drones

MDPI

Article

Towards Fully Autonomous Drone Tracking by a Reinforcement Learning Agent Controlling a Pan–Tilt–Zoom Camera

Mariusz Wisniewski [1,*], Zeeshan A. Rana [1], Ivan Petrunin [1], Alan Holt [2] and Stephen Harman [3]

[1] Digital Aviation Research and Technology Centre (DARTeC), Cranfield University, Cranfield MK43 0JR, UK; zeeshan.rana@cranfield.ac.uk (Z.A.R.); i.petrunin@cranfield.ac.uk (I.P.)
[2] Department for Transport, London SW1P 4DR, UK
[3] Thales, Manor Royal, Crawley RH10 9HA, UK
* Correspondence: m.wisniewski@cranfield.ac.uk

Abstract: Pan–tilt–zoom cameras are commonly used for surveillance applications. Their automation could reduce the workload of human operators and increase the safety of airports by tracking anomalous objects such as drones. Reinforcement learning is an artificial intelligence method that outperforms humans on certain specific tasks. However, there exists a lack of data and benchmarks for pan–tilt–zoom control mechanisms in tracking airborne objects. Here, we show a simulated environment that contains a pan–tilt–zoom camera being used to train and evaluate a reinforcement learning agent. We found that the agent can learn to track the drone in our basic tracking scenario, outperforming a solved scenario benchmark value. The agent is also tested on more complex scenarios, where the drone is occluded behind obstacles. While the agent does not quantitatively outperform the optimal human model, it shows qualitative signs of learning to solve the complex, occluded non-linear trajectory scenario. Given further training, investigation, and different algorithms, we believe a reinforcement learning agent could be used to solve such scenarios consistently. Our results demonstrate how complex drone surveillance tracking scenarios may be solved and fully autonomized by reinforcement learning agents. We hope our environment becomes a starting point for more sophisticated autonomy in control of pan–tilt–zoom cameras tracking of drones and surveilling airspace for anomalous objects. For example, distributed, multi-agent systems of pan–tilt–zoom cameras combined with other sensors could lead towards fully autonomous surveillance, challenging experienced human operators.

Keywords: drone detection; drone tracking; pan–tilt–zoom; reinforcement learning; deep learning; machine learning; drone surveillance; unmanned aerial vehicles; airport security; artificial intelligence

check for updates

Citation: Wisniewski, M.; Rana, Z.A.; Petrunin, I.; Holt, A.; Harman, S. Towards Fully Autonomous Drone Tracking by a Reinforcement Learning Agent Controlling a Pan–Tilt–Zoom Camera. *Drones* **2024**, *8*, 235. https://doi.org/10.3390/drones8060235

Academic Editor: Anastasios Dimou, Arne Schumann, Lars Sommer, Dimitrios Zarpalas, Alessio Fascista and Angelo Coluccia

Received: 9 April 2024
Revised: 22 May 2024
Accepted: 24 May 2024
Published: 30 May 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The misuse of drones poses a risk to public infrastructure security, especially safety-critical operations such as airports. In 2018, a drone sighting disrupted the air traffic at Gatwick Airport for 3 days. The British Government laid out a counter-unmanned aircraft strategy [1] that sets out deterring, detecting, and disrupting the misuse of drones as a key objective. Through the research presented in this paper, we aim to address the detection of drones by automating the pan–tilt–zoom camera. This paper is an extension of the conference paper presented at AIAA SciTech 2023 [2] (the Drone Tracking Gym environment, along with the code used to train agents, can be found here: https://github.com/mazqtpopx/cranfield-drone-tracking-gym (accessed on 29 May 2024) and 10.17862/cranfield.rd.25568943).

The detection of drones using electro-optical sensors can more precisely be broken down into detection, tracking, and classification. Detection aims to find whether a drone exists in a current frame recorded by the camera. Tracking traditionally aims to correlate the movement of a drone across a stream of frames. Classification aims to find whether

the detected object is a drone, bird, or another flying object. If it is a drone, what kind of a drone is it, and what kind of payload is it carrying?

A key problem with drone detection is that monitoring of a vast volume of airspace is required. For example, in the UK, the CAA restricts UAS flights around aerodromes [3]. These restrictions make it illegal to fly a drone within 2 or 2.5 nautical miles around the aerodrome (depending on the aerodrome), as well as within 5 km from the end of each runway. Radars are a popular sensor for monitoring airspace and are extensively used within the aeronautical industry. However, traditional radars are designed to track large aircraft, not drones. Holographic radars [4–6] are a new technology, with an application for detecting drones. However, due to natural constraints, the radar signal may be affected by things such as local landscapes. To accurately monitor a safety-critical piece of airspace around an aerodrome, the fusion of some of these sensors is required to improve the detection and classification accuracy, and to fill in sensor gaps. Cameras are a traditional sensor type which can be used to detect objects. Recent advances in AI make the use of cameras for tasks such as the detection and classification of drones more viable, due to the use of convolutional layers. The control of pan–tilt–zoom (PTZ) cameras has traditionally been done by a human (i.e., a security operator), or an integrated system that combines elements of automated detection, tracking, and classification. We explore the feasibility of employing an agent trained using reinforcement learning (RL) to provide an end-to-end solution; if successful, effectively replacing human operators, and providing an alternative to integrated systems.

1.1. Camera Configurations and Control Systems

Different camera configurations, such as static cameras, matrix of cameras, and PTZ cameras, can be used to detect drones. The field of regard (FOR) is the total area that can be captured by a movable sensor. Field of view (FOV) is the angular cone perceivable by the sensor at a particular time. Static cameras are limited when monitoring large air space volumes because the FOR and the FOV overlap. A matrix of cameras increases the FOV, but it still overlaps with the FOR. Pan–tilt–zoom (PTZ) cameras have the FOV of a single camera but also have a larger FOR than their FOV. The FOR is limited by the physical constraints of the camera and motors for pan and tilt. We can consider the different camera configurations to be the following:

- Single static camera [7–9]: using a single static camera where the FOR overlaps with the FOV, to detect a drone in the video feed.
- Matrix of cameras [10,11]: using multiple cameras stacked together (the FOR overlaps with the FOV), to detect a drone in the video feed.
- PTZ cameras: using a camera on a motorized pan–tilt platform. The FOR is greater than the FOV.

Single static camera methods are limited because they can only cover a small area. An example of a drone dataset produced using this method is the drone-vs.-bird dataset [9]. Drones can be detected using a fine-tuned object detection network [12].

A matrix of cameras increases the coverage area of the system and increases the resolution of the data. Demir et al. [10] present such a system, and use a background subtraction method to detect any moving objects. A problem with this is that the high throughput of data requires a lot of processing power for a real-time system. They get around it by creating a special real-time processor that performs the background subtraction. The state of the art of drone detection has improved with the advent of convolutional neural networks, which improve the detection and classification accuracy compared with traditional methods such as background subtraction, but they do so at an increased computational cost and processing such a large image in real time may be computationally expensive.

PTZ cameras effectively make static cameras dynamic by allowing freedom of movement in the pan and tilt rotation. The advantage is that the FOR is increased, but the FOV stays the same (although, the FOV is lower than a matrix of cameras). However, it presents an optimization problem: assuming that at some time t, a drone flies within

the volume of interest (VOI) in front of the camera, there exists a range of pan/tilt/zoom positions that would allow the camera to accurately classify the drone. The question is: how should an automated system behave at each timestep in order to detect or track a drone? Expert human operators can achieve this by understanding the context and are able to correctly control the camera. The problem is made harder by the drone being a second-order dynamic system controlled by (presumably) another human, meaning that the movement of the drone is not entirely predictable.

Hence, using a PTZ camera to monitor a volume increases the complexity of the problem because a control mechanism is required to move the camera. The PTZ cameras can be operated by a human, via an automated control system. Further, the automated system can be split into two cases: an 'integrated' system: where the detection, tracking, and control of the camera are independent, integrated units, and an RL-based agent, which is an end-to-end solution and learns how to detect, track, and to control the camera on its own. Hence, we consider that the problem can be split into the following methods:

- Human camera control [13]: A human is controlling a camera to record a drone. This can be either a person filming a drone freehand or a human operator controlling a PTZ camera. The Anti-UAV dataset is an example of this, as it features videos of drones being followed by an operator. The FOR is larger than the FOV.
- Automated (integrated) PTZ camera control [14,15]: An automated control system for controlling the PTZ cameras to detect and track drones with no human operator. It relies on a drone detection mechanism and then a tracking mechanism for defining the control demands of the PTZ to keep the drone within the FOV. Some elements of this solution might be AI-based (e.g., the detection part).
- Automated (RL) PTZ camera control [2,16,17]: An end-to-end AI system that takes the camera image as an input and outputs the control request to the PTZ camera trained using RL.

Anti-UAV dataset [13] is an example of human camera control as it contains videos of flying drones recorded by a PTZ camera operated by a human. In this dataset, we have access to the ground truth (position of the drone in each of the video frames), and the environment changes as the field of view of the camera moves. However, we do not have the possibility of controlling the PTZ of the camera. Hence, such a dataset is useful for finding the accuracy of a detection, tracking, or classification mechanism, but it is impossible to produce and test actions for the camera to take because the videos are already recorded.

Liu et al. [14] show an example of a PTZ control system based on the position of the detected drone in the image. Their architecture consists of detecting flying objects in the video feed, calculating the trajectory, controlling the PTZ, and classifying the objects. Similarly, Svanstrom et al. [15] use a pan–tilt platform to control multiple sensors: a video camera, an infrared camera, a fisheye lens, and a microphone. By fusing the data from the sensors, they can correctly classify objects.

An RL agent has the potential to be an end-to-end solution that performs all three of these tasks (detection, tracking, and classification) at once. This is compared with what we call an integrated solution, which integrates separate detection, tracking, and classification mechanisms. The integrated solution (which we will use as a baseline) would use a trained detection architecture (e.g., Faster-RCNN, YOLO, as shown by Medina et al. [18]). Then a tracking algorithm (e.g., TrackFormer [19], SORT [20], or SiamRPN++ [21]) would be used to continuously track the object across the frames. The position of the drone within the camera image would determine whether to move the PTZ by a tuned PID controller—this could be done by finding the distance in x and y from the centre of the frame and setting them as control signals.

