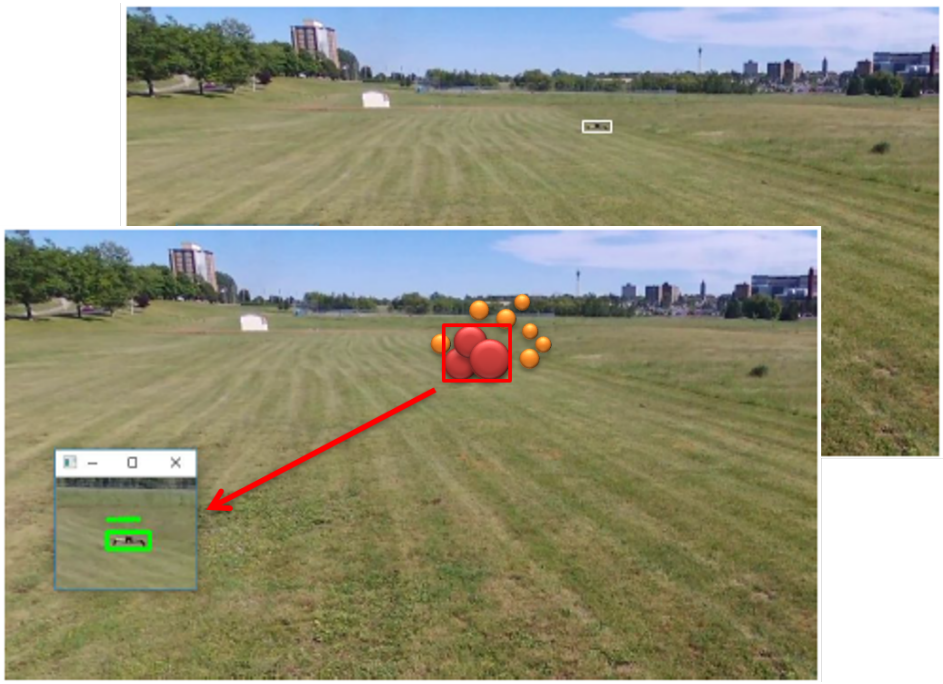

Figure 5. Example of a prediction given by the search area proposal. The particles with higher weights (in red) are used to detect the area of interest.

2.3. UAV Control

When the follower UAV detects the target UAV, it needs to control its motion to follow the target. We use the bounding box returned by the deep neural network to estimate the relative position of the target UAV and compute the necessary controls. We build a visual servoing based on the obtained data. The controls sent to the UAV are meant to keep the target close to the the center of the image. With this high-level control, we can follow the UAV when it flies in the following directions: right, left, up, down, and forward. The UAV built-in controller takes care of the remaining adjustments to keep the trajectory and motion smooth. For the reinforcement learning algorithm, these controls are predicted from the set of generated actions. Figure 6 shows the center of the image (video frame), the target position, the vectors used to estimate the controls, and the high-level controls sent to the UAV's onboard controller.

Du and Dv are the values we need to minimize to reduce the position error of Equation (4):

$$E\left(t\right) = s\left(t\right) - s^{*}\left(t\right), \tag{4}$$

with $s\left(t\right)$ being the target position at instant t and $s^{*}\left(t\right)$ the desired position at instant t (in our case, the image center).

$s\left(t\right)$ and $s^{*}\left(t\right)$ are defined by Equations (5) and (6):

$$s\left(t\right) = \left(u_{center} + Du\ v_{center} + Dv\right)^{t}, \tag{5}$$

and

$$s^{*}\left(t\right) = \left(u_{center}\ v_{center}\right)^{t}. \tag{6}$$

Thus, we need to select the best controls to apply from a set of possible controls S_c to minimize the error $E(t)$ (Equation (7)):