Sandha et al. [16] present Eagle, an application of using RL for controlling PTZ camera to track cars and humans. They train their policy on an environment within Unreal Engine. They use a discrete action space and their step reward is a function of the distance of the object to the center of the image multiplied by the size of the object and a clip parameter.

The state observation is the image from the PTZ camera. They successfully transfer their model to real-life scenarios and run their model on-board of a Raspberry Pi. Fahim et al. [17] also present an RL model for controlling a PTZ for the purpose of tracking human subjects. They train their policy within a simulator and evaluate it in real-world scenarios.

We can redefine the terms of detection, tracking, and classification specifically for the context of an automated PTZ agent attempting to find a drone within the VOI:

- Detection: does the drone exist in the airspace volume of interest?
- Camera-based-tracking: if we detect the drone at time t, can we continuously move the pan–tilt–zoom camera to record the drone while it is in the airspace volume of interest?
- Classification: how do we know that the object being detected by the pan–tilt–zoom camera is a drone and not a bird?

Crucially, camera-based tracking differs from standard tracking typically referred to in literature. While standard tracking focuses on matching points across video frames, we define camera-based tracking as actually moving a physical (or a simulated) camera to track the object (or objects). Further, as the RL agents learn from experience as opposed to relying on a predefined tracking algorithm, we hypothesize that they have the potential to better perform on context-dependent scenarios, such as during object occlusion.

But, a key problem with research involving PTZ cameras is reproducibility. To test the methods presented by other researchers, the same hardware setup is required. A potential way to solve this problem would be to create a simulated (synthetic) environment to test the ability of algorithms to detect, track, and classify flying drones. This synthetic environment would be dual-purpose; it would train RL agents and benchmark other tracking methods and human performance.

The use of synthetic environments for drone detection and classification has been tried in the literature. Scholes et al. [22] use a synthetic environment to generate a dataset to train a neural network to detect, segment, and classify drones. Wisniewski et al. [23] use synthetic images to classify drone models—DJI Phantom, DJI Mavic, and DJI Inspire. The synthetic dataset is used to train a convolutional neural network (CNN) which identifies the drone models in a real-life dataset. Hence, a similar approach could be used to create a synthetic environment which allows for the control of the PTZ camera to detect flying drones. This is common in other fields such as robotics, where simulations of robots are used to test algorithms and are later transferred to real life.

To summarize, the system that we consider in this paper is an RL agent to control the camera actions based on the camera image. The input into the system is an image from the camera, and the output is a PTZ action to control the camera. So far, we have defined the problem and explored differences in optical/mechanical systems used to detect drones. We continue by exploring the literature regarding RL and robotics.

1.2. Reinforcement Learning

RL is a popular method of optimizing a solution for problems in dynamic environments. It comes at the cost of being sample inefficient compared to other types of AI such as supervised learning. Big milestones in RL research include AlphaGo beating a human expert at the game GO [24], folding proteins [25], as well as beating humans in video games [26]. Other applications of RL are playing Atari 2600 games [27–29] by using deep Q-networks (DQN) and deep recurrent Q-networks (DQRN). The image from the game is passed through a convolutional neural network, and an action is predicted by the network. The trained agent can outperform humans in certain games such as Video Pinball.

Hausknecht et al. [30] also used DQRNs by adding a long-short-term-memory (LSTM) to the DQN model. They flickered the screens of the Atari games and found that the DQRN model could still perform. They present the use of RNNs as an alternative to standard DQNs by stacking the history of frames in the DQN input. However, recurrency on its own does not improve the performance of the agent in how well they play the games, and the same performance can be achieved with standard DQNs by stacking the history of observations in the input layer of the CNN.

Newer applications of RL include rotating objects using a robotic hand [31], playing Minecraft by using learning world-models [32], and controlling legged robots [33]. Lample and Chaplot [34] trained an AI agent that plays the FPS game Doom using only pixels on screen as input to the neural network. They used VizDoom [35], which is a wrapper of the game that allows for RL agents to learn to play it. Further, there exists a number of open-source environments for testing agents such as OpenAI gym [36]. RL environments can be split into discrete action space and continuous action space. In discrete problems, the agent can pick from a number of discrete actions every step. In continuous problems, the agent can select a range of values, usually ranging between -1 and $+1$ depending on the environment. Mirowski et al. show that RL can also be used for navigating mazes [37]. The agent uses discrete actions and is rewarded for reaching the goal, and for finding intermediate rewards. The best-performing network learns to predict depth and uses an LSTM layer.

There exists sparse literature on using RL for PTZ camera control for surveillance applications. A similar application that we have found literature on is drone control. Both of the problems take an image as an input and output as a control value. The difference is the difference in constraints between the two systems—the drone is free to control in the 6 DOF, while the PTZ camera is fixed in the XYZ axes, but free to pan and tilt (and zoom). While many drones also have a PTZ camera on a gimbal, we have not found any literature considering the control of this—instead, they focus on the control of the drone itself. Nonetheless, because of the similarities, we review the RL for drone control literature to investigate common themes and challenges in the application of RL algorithms.

CAD2RL [38] presents synthetic environments used to train an agent to control a drone flying through a maze without crashing into walls. An image is taken as an input and passed through a CNN to learn the Q-function. By employing domain randomization, the agent is able to transfer to real-world environments. Vorbach et al. [39] apply a continuous-time neural network to a drone chasing a target through a forest and find that it performs better than other architectures under challenging conditions such as fog or heavy rain. Kaufmann et al. [40] use deep RL to control racing drones around an indoor flying course. The drones are rewarded for correctly passing through the gates, and penalized for crashing. They train the model in a simulation and deploy it in the real world. For the observation, they combine the image, visual-inertial odometry, and gate detections using a Kalman filter. They show that the RL agents outperform expert humans, even in real-life experiments. Pham et al. [41] use cooperative and distributed RL for training drones to optimally inspect a field. Muñoz et al. [42] apply deep RL for the control of drones to reach a target within AirSim, a simulation of drones built on Unreal Engine. They used two action spaces: a discrete space where the drone can move forward, yaw left, and right, and a continuous space that allows the drone to move freely in the XYZ directions. The agent is rewarded for reaching the goal and penalized for collisions. They find that an architecture which concatenates the features of the image encoded through a convolutional neural network with other parameters like the velocity of the drone, distance to the goal, and geofencing coordinates performs the best. Akhloufi et al. [43] also apply RL for the control of drones for the problem of following another drone in a real environment. They use supervised learning (YOLO) to detect a drone in the image. The difference between the image centre and the detected target position (using the supervised learning network) is then input to the RL agent as an observation. They use the RL algorithm to reposition the bounding box to the position of the drone, to effectively create a drone tracking RL agent. Lastly, a drone controller tries to keep the drone target in the centre of the image.

Hence, we can see that the application of RL for drone control is a popular area of research and more importantly it appears that RL can be applied to these sorts of control problems. We fail to see a similar interest in the application of surveillance using PTZ cameras. A possible explanation is the lack of training and evaluation environments—something we present in this publication. We believe that the application of AI methods to

ground-based sensors, and in particular cameras, is more feasible than to drones as they do not have to pass the same regulatory scrutiny as airborne objects.

1.3. Research Gap, Hypothesis, and Contribution

Based on the discussed literature, we found there to be a sparse amount of research in the area of using PTZ cameras for tracking drones, and in the area of using RL to track objects. We have identified a gap in the literature—to the best of our knowledge there is no existing literature on tracking drones using an RL agent controlling a PTZ camera. While similar approaches of using RL for PTZ control have recently been attempted for the applications of tracking cars [16] or humans [17], we believe that the task of drone tracking presents unique challenges, and to our knowledge has not been attempted.

Hence, in this paper, we aim to address these two problems. First, we present a synthetic environment for drone tracking called *Drone Tracking Gym*. The environment was designed to train an RL agent and can interface with a Python client. The importance of the environment is twofold: first, it can be used for training RL agents. It contains the Gymnasium interface, allowing most RL algorithms to be trained. Second, it can be used to benchmark against any other method; for example, human input via a joystick, or any other non-RL-based tracking algorithms. To the best of our knowledge, there does not exist an equivalent environment for the purpose of testing training algorithms in the loop (drone tracking benchmarks rely on videos with human controllers tracking the drone by controlling the camera, such as the Anti-UAV dataset).

Based on the potential of RL shown in literature combined with our environment, we generate the following hypothesis: an RL agent should be capable of learning to control a PTZ camera to track a drone based on only the image input within our scenario.

Later, we present a working RL agent, trained in this environment, that controls the camera PTZ to correctly track the drone. To the best of the authors' knowledge, RL has not been tried for controlling PTZ cameras to track drones in the literature and provides the potential for end-to-end tracking compared with traditional systems.

1.4. Layout

In Section 2, we present the synthetic environment, the RL agent and the architecture used, and the interface we used to transfer the data between the environment and the agent. In Section 3, we present the results of the training of our RL agent. We also discuss how this compares with the literature. In Section 4, we conclude our results and, lastly, we suggest future work on how to improve this work in Section 5.

This paper is an extension of the conference paper presented at AIAA SciTech 2023 [2]. We substantially expand on the results section by updating the reward function. After successfully solving the Basic Tracking scenario, we seek to better understand the capabilities of the agent by adding two new scenarios: Dynamic Tracking and Obstacle Tracking. We train and evaluate the performance of the agent on these scenarios. Lastly, we add a 'Solved Scenario' value—i.e., what we would expect an experienced human operator to achieve in these scenarios to indicate whether each scenario is solved.