$$\underset{S_c}{\operatorname{argmin}} E(t). \tag{7}$$

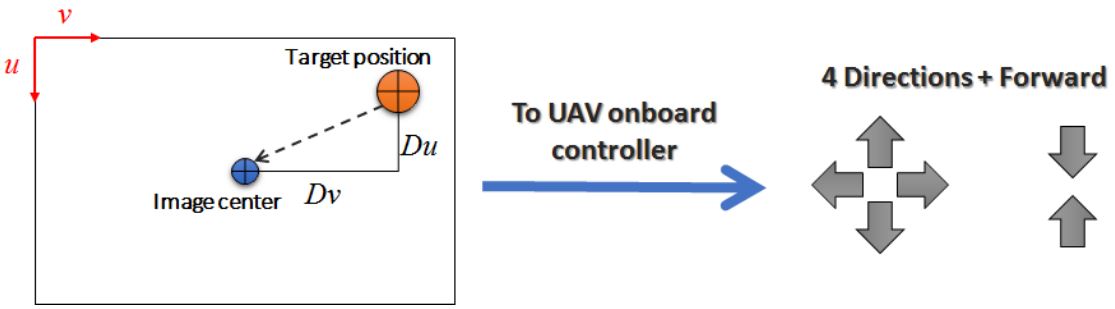

Figure 6. Target representation in the image frame for UAV control and the generated high-level controls sent to the UAV onboard controller.

While the estimation of the four directions of UAV control is straight forward, estimating forward motion when no 3D data is available can be challenging. To solve this problem, we developed a simple estimation based on the size of the UAV. The YOLO v2 deep convolutional neural network is good enough to estimate the closest bounding box to the target. The obtained coordinates give a good estimate of the UAV size. This size is compared with a reference size, and when a scale is below a defined threshold, a move forward control is generated. This way the follower UAV moves forward to be close to the target UAV until the estimated target UAV size is within a defined tolerance. This approach requires an initialization step before the flight. In our experiments, we used the size of a UAV positioned at 30 cm as a reference.

3. Datasets

To train our networks, we needed a specific dataset that could be trained for tracking a UAV by another UAV. We collected our own data to achieve this task. Two UAV models, Bebop 2 [29] and Mambo [30], were used.

3.1. Reinforcement Learning Dataset

For training the ADNet and reinforcement learning work, eight video sequences with two UAV types (Bebop 2 and Mambo) were captured. Our dataset was composed of 25,629 frames. For supervised learning, we added Gaussian noise, as described in Section 2.1.1. This substantially increased the number of images in our training dataset. Indoor and outdoor images were captured in this dataset, and the images were labeled using vision-based background removal, thresholding, color processing algorithms, and manual annotations in order to extract the shape of the UAV without background interference.

3.2. Deep Object Detection and SAP Training Dataset

To train the YOLO convolutional neural network to detect UAVs, we needed to develop a specific dataset of UAV images, especially with horizontal UAV views (back view, side view, and front view). We used two UAV models (Bebop 2 and Mambo) to capture multiple image sequences in indoor and outdoor conditions. Each frame was annotated with a bounding box that served as a reference for training. Various information are available in an XML format, such as bounding box coordinates, associated file name, etc.

We created a first dataset of 6000 images of different sizes and tested the performance of object detection using YOLO v2. The majority of the first images were from indoors. To refine the detection,

we collected additional images, mostly outdoors. A total of 2500 new images were added to the previous dataset. The complete dataset contained 8500 images of two different UAVs taken at different angles and in different conditions (e.g., changes in outdoor illumination conditions: sunny, cloudy, etc.).

4. Experimental Results

To test the proposed approaches, we developed a prototype in Python using Keras [31] and Tensorflow [32]. Algorithms were developed for deep learning and high-level UAV control. During the experimental tests, the escaping target UAV was piloted by a human pilot, and the tracker UAV was using the developed algorithms to lock onto the target and track it during the pursuit.

4.1. Reinforcement Learning

For training, we selected the first layers of VGG-M [17] as the core network for our architecture. A pretrained VGG network was used to initialize the convolutional layers, and fine-tuning was conducted in the remaining layers (Section 2.1).