2. Methodology

We present an environment of a flying drone containing a controllable pan–tilt–zoom (PTZ) camera with the primary aim of using it to train a reinforcement learning (RL) agent. The holistic approach of this paper is shown in Figure 1. The approach is split into three parts: environment simulation used to simulate the flying drone and the PTZ camera used to track it, the RL model used to control the PTZ to track the drone, and finally the evaluation approach. Several scenarios were designed to train and evaluate the agent in different tasks: Basic Tracking, Dynamic Tracking, and Obstacle Tracking. The RL agent is trained on each environment, and compared with an optimal model. The training of an RL agent to control a PTZ camera to track a flying drone requires an environment, an agent, and an interface between the two. The environment contains the PTZ camera and a flying

drone. The agent is a neural network trained using an RL algorithm that learns to control how to move the camera. The interface controls the flow of data between the environment and the agent. The agent is rewarded for correctly tracking the drone and is penalized if it loses sight of the drone. In the following subsections, we explain the environment, the agent, and the interface in more detail.

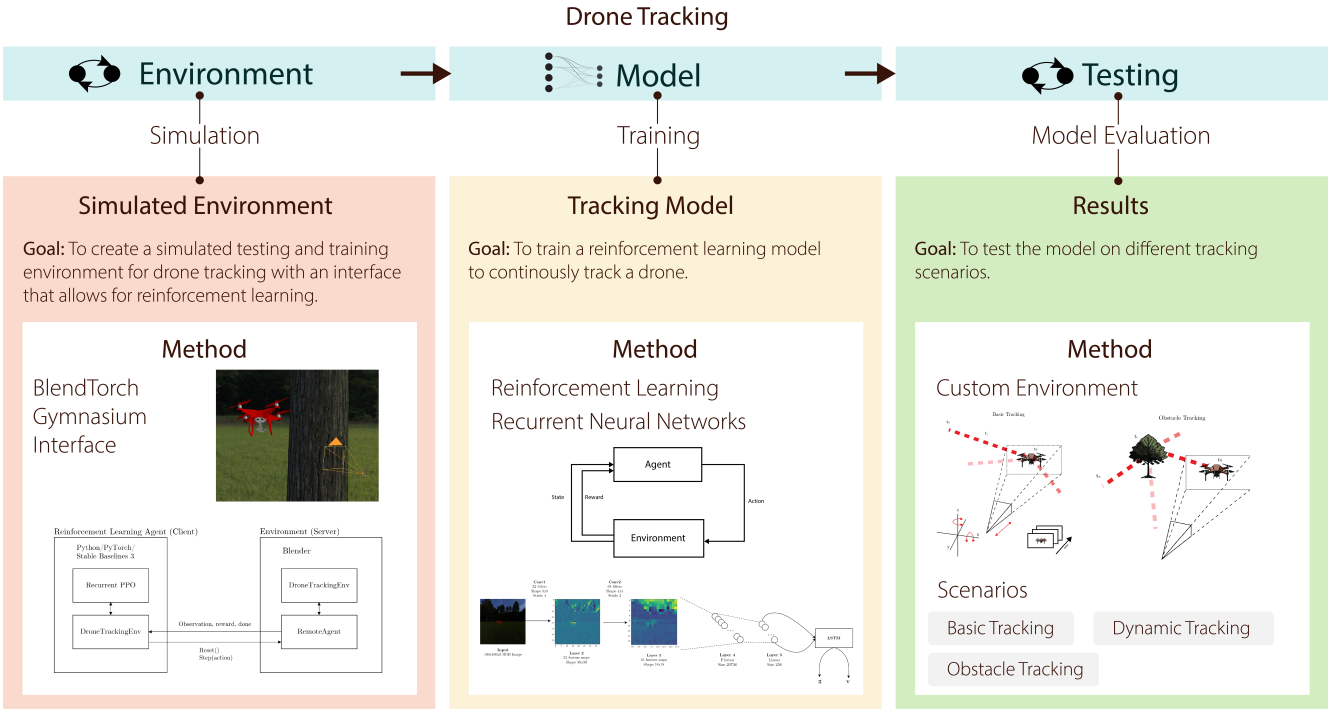


Figure 1. The holistic approach shown in this paper, split into three main sections: environment, the model, and testing. The environment is the simulation of the pan–tilt–zoom (PTZ) camera and the flying drone. This is carried out in Blender, combined with the Blendtorch Gymnasium Interface. The model used to control the PTZ is trained using reinforcement learning (RL). The model is trained and evaluated in multiple environments: Basic Tracking, Dynamic Tracking, and Obstacle Tracking.

2.1. Mathematical Description

Let us consider a camera sensor mounted on a pan–tilt mount, with parameters that can be controlled by an agent or a system. The view from the camera sensor at any time t is the Field of View (FOV), and is a function of the control parameters pan θ, tilt ω, and zoom z, which are output values of a system after receiving a control target from the agent:

$$FOV_t = f(\theta_t, \omega_t, z_t) \quad \theta, \omega, z \in \mathbb{R}$$

The total possible coverage of the sensor amounts together to the Field of Regard (FOR). This is the total volume that the sensor can operate in. Hence, $FOV \subseteq FOR$. Let Volume of Interest (VOI) be the volume in which any foreign flying drone shall be detected while it exists within the volume. For simplicity, assume the Volume of Interest (VOI) fully overlaps with the FOR ($VOI = FOR$).

The goal of the external agent/system is to control the PTZ parameters such that it optimally positions the sensors. It can achieve this by manipulating the θ, ω, and z values, such that it detects, and then tracks the drone D as it enters the VOI.

Assume the view from the camera is processed by an imperfect external classifier CL that finds the type of the object within the sensor view. The classifier will perform optimally when it receives a clear view of the object, which can be achieved by zooming the camera in. This narrows the FOV, and in turn reduces the the volume that the sensor is observing.

A larger FOV can be achieved by zooming the camera out, and reducing the clarity of any target, in turn lowering the classification confidence of CL.

The problem becomes an optimization problem; it is a trade off between capturing the targets entering the VOI (by keeping the zoom low), and correctly identifying the targets (by increasing the zoom). Hence, we can define the problem mathematically by calculating the next FOV position, FOV', as a maximization function between the classification CL, and the detection DET, which are both functions of the current FOV:

$$FOV_{t+1} = max(CL(FOV_t), DET(FOV_t))$$

where the agent attempts to both, maximize the accuracy of the classifier, whilst also maximizing the performance of the detection of objects. Becuase of the real-world, dynamic environment, the solution to this is not static and changes overtime based on environmental parameters.

The generic reward criteria of the scenario therefore shall therefore depend on the agents' ability to contain the drone within its viewport:

$$R = \begin{cases} 0 & \text{if drone not in VOI} \\ -\alpha & \text{if drone in VOI, not captured in FOV} \\ \beta & \text{if drone in VOI, captured in FOV} \end{cases}$$

(Note that, for our specific case, β is calculated by multiplying the amount of pixels that the drone occupies multiplied by a gradient map, and is described further in Section 2.2.4)

We do not believe that a direct mathematical solution exists to this problem, given the data from the camera sensor is highly dimensional, making the classification and detection of objects an imperfect process; and further, since there is no direct observation of the entirety of the VOI, the problem becomes a partially observable Markov decision process (POPMDP) [44]—i.e., a process which does not have the state observation of the whole environment. Finally, real environments are dynamic, making this trade off not a constant parameter, but rather a variable that changes with the environmental conditions.

Figure 2 shows two problems. A more general case, Figure 2a, in which a foreign object crosses the boundary of the VOI/FOR , and a simplified tracking case, Figure 2b, which assumes that the foreign object starts within the FOV of the camera.

A: Assume a foreign drone follows a certain trajectory such that it enters the VOI. The agent should guide the sensor to detect and track the object if it enters the VOI.

B: Assume that the object is already within the FOV. The agent should track the foreign flying object while it remains within the FOR.

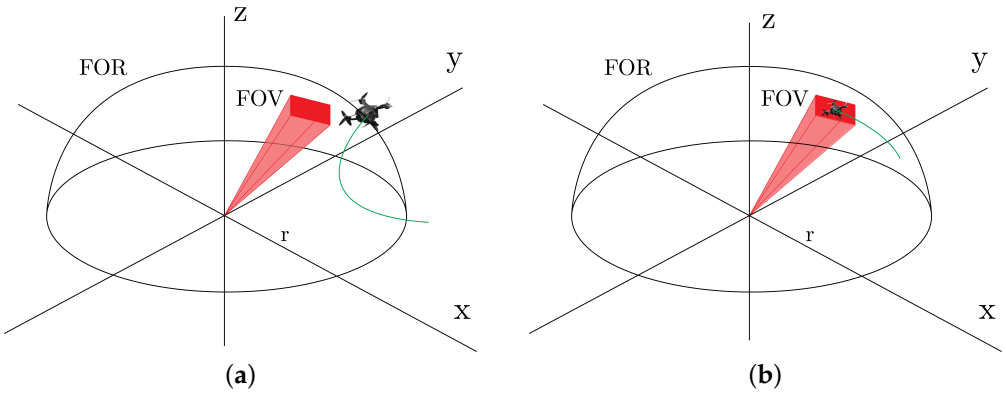


Figure 2. The general case of the problem, where a drone can enter the Field of Regard (FOR) of the camera sensor, and the goal of the agent is to optimize the positioning of the Field of View (FOV) placement (**a**), and the specialized case (**b**) where the drone is already in the FOV of the camera sensor, and the goal of the agent is to carry on the track.

In this paper, we consider the simplified case B. For simplicity, we also assume a static environment. However, we hope that our formulation of the problem and developments here lead to a more generalizable case that can be used to solve A. Solving the general case A would be hugely beneficial to the surveillance capabilities of PTZ cameras.

2.2. The Environment

Blender 3.4, a 3D modelling and animation software package, is used to create the 3D environment. Although it is not a game engine, it contains a rendering engine and a Python interface that allows for the programming of parameters. This allows us to create an observation (by rendering the image produced by a camera), change the pan, tilt, and zoom parameters of the camera, program the 3D model of the drone to move across the 3D space, and include some basic dynamic models for the camera and drone movement. To interface between the agent and Blender, BlendTorch [45,46], a framework that remotely launches the Blender environment and the PyTorch agent, is used. The interface is described in more detail in the original conference paper [2]. The environment implements a reset and a step function to be compatible with the interface. Reset returns the environment to the default settings, and the step defines how the environment changes every time step. The step function takes in an action and returns the observation, reward, and termination information. This is based on the Gymnasium interface [47].