Supervised learning was carried out in batches. The number of epochs was set to 100; for each epoch, we have 100 batches with a batch size of 128 frames. As described in Section 2.1.1, Gaussian noise was added to generate more samples for training with ground-truth labeled images. A total of 250 new noisy samples per frame were generated. With this processing, we obtained a total of 3,200,000 samples that could be used to train our network. The learning rate was set to 1×10^{-2}, with a decay of 1×10^{-4} per epoch.

Figure 7 shows the prediction accuracy based on the IoU measure during training. We can see that the accuracy quickly achieves high performances and keeps its value above 80% during training.

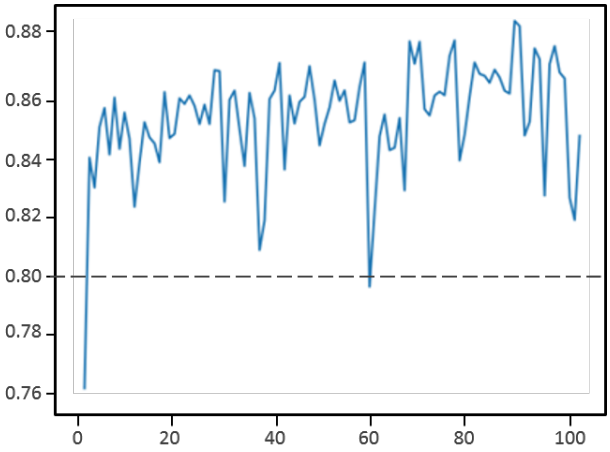

Figure 7. Accuracy learning curve during supervised learning.

Following the supervised learning phase, we trained our model using reinforcement learning to predict the actions for UAV tracking. The learning rate was set to 1×10^{-4}. This training works by episodes. An episode has a varying size, depending on the number of iterations the algorithm needs to find the target. Predicted actions can be translations or deformations (e.g., move forward) and stop when the target is correctly detected. These actions are shown in Figure 8. The proposed set of actions differs from the original ADNet actions and takes into account the change in aspect ratio due to UAV maneuvers.

Figure 8. Example of actions used for reinforcement learning. The first block represent translations, the second block deformations (scaling), and the last block is the stop sign representing the end of the predicted actions.

Figure 9 shows an example where the initial bounding box was positioned at the right side of the UAV. A set of actions was predicted in order to position the bounding box around the UAV; we can see a total of 5 actions. A first action quickly moves the bounding box to the left, followed by a movement down, then left, two moves forward (downscale), and finally, stop (meaning that the bounding box encloses the UAV).

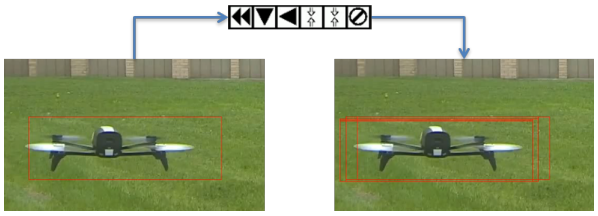

Figure 9. Set of predicted actions, from original position (**left**) and all the positions until stopping (**right**).

The rewards obtained for each episode during reinforcement learning are shown in Figure 10. We can see that the algorithm quickly learns to predict the right actions necessary for tracking the target UAV and keeps the rewards above 0.8 during the episodes.

During our experimental tests, the algorithm showed a good performance in tracking the object of interest. Figure 11 shows an example of UAV target detection and the predicted actions during an outdoor experimental test. However, some limitations remain. Since the processing is time consuming, it was difficult to chase a UAV at a higher speed. This is because the network predicts an action based on the last position of the target. If the target moves too fast, or if the network makes a wrong prediction, we lose the target and have to reinitialize the detection to start again. This limitation can be improved by optimizing the processing and deployment of the proposed algorithms.

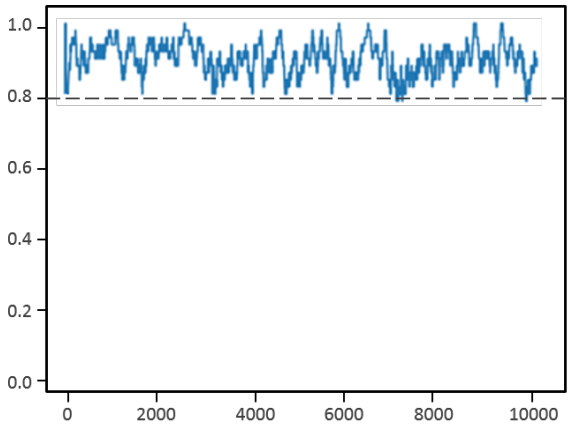

Figure 10. Rewards obtained during reinforcement learning.

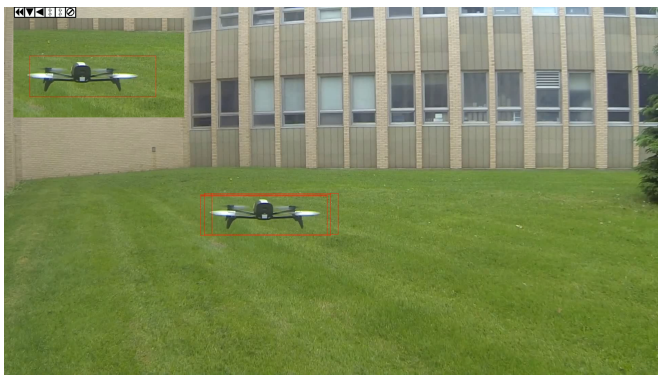

Figure 11. Example of detection and predicted actions during tracking.

4.2. Deep Object Detection and Tracking

4.2.1. Object Detection Results

We trained YOLO v2 on the collected and labeled data (Section 3.2). After 150 epochs of training on 90% of the images, we achieved an mAP score of 0.96. Training was conducted using a batch size of 14 and a learning rate of 1×10^{-4}. This score shows the performance of the proposed network for specifically detecting UAVs. Since we only deal with detecting UAVs, the number of classes is reduced, which explains the higher detection performance. The loss curve is given in Figure 12. We can see that the loss decreases quickly, and we obtain a lower loss value after only 80 epochs.

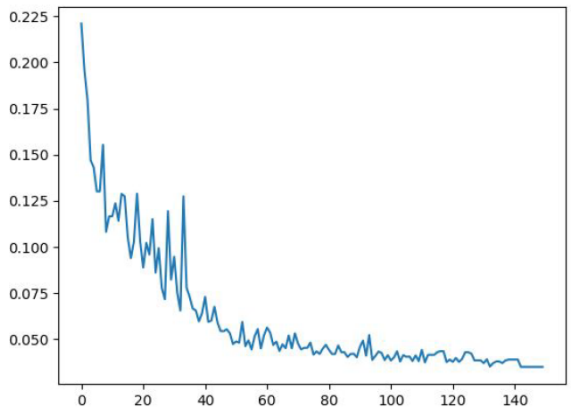

Figure 12. Loss learning curve for object detection.

To evaluate the performance of the detection, we computed the intersection over union (IoU) measure. The IoUs were computed on annotated videos not previously used for learning. Figure 13 shows examples of the annotated test videos and the bounding box of the detection.

Figure 13. Examples of UAV detection in outdoor tests.

Figure 14 shows the measured values of IoUs during an outdoor video test. We can see that the obtained IoU values vary between 0.7 and 0.91, meaning that the majority of the time, we have an

overlap between the detection result and the real position above the threshold of 70%. These IoU values decrease in the middle of the curve. This decrease corresponds to a UAV located far away and covering a smaller area of the image (so it is more difficult to detect).

To evaluate the performance of the algorithm for distant objects, we conducted some tests to evaluate the limits of the detection. We tested the detection confidence between 1 m and 6 m. We found that the confidence score for the detected object decreases with the increase in the distance from an average of 0.83 for 1 m to 0.35 for 6 m.