A diagram of the environment step is shown in Figure 3. Each step, an action is generated by the policy. This action is input into the environment—setting a control target to the camera controller. The drone controller independently updates the position of the drone (the conditions of the drone targets are based on the specific scenario). The render engine creates an image of the scene, which becomes the observation state and is output from the environment. The reward calculator finds the area occupied by the drone in the image and calculates the reward for the time step (further described in Section 2.2.4). The environment outputs the image state, the reward, and a Boolean done value.

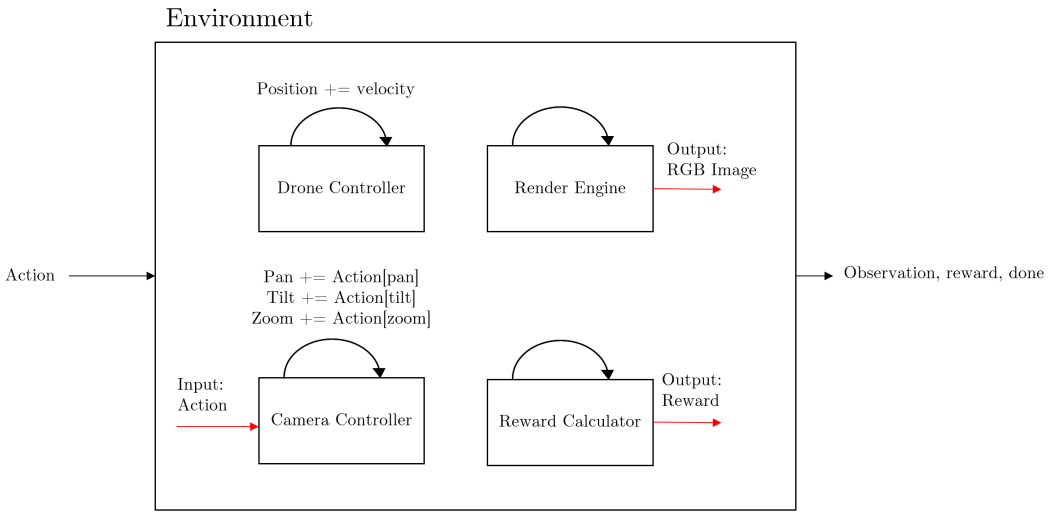


Figure 3. Environment step. At every step, an action is input into the environment. The camera controller takes the action and updates the camera's pan–tilt–zoom (PTZ) parameters. The drone controller updates the position of the drone. The render engine renders the scene and outputs an RGB image (which is the observation). The reward calculator finds the area occupied by the drone in the image and outputs the reward/termination based on this.

2.2.1. Scenarios

Figure 4 shows a representation of the three different scenarios. We test and evaluate the agent in each of them:

- **Basic Tracking**. A random velocity is assigned at the beginning of each scenario. The drone follows this trajectory throughout.

- **Dynamic Tracking A**. The movement of the drone is unpredictable. Every 30 steps (effectively, around 1 s) a new position target is assigned to the drone—hence changing the acceleration. Unlike in the basic tracking scenario where the trajectory is unchanging, we now have variable trajectories. The maximum drone velocity now exceeds that of the rotational velocity of the camera. Hence, it is possible for the drone to go completely out of bounds of the viewport, even when the requested rotational velocity is at its limit. Reward/termination mechanism is altered: termination only happens at 300 steps (and not when the drone is lost from the viewport). For every frame that a drone that is not visible in the viewport, -2 reward is accumulated. This allows the drone to disappear, and in principle, forces the agent to learn to re-detect the drone. This is a unique scenario in that it presents a challenge where the velocity of the drone might outpace that of the camera rotation. Human operators might be able to deal with this by understanding the expected path.
- **Dynamic Tracking B**. This is a variation of the Dynamic Tracking A scenario, with the dynamics of the drone tuned such that the maximum velocity should not outpace the maximum allowable rotational velocity of the camera. If the drone leaves the viewport, the agent is penalized (like in the Basic Tracking scenario).
- **Obstacle Tracking**. The drone flies behind an obstacle. When behind the obstacle, the drone waits for a random amount of steps, between 0–100. After waiting, it flies to a randomly assigned position. This scenario was designed to challenge the tracker on whether it can deal with a context-dependent scenario—of anticipating a drone to reappear from behind an obstacle.

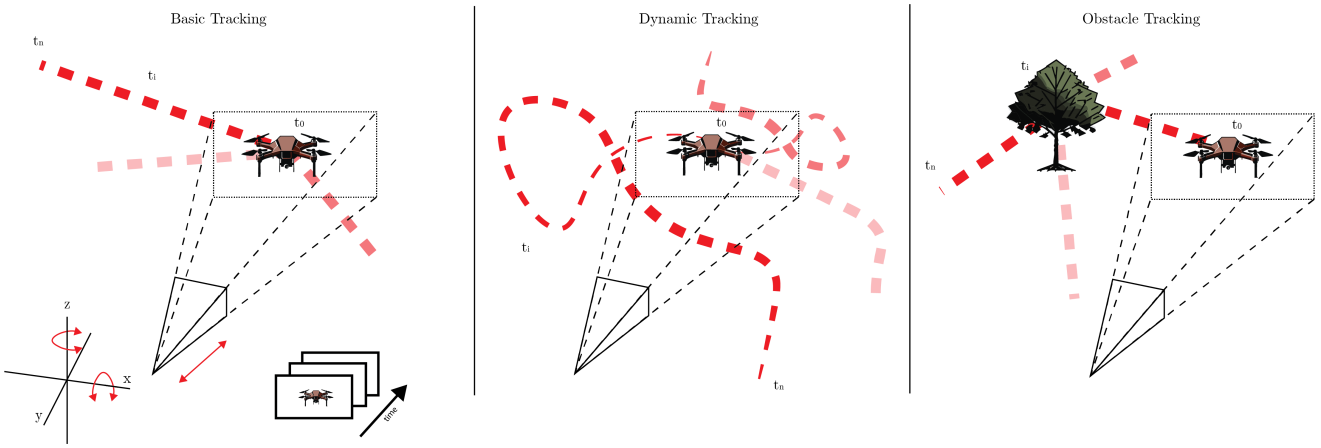


Figure 4. Different scenarios used to train and evaluate the RL agent. Basic tracking, dynamic tracking, and obstacle tracking. In basic tracking, the drone is assigned a random velocity at the start of the scenario and the velocity does not change until the end or termination. In dynamic tracking, the drone changes a waypoint every 30 steps. The drone is modelled as a second-order dynamic system, hence the acceleration is variable. In obstacle tracking, the drone flies behind an obstacle and then is assigned a random velocity when behind the obstacle.

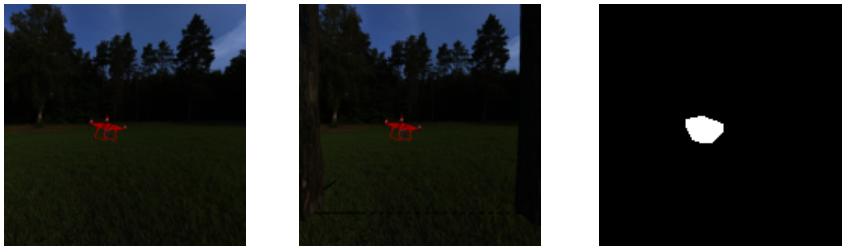
2.2.2. Observation

The image from the camera is rendered at every step. It produces a $160 \times 160 \times 3$ pixel RGB image.

Figure 5 shows the observation from the Basic and Dynamic Tracking scenarios (Figure 5a), Obstacle Tracking scenario (Figure 5b), and the mask (Figure 5c) used to internally calculate the reward. The environment is created by using a low-poly drone model, an HDRI as a background, and some basic tree stump models for the Obstacle Tracking scenario to act as obstacles to occlude the drone. The Eeevee rendering engine is used to render the environment.

2.2.3. Action

A continuous action space is used. The possible actions are pan, tilt, and zoom. They have limits set between −1 and +1, and each action is a floating point value. Table 1 shows the continuous actions, their range and how they affect the environment. The physical PTZ camera limits are: tilt [30, 330]°, and zoom [15, 200] mm. Yaw is unlimited. Within the environment, the camera is modelled as a second-order dynamic system to imitate real-life control. In every step, the input action is translated to a request for a position change of the PTZ position.



(**a**) (**b**) (**c**)

Figure 5. The observation from the Basic and Dynamic Tracking scenarios (**a**), observation from the Obstacle Tracking scenario (**b**), and the associated segmentation mask (**c**) used to internally calculate reward within the environment.

Table 1. Continuous action space—action, range and environment response.

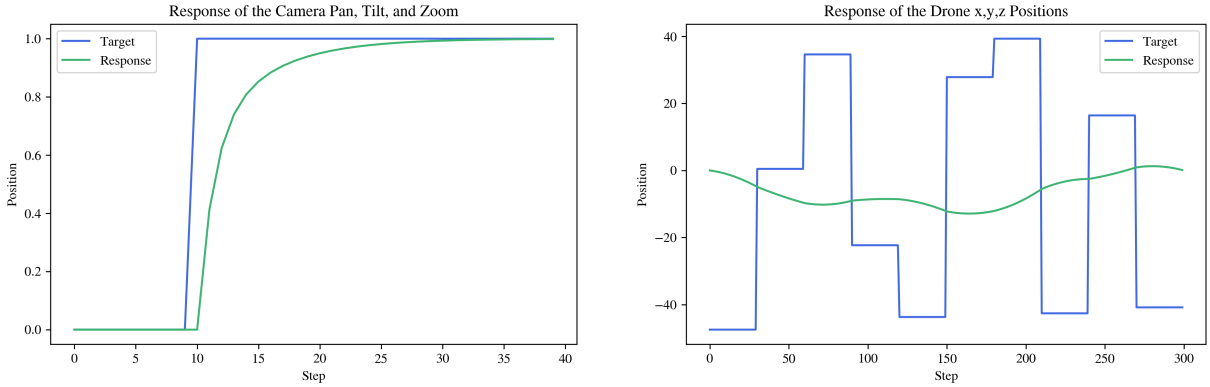
Action	Range	Environment Response
Pan	$[-1, +1]$	+1 translates to +0.02 radians pan to the right
Tilt	$[-1, +1]$	+1 translates to +0.02 radians tilt up
Zoom	$[-1, +1]$	+1 translates to +1 mm of focal length

In the original conference paper, the drone and the camera were effectively modelled as a direct response mechanism. This means that an input of 1.0 resulted in a maximum response in the following step. This is not realistic, as physical systems typically have some form of a damped response. This resulted in some problems; when zoomed in on the drone, the agent would continuously flicker between the right and left. At high zoom levels, this resulted in flickering, not representative of a real system. The response of the camera angle and zoom targets is shown in Figure 6a. This effectively results in the output being damped. Likewise for the drone, for the dynamic scenarios, we wanted it to have more temporally realistic dynamics. The response for the x,y, and z positions of the drone is shown in Figure 6b. These are the settings used in Dynamic Tracking B and Obstacle Tracking. For the Dynamic Tracking A scenario, the response is less damped and may outpace the maximum angular velocity of the camera.