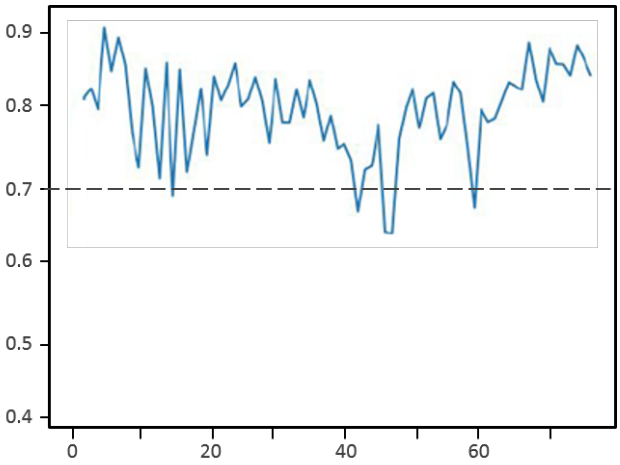

Figure 14. Evolution of intersection over union (IoU) measures during an outdoor test.

4.2.2. Object Detection with a Search Area Proposal (SAP)

Particle filters based search area proposal (SAP) were used to predict the next search area and improve the detection. Figure 15 shows the obtained IoU measures over the video frames. We can see that when the UAV is very close (start and end of the curve), the detection has a much lower performance. This is because the UAV area takes almost all the frame. In this scenario, performing predictions becomes difficult. However, we can see that in the middle (the area circled in red in Figure 15), when the UAV is far away, the IoU measure improves, with values between 0.8 and 0.9 compared to around 0.7 without SAP. This shows that the proposed search area proposal (SAP) is very useful when the target is small (far away and located at more than 4 m away in our experiments). However, the performance degrades when the target is at a closer distance (less than 2 m). Between these distances, the result is almost the same with and without SAP.

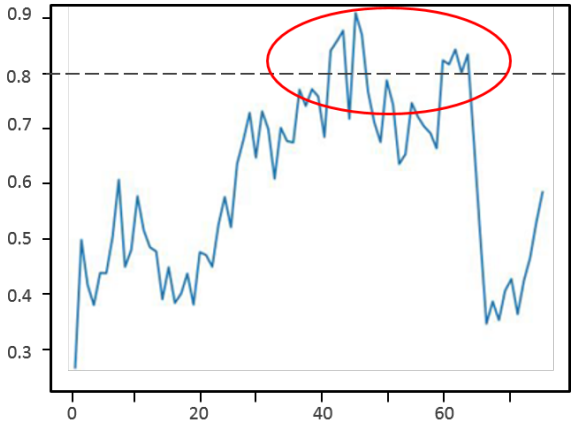

Figure 15. Evolution of IoU measures with search area proposal (SAP).

5. Conclusions

This work presents two approaches for UAV pursuit-evasion. The two approaches are based on the use of deep learning and convolutional neural networks (CNNs) as the core of the tracking architecture. The first approach uses deep reinforcement learning and a modified action decision network (ADNet). The algorithm predicts the actions for tracking the target based on previous actions and its last known position. The obtained results show that the proposed approach is effective in tracking moving objects in complex outdoor scenarios. Some limitations remain due to the processing time needed for action estimation, leading to some detection problems for fast UAV motions. The second algorithm uses deep object detection and a search area proposal (SAP) algorithm for prediction. A deep convolutional neural network (CNN) is used to detect the UAV and estimate its position. YOLO v2 was used as the object detector and was trained to detect UAVs. A search area proposal (SAP) based on particle filters was developed in order to predict the next position of the UAV using past detected positions. SAP improved the detection of distant UAVs. The obtained results demonstrate the efficiency of the proposed algorithms and show that both approaches are interesting for a UAV pursuit-evasion scenario. Future developments include improving the processing time of the reinforcement learning algorithm by adding a confidence score to predictions and a scale factor to the generated actions. This scale factor will help reduce the number of actions by improving the predicted distance used to move the follower UAV close to the target. For the SAP algorithm, an end-to-end approach will be studied to integrate predictions within the deep learning architecture and improve the processing time. Combining deep object detection, SAP, and reinforcement learning will also be considered. Finally, it could be interesting to evaluate the performance of other deep object detectors within the proposed framework.

Author Contributions: Conceptualization, M.A.A.; Funding acquisition, M.A.A.; Methodology, M.A.A.; Project administration, M.A.A.; Software, S.A. and A.B.; Supervision, M.A.A.; Validation, S.A. and A.B.; Writing—original draft, M.A.A., S.A., and A.B.; Writing—review & editing, M.A.A.

Funding: This research was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) grant number RGPIN-2018-06233.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Schmuck, P.; Chli, M. Multi-UAV collaborative monocular SLAM. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 3863–3870. [CrossRef]
2. Vemprala, S.; Saripalli, S. Monocular Vision based Collaborative Localization for Micro Aerial Vehicle Swarms. In Proceedings of the 2018 International Conference on Unmanned Aircraft Systems (ICUAS), Dallas, TX, USA, 12–15 June 2018; pp. 315–323. [CrossRef]
3. Shah, S.; Dey, D.; Lovett, C.; Kapoor, A. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In *Field and Service Robotics*; Springer: Cham, Switzerland, 2017.
4. Tian, Y.; Liu, K.; Ok, K.; Tran, L.; Allen, D.; Roy, N.; How, J.P. Search and Rescue under the Forest Canopy using Multiple UAS. In Proceedings of the International Symposium on Experimental Robotics (ISER), Buenos Aires, Argentina, 5–8 November 2018.
5. Chung, S.; Paranjape, A.A.; Dames, P.; Shen, S.; Kumar, V. A Survey on Aerial Swarm Robotics. *IEEE Trans. Robot.* **2018**, *34*, 837–855. [CrossRef]
6. Alexopoulos, A.; Schmidt, T.; Badreddin, E. A pursuit-evasion game between unmanned aerial vehicles. In Proceedings of the 2014 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO), Vienna, Austria, 1–3 September 2014; Volume 2, pp. 74–81. [CrossRef]
7. Krishnamoorthy, K.; Casbeer, D.; Pachter, M. Minimum time UAV pursuit of a moving ground target using partial information. In Proceedings of the 2015 International Conference on Unmanned Aircraft Systems (ICUAS), Denver, CO, USA, 9–12 June 2015; pp. 204–208. [CrossRef]