2.2.4. Reward and Termination

The reward is given to the agent for every step that the agent correctly tracks the drone. In the conference paper we used a simple reward mechanism where if the area was greater than 20 px^2, the agent was rewarded +1. If the area was less than 1 px^2, the agent was rewarded −100. Instead, we found a better way to reward the agent, which resulted in better convergence. We create a circular gradient of the same size as the image with white pixels in the middle and black pixels towards the boundaries of the image. This image is multiplied by the segmentation mask of the drone generated within the environment. The point of this is to reward the agent more if it keeps the drone towards the centre of the image. Further, the bigger the proportion of the drone that is kept within the bounds of the image, the bigger the reward—this in turn allows for a higher quality image to be fed

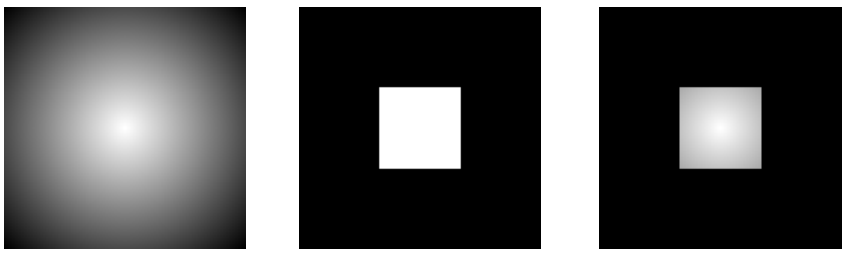
into a hypothetical classifier (which would distinguish if it is a bird, drone, or what kind of a drone is it etc.). If the agent loses sight of the drone, it is penalized and the episode is terminated.



(**a**) (**b**)

Figure 6. The input and response of the systems for pan, tilt, and zoom controls for the camera (**a**), and for x, y, z, target positions of the drone (**b**).

Figure 7 shows an example of the circular gradient being multiplied by a hypothetical mask to produce a reward calculation. In Figure 7c, all of the pixels are summed together and divided by 200 to produce the final reward. The hypothetical mask, shown in Figure 7b, contains a mask which is meant to represent an optimal observation (it is not taken directly from the environment) and is modelled a rectangle of the size of a quarter of the area of the image, an resides in the centre of the image. This represents an 'ideal' observation—and is meant to be hard to obtain continuously for the entire episode. The total area of the white rectangle is $1/9 \times 160 \times 160 = 2916.0$. When multiplied it results in 2395.2. Divided by 200, it results in an 'optimal' reward of 12.0 per scenario. The absolute maximum possible value, assuming every pixel in the image is occupied, is 58.7. In our reward plots in the results section, we use this optimal value as a baseline for an optimal reward in a scenario. This value is calculated by multiplying 12.0 by the number of steps in a scenario. Note that the optimal reward is meant as a guideline, and we model it as what we expect a hypothetical experienced human to be able to achieve in such a scenario. This value is labelled as 'Solved Scenario' in the episode reward graphs in the results section.



(**a**) (**b**) (**c**)

Figure 7. The circular gradient (**a**) is multiplied by the mask of the drone (an example mask (**b**)), to produce the final reward which takes a sum of all of the pixels as shown in (**c**). The sum is divided by 200 to produce a reward of 12.0, which we use as the 'optimal' reward and reference it as a baseline in our results. This value is labelled as 'Solved Scenario' in the episode reward graphs.

2.3. Reinforcement Learning Algorithms

To train the agent, we use Stable Baselines 3 [48], which provides implementations of common RL algorithms. The library is based on PyTorch [49]. We choose a recurrent PPO [50] with an LSTM layer as our algorithm. PPO is an on-policy algorithm that optimizes

the objective function using stochastic gradient ascent by batching the observation data and updating the gradients in multiple epochs (as opposed to only updating the gradient once per data sample). The original paper shows that PPO outperforms other policy gradient methods and performs well in environments with a continuous action space.

PPO was chosen to train our agent because it is proven to work well on continuous tasks. Our environment is complex because it involves tracking an object in 3D space, which requires some sort of knowledge of the prior states in the system, for example, to estimate the velocity of the object, and predict where to move the camera. Mnih et al. [27] found that stacking frames worked for DQN. Another solution to this problem is proven to be the use of recurrent networks [29]. We decided to use a recurrent neural network because this is consistent with similar studies [34,37]. Hence, we use the recurrent PPO algorithm, which uses a long short-term memory (LSTM) layer. For the training, we use mostly default hyperparameters, provided by Stable Baselines 3 (we change the learning rate to 0.00075).

In the conference paper [2], we compared the performance of PPO vs. PPO with LSTM and found that PPO did not converge on the basic tracking problem, while PPO with LSTM did converge. We do not present these results here and instead build on top of them by continuing to use the PPO LSTM algorithm. We hypothesized that because our problem involves dynamic elements in the 3D space, an understanding of the velocity between the video frames is required. This could either be achieved by stacking multiple observations on top of each other, or by using recurrence. By adding an LSTM layer, we are adding recurrence to our network, which is able to solve our environment.

Figure 8 shows the architecture of our neural network. Because we use an image as an observation, we first use an encoder that consists of convolutional layers to extract features. These features are then input into an LSTM and an actor–critic multilayer perceptron which predicts the value and the policy.

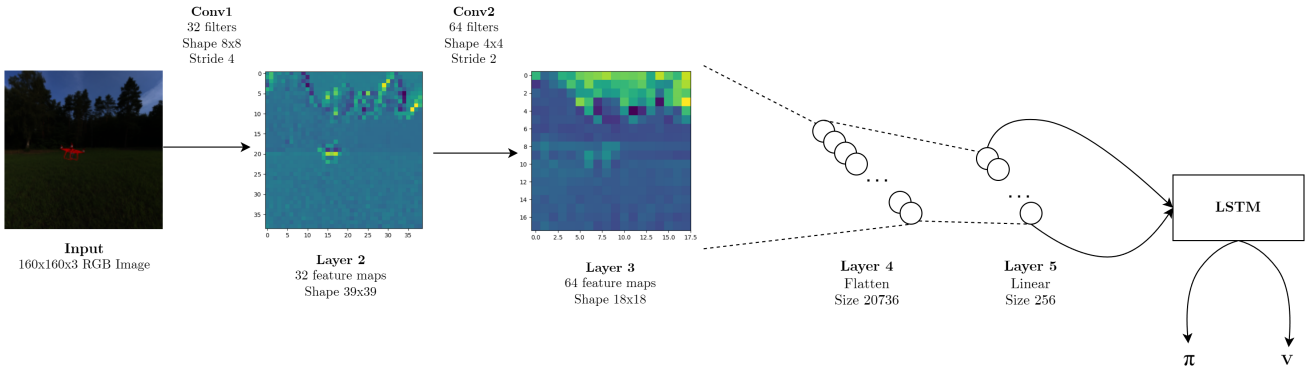


Figure 8. Architecture of the neural network. The input is an RGB image received from the environment. It is passed through an encoder which consists of two convolutional layers. The features from layer 3 are flattened in layer 4 and outputted to 256 features in layer 5. These features are input into an LSTM layer and an actor–critic multilayer perceptron which predicts the value and the policy.

Table 2 shows the hyperparameters used for training of the model. Other hyperparameters are taken as defaults from the Recurrent PPO implementation from Stable Baselines 3.

Table 2. Training hyperparameters.

Hyperparameter	Value
Learning Rate	0.000075
Batch Size	256
LSTM Layers	1
Encoded Image Features Dimension	256

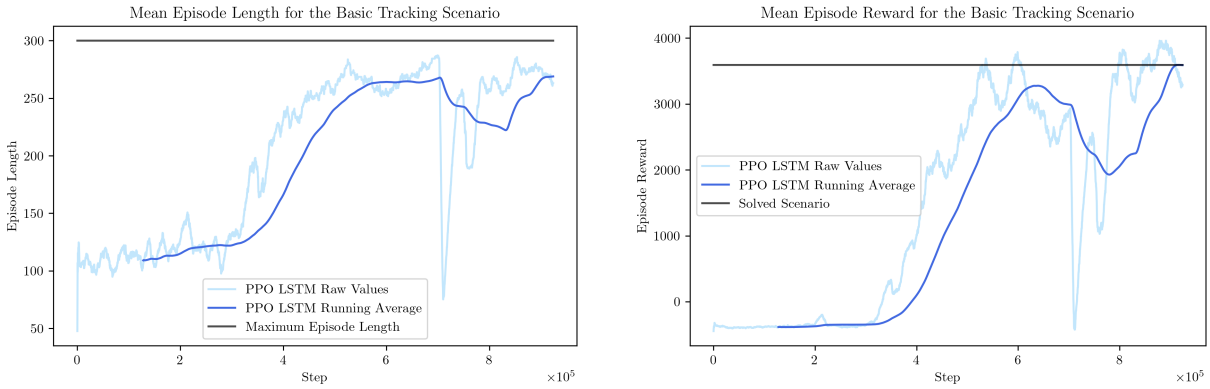
3. Results

The RL agent described in Section 2.3 is trained on the scenarios described in Section 2.2.1. Quantitative results showing the mean episode length and mean episode reward are presented, alongside qualitative results acquired by investigating the actions of the agent during an episode.