8. Alexopoulos, A.; Kirsch, B.; Badreddin, E. Realization of pursuit-evasion games with unmanned aerial vehicles. In Proceedings of the 2017 International Conference on Unmanned Aircraft Systems (ICUAS), Miami, FL, USA, 13–16 June 2017; pp. 797–805. [CrossRef]
9. Fargeas, J.C.L.; Kabamba, P.T.; Girard, A.R. Cooperative Surveillance and Pursuit Using Unmanned Aerial Vehicles and Unattended Ground Sensors. *Sensors* **2015**, *15*, 1365–1388. [CrossRef] [PubMed]
10. Fu, X.; Feng, H.; Gao, X. UAV Mobile Ground Target Pursuit Algorithm. *J. Intell. Robot. Syst.* **2012**, *68*, 359–371. [CrossRef]
11. Camci, E.; Kayacan, E. Game of Drones: UAV Pursuit-Evasion Game with Type-2 Fuzzy Logic Controllers tuned by Reinforcement Learning. In Proceedings of the 2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Vancouver, BC, Canada, 24–29 July 2016; pp. 618–625. [CrossRef]
12. Bonnet, A.; Akhloufi, M.A. UAV pursuit using reinforcement learning. In Proceedings of the SPIE, Unmanned Systems Technology XXI, Baltimore, MD, USA, 14–18 April 2019; Volume 11021, p. 1102109.
13. Watkins, C.J.C.H. Learning from Delayed Rewards. Ph.D. Thesis, King's College, Cambridge, UK, 1989.
14. Arola, S.; Akhloufi, M.A. Vision-based deep learning for UAVs collaboration. In Proceedings of the SPIE, Unmanned Systems Technology XXI, Baltimore, MD, USA, 14–18 April 2019; Volume 11021, p. 1102108.
15. Arola, S.; Akhloufi, M.A. UAV Pursuit-Evasion using Deep Learning and Search Area Proposal. In Proceedings of the IEEE International Conference on Robotics and Automation 2019—ARW, Montreal, QC, Canada, 20–24 May 2019; pp. 1–6.
16. Yun, S.; Choi, J.; Yoo, Y.; Yun, K.; Choi, J.Y. Action-Decision Networks for Visual Tracking with Deep Reinforcement Learning. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 1349–1358. [CrossRef]
17. Chatfield, K.; Simonyan, K.; Vedaldi, A.; Zisserman, A. Return of the Devil in the Details: Delving Deep into Convolutional Nets. *arXiv* **2014**, arXiv:1405.3531.
18. François-Lavet, V.; Henderson, P.; Islam, R.; Bellemare, M.G.; Pineau, J. An Introduction to Deep Reinforcement Learning. *Found. Trends Mach. Learn.* **2018**. [CrossRef]
19. Williams, R.J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* **1992**, *8*, 229–256. [CrossRef]
20. Levandowsky, M.; Winter, D. Distance between sets. *Nature* **1971**, *234*, 34–35. [CrossRef]
21. Daniel McNeela. The REINFORCE Algorithm Aka Monte-Carlo Policy Differentiation. Available online: http://mcneela.github.io/math/2018/04/18/A-Tutorial-on-the-REINFORCE-Algorithm.html (accessed on 15 March 2019).
22. Redmon, J.; Divvala, S.K.; Girshick, R.B.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788. [CrossRef]
23. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017; pp. 6517–6525. [CrossRef]
24. Akhloufi, M.; Regent, A.; Ssosse, R. 3D target tracking using a pan and tilt stereovision system. In Proceedings of the SPIE, Defense, Security, and Sensing, Airborne Intelligence, Surveillance, Reconnaissance (ISR) Systems and Applications X, Baltimore, MD, USA, 29 April–3 May 2013; Volume 8713, pp. 8713–8738.
25. Akhloufi, M. Pan and tilt real-time target tracking. In Proceedings of the Airborne Intelligence, Surveillance, Reconnaissance (ISR) Systems and Applications VII, Orlando, FL, USA, 5–9 April 2010; Volume 7668.
26. Dellaert, F.; Fox, D.; Burgard, W.; Thrun, S. Monte Carlo localization for mobile robots. In Proceedings of the 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C), Detroit, MI, USA, 10–15 May 1999; Volume 2, pp. 1322–1328. [CrossRef]
27. Dellaert, F.; Burgard, W.; Fox, D.; Thrun, S. Using the condensation algorithm for robust, vision-based mobile robot localization. In Proceedings of the 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149), Fort Collins, CO, USA, 23–25 June 1999; Volume 2, pp. 588–594.
28. Doucet, A. *On Sequential Simulation-Based Methods for Bayesian Filtering*; Technical Report; Signal Processing Group, Department of Engineering, University of Cambridge: Cambridge, UK, 1998.
29. Parrot SA. Bebop 2. Available online: https://www.parrot.com/us/drones/parrot-bebop-2#parrot-bebop-2-details (accessed on 15 March 2019).

30. Parrot SA. Mambo FPV. Available online: https://www.parrot.com/ca/drones/parrot-mambo-fpv (accessed on 15 March 2019).
31. Chollet, F. Keras. 2015. Available online: https://keras.io (accessed on 15 March 2019).
32. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Available online: tensorflow.org (accessed on 15 March 2019).