3.1. Basic Tracking

In the basic tracking scenario, the drone starts in front of the PTZ camera and is assigned a random constant velocity for 300 steps.

Figure 9 shows the results of the RL training for the Basic Tracking scenario. The agent reaches a maximum mean reward of almost 3960 per episode with less than 1,000,000 training steps. The mean episode length reaches a maximum of 287 steps after 699,000 training steps. This is almost 300, which is the maximum possible value for this scenario. Overall, we can see that the agent converges to an optimum solution, and is able to learn to correctly track the drone. At points, the agent outperforms the 'Solved Scenario' benchmark value. This is a sign that the agent learns to zoom in strongly on the drone, more so than the optimal discussed in Section 2.2.4.



(**a**) (**b**)

Figure 9. Average episode length (**a**) and average episode reward (**b**) for the Basic Tracking scenario.

By looking at the best-performing model (which achieves a mean reward of 3960 at step 891,000) we can qualitatively evaluate its performance during testing.

Figure 10 shows an example of a successful tracking episode. The agent correctly tracks the drone for the entirety of 300 frames and keeps it zoomed in in the viewport—theoretically allowing a classifier to identify the object.



(**a**) (**b**) (**c**) (**d**) (**e**)

Figure 10. Viewport from the camera at (**a**) frame 1, (**b**) frame 75, (**c**) frame 150, (**d**) frame 225, (**e**) frame 300 during an episode of the basic tracking scenario. The agent correctly tracks the drone for the entire 300 frames in this episode.

Figure 11 shows an example of a failure mode of the agent. In this example, the drone flies straight towards the camera. As it flies past it goes out of sight of the viewport. This is not necessarily the fault of the agent, but rather that of the environment, as the drone flies too close to the camera. It is likely due to episodes such as this that the agent does not reach the maximum possible value of mean episode length of 300.



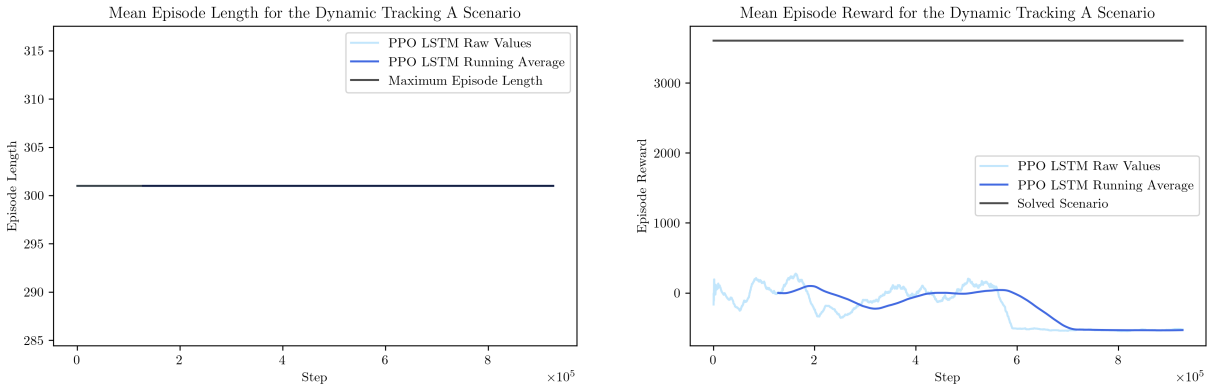
(**a**) (**b**) (**c**)

Figure 11. Viewport from the camera at (**a**) frame 1, (**b**) frame 50, (**c**) frame 83 during an episode of the basic tracking scenario. The agent fails at frame 84 as the drone goes out of the viewport.

3.2. Dynamic Tracking A

In the Dynamic Tracking A scenario, the termination conditions are removed. Every episode lasts 300 steps and the agent's goal is to accumulate as much reward in this time. The speed of the drone can outpace that of the maximum angular velocity of the camera, resulting in the agent having to potentially re-detect the drone. We started this scenario with the best weights from the Basic Tracking scenario.

Figure 12 shows the results of the RL training for the Dynamic Tracking A scenario. The agent appears to struggle to learn in this scenario. Finally, after 600,000 steps, the performance of the agent drops significantly—the learned policy is to zoom out and look around in circles. The mean reward is just above -500, which is the termination penalty (and the minimum possible reward for this scenario). We can investigate the last model (after 900,000 steps).



(**a**) (**b**)

Figure 12. Average episode length (**a**) and average episode reward (**b**) for the dynamic tracking A scenario.

Figure 13 shows an example of unsuccessful learning during the Dynamic Tracking A scenario. The agent does not attempt to track the drone, as in the Basic Tracking scenario, and instead goes around in circles. This shows the challenge of designing RL scenarios. By making the scenario too challenging we can completely 'confuse' the agent. The agent may learn to solve this scenario given enough training steps, but we have also failed to see evidence of any learning. This is also outside the computing and time resources we had available for this publication. We hypothesize that for the agent to learn to solve this scenario, it needs to learn to detect the drone. Instead, in the Basic Tracking scenario, the agent was simply learning to track a drone that was already in its viewport. Still, a human is still able to re-detect the drone as it goes out of the viewport, hence why we believe that this scenario is solvable.



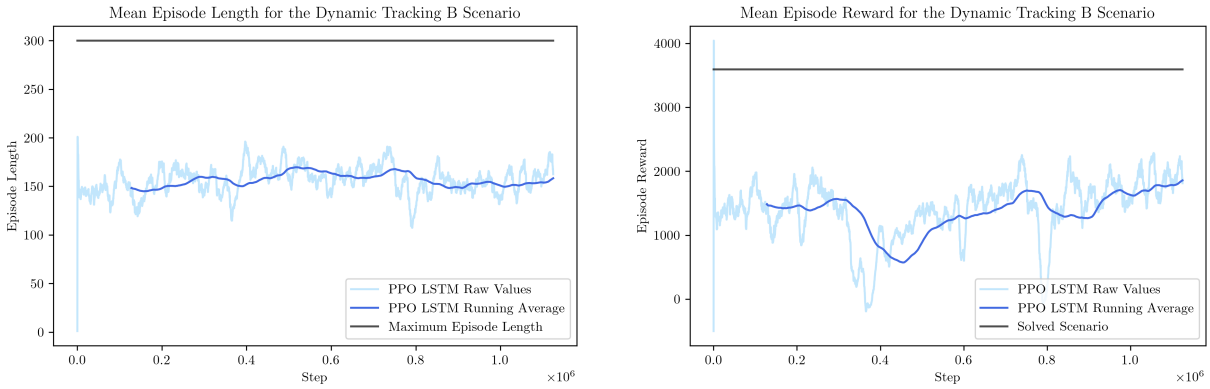
(**a**) (**b**) (**c**) (**d**)

Figure 13. Viewport from the camera at (**a**) frame 1, (**b**) frame 75, (**c**) frame 225, (**d**) frame 300 during an episode of the Dynamic Tracking A scenario. The agent fails to learn to track the drone in this episode and instead turns around in circles.

3.3. Dynamic Tracking B

In the Dynamic Tracking B scenario, the speed of the drone is slowed down so that it does not outpace the camera (compared with Dynamic Tracking A). As in the Basic Tracking scenario, the agent terminates after it loses sight of the drone. The agent starts this scenario with the best weights from the Basic Tracking scenario.

Figure 14 shows the results of the RL training for the Dynamic Tracking B scenario. Overall, the agent does not learn significantly from the starting weights. After 2,000,000 steps there is a slight bump in the episode length and episode reward, but it is not very significant. The scenario is harder than the Basic Tracking scenario as evidenced by a mean episode length of around 150 at the start. Overall, we see that the agent has not been able to learn to fully solve this environment. We can investigate the best model (after 2,370,000 steps), which achieves a 2490 mean reward.



(**a**) (**b**)

Figure 14. Average episode length (**a**) and average episode reward (**b**) for the dynamic tracking B scenario.

Figure 15 shows the results of the best agent performing an episode where it unsuccessfully tracks the drone for the entirety of the episode.



(**a**) (**b**) (**c**) (**d**) (**e**)

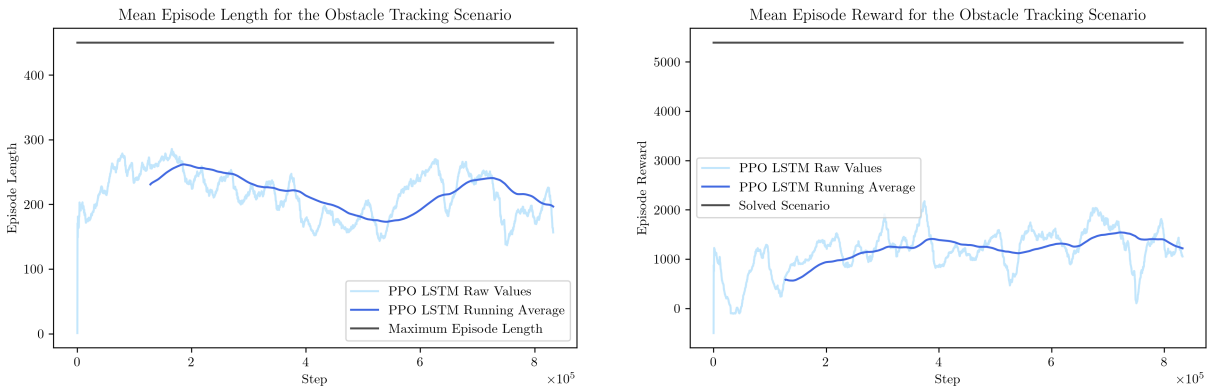
Figure 15. Viewport from the camera at (**a**) frame 1, (**b**) frame 75, (**c**) frame 150, (**d**) frame 250, (**e**) frame 286 during an episode of the Dynamic Tracking B scenario. The agent correctly tracks the drone for most of the frames but the drone leaves the viewport in frame 287 in this episode.

In general, the agent attempts to maximize the zoom on the drone, which affects its ability to deal with the dynamic non-linear movements of the drone. The challenge with this scenario is that it is better, in the long run, to have a safer strategy and reduce the zoom. However, the agent appears to have failed to learn this and attempts to maximize the zoom, as in the Basic Tracking scenario, leading to a strategy with a higher risk of the drone leaving the viewport.

3.4. Obstacle Tracking

In the Obstacle Tracking scenario, four tree obstacles are added around the PTZ camera. The drone flies behind a random tree, waits between between 0–100 steps, and then flies to a random point. The maximum length of the scenario is increased to 450 steps, and after 300 steps the reward is doubled.

Figure 16 shows the results of the RL training for the Obstacle Tracking scenario. The results show a slow drift in the increase of mean episode reward. However, the mean episode length fails to increase significantly to the maximum of 450, and instead hovers between 200–300 steps. This suggests that, on average, the agent fails to re-detect the target after it hides behind the obstacle. The reward plot shows that the agent falls very short of the 'Solved Scenario' benchmark value, mostly due to the inability of the agent to get close to the maximum of 450 mean episode length. This results in the agent also accumulating −500 penalties for termination. We can investigate the best model (after 375,000 steps) which achieves a 2180 mean reward and 247 mean steps for signs of the agent learning to track objects behind obstacles.



(**a**) (**b**)

Figure 16. Average episode length (**a**) and average episode reward (**b**) for the obstacle tracking scenario.

Figure 17 shows an example of a successful tracking episode. The drone flies behind the tree around step 128 as shown in Figure 17c. As the drone waits behind the obstacle for a random time, the agent correctly waits looking at the tree. In step 275, shown in Figure 17e, the drone reappears from behind the tree. The agent correctly identifies the movement and subsequently tracks the drone. In step 408, shown in Figure 17h, the drone flies behind another tree, and the agent loses track in step 428, shown in Figure 17i. Although the agent did not reach the end of the scenario and received a negative reward at the end, it shows a sign of learning to track the drone behind an obstacle.



(**a**) (**b**) (**c**) (**d**) (**e**)

(**f**) (**g**) (**h**) (**i**)

Figure 17. Viewport from the camera at (**a**) frame 1, (**b**) frame 100, (**c**) frame 128, (**d**) frame 200, (**e**) frame 275, (**f**) frame 300, (**g**) frame 400, (**h**) frame 408, (**i**) frame 428 during an episode of the Obstacle Tracking scenario. The agent correctly tracks the drone for 428 frames in this episode, although loses sight of the drone as it gets lost behind another obstacle and does not reach 450 frames.

This presents a question: why is the agent able to learn to lose track of the drone behind an obstacle, but not completely, as in the Dynamic Tracking A scenario? A possible answer is that the agent still accumulates reward if it looks at the drone from behind the obstacle, meaning it simply learns to look at obstacles. But, in Dynamic Tracking A, if the agent loses sight of the drone it is not rewarded. We hypothesized that the agent might

remember to do nothing when looking at the tree because it cannot re-detect the drone (as shown in dynamic tracking A). Because the time that it takes for the drone to hide behind the obstacle is fixed, this is possible. But, the time that the drone waits behind the obstacle is random (between 0–100 steps), and the agent is still able to continue the tracking after the drone re-appears. This suggests that the agent is re-detecting the drone.

We have found that the agent can learn to track the drone for one tree at a time, but then fails on others (there are four trees around, and the drone flies randomly behind one of them every episode). This is a sign that there is a limitation either in the network or in the number of steps during training. If we designed the scenario to only contain one obstacle we believe the agent would be able to learn this. The challenge here comes with the randomness and unpredictability, requiring the agent to understand the context. We have shown that our agent shows signs of being able to learn to track behind obstacles, although it does not do this consistently.

Further, it is possible that just like in Lample and Chaplot [34] and Mirowski et al. [37], a decoder-like network with a separate loss would benefit the training. At the moment, the network is not explicitly learning to detect the drone. In Lample and Chaplot, the authors showed that the detection of enemies greatly improved the performance of the agent. We did not formally investigate this, but instead set it as a suggestion for further work.

3.5. Discussion

To summarize the quantitative results, the PPO LSTM agent can learn to track the drone in the Basic Tracking scenario. It fails in the Dynamic Tracking A scenario, where the drone often outpaces the maximum camera angular velocity. It can track the drone on the Dynamic Tracking B scenario, although falls short of being able to do so consistently for the entirety of the episode, often loses sight of the drone, and falls short of our 'Solved Scenario' benchmark value. The agent fails to solve the Obstacle Tracking scenario, falling way short of the maximum steps and the hypothetical model, but qualitatively shows signs of learning to track the drone in select instances. Given more training time, hyperparameter tuning, and/or more exotic model/method selection, we believe RL has the potential to fully solve every one of these environments.

Examining the failure of the agent to converge in the complex and occluded scenarios suggests that the RL agent either struggles to learn real-world dynamics or is unable to learn the uncertainty. The key difference between the Basic Tracking and Dynamic Tracking B scenarios is that, in the former, the drone follows a linear path with a constant velocity, while in the latter, it follows a non-linear trajectory as the drone follows a random point with a second-order dynamic model, meaning it has an unpredictable velocity profile. Neural networks are known for being able to learn the dynamics of dynamic models from data [51,52]. The question then becomes whether the RL models can learn these dynamics based on the image input only. While it is impossible to answer this question based on the presented data, it is a possibility that model-based RL algorithms might perform better here. If the agent has a model of how the drone moves, it may be able to predict the movements better than the current approach. Model-based RL algorithms such as DreamerV3 [32], which builds a world model using an autoencoder based on the image inputs, might outperform the PPO LSTM model in these scenarios if the thesis that the current agent has failed to learn the dynamics of the drone is correct. The other possibility is that the reason for the failure to converge is the large amount of randomness in the scenario. In the Basic Tracking scenario, the assigned velocity of the drone at the start of the scenario is random, and the agent can learn this. This seems to counter the randomness argument. However, Mirowski et al. [37] shows that increasing randomness (e.g., by randomizing the starting position and orientation of the agent, or by randomizing the goal) makes it harder for the agent to solve the scenario. Mapping that to our results, it could be the case that the Basic Tracking scenario has a small amount of randomness that is learnable by the agent, while the Dynamic Tracking B and Obstacle Tracking scenarios contain too much randomness for the agent and the agent fails to solve these scenarios. Hence, there exist valid arguments

for both the learning complex dynamics hypothesis and the randomness hypothesis. Both may affect learning and, hence, require further study.

Sim-to-real, the transfer of a neural network model trained in a simulated environment to real-world scenarios, was attempted by Sanhda et al. [16] and is a natural progression from simulations as presented here. A difference with our approach is that we used a continuous action space and investigated different types of scenarios. Their success in sim-to-real suggests that it is possible to transfer tracking policies to real-world scenarios. Domain randomization [53], i.e., the use of randomizing simulation parameters such as the object textures, was also used by Sanhda et al. to improve the sim-to-real transfer. Domain Randomization is something that could be expanded in our simulated environment to enable sim-to-real transfer and improve generalizability.

4. Conclusions

To conclude, we have presented a novel simulated environment designed to train reinforcement learning (RL) agents to track drones and to benchmark tracking methods in the loop. We believe this is a key challenge that could aid the advancements of fully autonomous surveillance cameras for the detection and tracking of drones. During the literature review, we were challenged with the lack of benchmarks and metrics for successful pan–tilt–zoom (PTZ) tracking. Hence, we focused on creating a simulated environment and appropriate scenarios, to aid the data availability in this area. We have presented different scenarios in this simple environment, which challenge the tracking method in different ways: from simple linear trajectories to tracking occluded objects with non-linear trajectories. We did not present a solution to all of the environments in this publication and found limitations of the current RL method. By understanding these limitations, we hope that further research can lead to overcoming them. We release the environment as a challenge to other researchers to solve the environments. We believe advances in the ability of RL agents to track airborne objects within a simulation may lead towards the generalizability of scanning the skies around airports for anomalous flying objects in real environments.

5. Further Work

There are several things to consider for further work, ranging from technical improvements to the environment, studying the performance of reinforcement learning (RL) algorithms, to expanding the concept to more complex systems with multiple agents.

Regarding the technical improvements to the environment, a study into the reward function and its effect on the convergence of training could be done. At the moment, the reward function incentivizes the agent to zoom in on the drone as much as possible, without considerations for increasing the length of the tracking. Saturating the reward might solve this, or using some sort of inverse discount function (i.e., increasing the reward in the future). Rewarding the agent at the end of the scenario, which is more similar to the navigation scenarios reward, is another way of rewarding the agent and it is something we did not investigate.

Regarding the RL algorithms, a benchmark of different methods (such as SAC and TD3 for a single process), or increasing the number of processes (to make full use of PPO) could be performed. A thorough ablation study of hyperparameters, investigating repeatability, could be performed to better understand their effect on this environment. More exotic RL methods could be investigated—Liquid Time-Constant [54], model-based methods [32] which build world-models and learn to predict the expected future observations might be a good application for this problem, or human imitation learning [55]. It would be also interesting to further investigate why the agent can re-track the drone in the Obstacle Tracking scenario (and not on the Dynamic Tracking A scenario) using explainable AI techniques.

Creating a distributed system of autonomous pan–tilt–zoom (PTZ) cameras for the surveillance of a particular area could be deployed. This could be carried out in a similar

fashion to Pham et al. [41] who used cooperative and distributed RL for training drones to optimally inspect a field. If successful, the combination of multiple types of sensors to surveil an area would be possible—from radar to different configurations of cameras. The scenarios we presented here focus on the tracking of a single drone. However, what if there exists more than one drone? Which one should the camera prioritize? In a distributed autonomous PTZ system, maybe it is possible to track a number of drones at once. Other camera configurations could also be considered. For example, putting the camera on a moving platform (e,g. a car, or drone) and having the camera controller act as an independent agent taking actions to surveil the area would increase the complexity of the task. Further, utilizing a multi-camera system enables the accurate localization of the drone/foreign flying objects [56,57].

We have focused on tracking an object that already exists in a viewport. But, we did not consider the detection of the object within the volume of interest (VOI). Again, the detection in such a case would differ from traditional, single-frame detection. In this scenario, the agent would have to explore the 3D volume to find the object. Lastly, a challenge with simulated environments is sim-to-real, i.e., deploying the trained agents to real-world scenarios. Using techniques such as domain randomization [58,59], we believe the agent could be deployed to real PTZ cameras to track real drones.

Author Contributions: Conceptualization, M.W., Z.A.R. I.P., A.H, and S.H; methodology, M.W.; software, M.W.; validation, M.W.; formal analysis, M.W.; investigation, M.W.; resources, Z.A.R.; data curation, M.W.; writing—original draft preparation, M.W.; writing—review and editing, M.W., Z.A.R., I.P., A.H, and S.H; visualization, M.W.; supervision, Z.A.R. and I.P.; project administration, Z.A.R. and I.P.; funding acquisition, Z.A.R. All authors have read and agreed to the published version of the manuscript.

Funding: This project was funded through the UK Government's Industrial PhD Partnership (IPPs) Scheme by the Future Aviation Security Solutions Programme (FASS)—a joint Department for Transport and Home Office venture, in collaboration with Aveillant Ltd. and Autonomous Devices Ltd.

Data Availability Statement: The Drone Tracking Gym environment, along with the code used to train agents, can be found here: https://github.com/mazqtpopx/cranfield-drone-tracking-gym (accessed on 29 May 2024) and 10.17862/cranfield.rd.25568943.

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

Abbreviations

The following abbreviations are used in this manuscript:

FOV	Field of View
FOR	Field of Regard
LSTM	Long Short-Term Memory
PTZ	Pan–Tilt–Zoom
RL	Reinforcement Learning
VOI	Volume of Interest

References

1. UK Counter-Unmanned Aircraft Strategy. p. 38. https://assets.publishing.service.gov.uk/media/5dad91d5ed915d42a3e43a13/Counter-Unmanned_Aircraft_Strategy_Web_Accessible.pdf (accessed on 29 May 2024)
2. Wisniewski, M.; Rana, Z.A.; Petrunin, I. Reinforcement Learning for Pan-Tilt-Zoom Camera Control, with Focus on Drone Tracking. In Proceedings of the AIAA SCITECH 2023 Forum, National Harbor, MD, USA & Online, 23–27 January 2023. https://doi.org/10.2514/6.2023-0194.
3. UAS Airspace Restrictions Guidance and Policy **2022**.
4. Jahangir, M.; Baker, C. Robust Detection of Micro-UAS Drones with L-Band 3-D Holographic Radar. In Proceedings of the 2016 Sensor Signal Processing for Defence (SSPD), Edinburgh, UK, 22–23 September 2016; pp. 1–5. https://doi.org/10.1109/SSPD.2016.7590610.

5. Doumard, T.; Riesco, F.G.; Petrunin, I.; Panagiotakopoulos, D.; Bennett, C.; Harman, S. Radar Discrimination of Small Airborne Targets Through Kinematic Features and Machine Learning. In Proceedings of the 2022 IEEE/AIAA 41st Digital Avionics Systems Conference (DASC), Portsmouth, VA, USA, 18–22 September 2022; pp. 1–10. https://doi.org/10.1109/DASC55683.2022.9925778.
6. White, D.; Jahangir, M.; Wayman, J.P.; Reynolds, S.J.; Sadler, J.P.; Antoniou, M. Bird and Micro-Drone Doppler Spectral Width and Classification. In Proceedings of the 2023 24th International Radar Symposium (IRS), Berlin, Germany, 24–26 May 2023; pp. 1–10. https://doi.org/10.23919/IRS57608.2023.10172408.
7. Seidaliyeva, U.; Akhmetov, D.; Ilipbayeva, L.; Matson, E.T. Real-Time and Accurate Drone Detection in a Video with a Static Background. *Sensors* **2020**, *20*, 3856. https://doi.org/10.3390/s20143856.
8. Mueller, T.; Erdnuess, B. Robust Drone Detection with Static VIS and SWIR Cameras for Day and Night Counter-UAV. In Proceedings of the Counterterrorism, Crime Fighting, Forensics, and Surveillance Technologies III, Strasbourg, France, 9–12 September 2019; p. 10. https://doi.org/10.1117/12.2537940.
9. Coluccia, A.; Fascista, A.; Schumann, A.; Sommer, L.; Ghenescu, M.; Avenue, A.O.; Piatrik, T. Drone-vs-Bird Detection Challenge at IEEE AVSS2019. In Proceedings of the 16th IEEE International Conference on Advanced Video and Signal Based Surveillance, AVSS 2019, Taipei, Taiwan, 18–21 September 2019; p. 7.
10. Demir, B.; Ergunay, S.; Nurlu, G.; Popovic, V.; Ott, B.; Wellig, P.; Thiran, J.P.; Leblebici, Y. Real-Time High-Resolution Omnidirectional Imaging Platform for Drone Detection and Tracking. *J. -Real-Time Image Process.* **2020**, *17*, 1625–1635. https://doi.org/10.1007/s11554-019-00921-7.
11. Liu, H.; Wei, Z.; Chen, Y.; Pan, J.; Lin, L.; Ren, Y. Drone Detection Based on an Audio-Assisted Camera Array. In Proceedings of the 2017 IEEE Third International Conference on Multimedia Big Data (BigMM), Laguna Hills, CA, USA, 19–21 April 2017; pp. 402–406. https://doi.org/10.1109/BigMM.2017.57.
12. Mediavilla, C.; Nans, L.; Marez, D.; Parameswaran, S. Detecting Aerial Objects: Drones, Birds, and Helicopters. In Proceedings of the Artificial Intelligence and Machine Learning in Defense Applications III, Online Only, Spain, 13–18 September 2021; p. 18. https://doi.org/10.1117/12.2600068.
13. Jiang, N.; Wang, K.; Peng, X.; Yu, X.; Wang, Q.; Xing, J.; Li, G.; Zhao, J.; Guo, G.; Han, Z. Anti-UAV: A Large Multi-Modal Benchmark for UAV Tracking. *arXiv* **2021**, arXiv:2101.08466. http://arxiv.org/abs/2101.08466.
14. Liu, Y.; Liao, L.; Wu, H.; Qin, J.; He, L.; Yang, G.; Zhang, H.; Zhang, J. Trajectory and Image-Based Detection and Identification of UAV. *Vis. Comput.* **2020**, *37*, 1769–1780. https://doi.org/10.1007/s00371-020-01937-y.
15. Svanstrom, F.; Englund, C.; Alonso-Fernandez, F. Real-Time Drone Detection and Tracking With Visible, Thermal and Acoustic Sensors. In Proceedings of the 25th International Conference on Pattern Recognition, ICPR 2020, Virtual Event / Milan, Italy, 10–15 January 2021
16. Sandha, S.S.; Balaji, B.; Garcia, L.; Srivastava, M. Eagle: End-to-end Deep Reinforcement Learning Based Autonomous Control of PTZ Cameras. In Proceedings of the 8th ACM/IEEE Conference on Internet of Things Design and Implementation, San Antonio, TX, USA, 9–12 May 2023; pp. 144–157. https://doi.org/10.1145/3576842.3582366.
17. Fahim, A.; Papalexakis, E.; Krishnamurthy, S.V.; K. Roy Chowdhury, A.; Kaplan, L.; Abdelzaher, T. AcTrak: Controlling a Steerable Surveillance Camera Using Reinforcement Learning. *ACM Trans. -Cyber-Phys. Syst.* **2023**, *7*, 1–27. https://doi.org/10.1145/3585316.
18. Isaac-Medina, B.K.S.; Poyser, M.; Organisciak, D.; Willcocks, C.G.; Breckon, T.P.; Shum, H.P.H. Unmanned Aerial Vehicle Visual Detection and Tracking Using Deep Neural Networks: A Performance Benchmark. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW), Montreal, BC, Canada, 11–17 October 2021; pp. 1223–1232. https://doi.org/10.1109/ICCVW54120.2021.00142.
19. Meinhardt, T.; Kirillov, A.; Leal-Taixe, L.; Feichtenhofer, C. TrackFormer: Multi-Object Tracking with Transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, 18–24 June 2022; p. 16.
20. Bewley, A.; Ge, Z.; Ott, L.; Ramos, F.; Upcroft, B. Simple Online and Realtime Tracking. In Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 25–28 September 2016; pp. 3464–3468, https://doi.org/10.1109/ICIP.2016.7533003.
21. Li, B.; Wu, W.; Wang, Q.; Zhang, F.; Xing, J.; Yan, J. SiamRPN++: Evolution of Siamese Visual Tracking with Very Deep Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, 16–20 June 2019.
22. Scholes, S.; Ruget, A.; Mora-Martin, G.; Zhu, F.; Gyongy, I.; Leach, J. DroneSense: The Identification, Segmentation, and Orientation Detection of Drones via Neural Networks. *IEEE Access* **2022**, *10*, 38154–38164. https://doi.org/10.1109/ACCESS.2022.3162866.
23. Wisniewski, M.; Rana, Z.A.; Petrunin, I. Drone Model Classification Using Convolutional Neural Network Trained on Synthetic Data. *J. Imaging* **2022**, *8*, 218.
24. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. https://doi.org/10.1038/nature16961.

25. Jumper, J.; Evans, R.; Pritzel, A.; Green, T.; Figurnov, M.; Ronneberger, O.; Tunyasuvunakool, K.; Bates, R.; Žídek, A.; Potapenko, A.; et al. Highly accurate protein structure prediction with AlphaFold. *Nature* **2021**, *596*, 583–589. https://doi.org/10.1038/s41586-021-03819-2.
26. Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv* **2019**, arXiv:1912.06680. http://arxiv.org/abs/1912.06680.
27. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with Deep Reinforcement Learning. *arXiv* **2013**, arxiv:1312.5602. http://arxiv.org/abs/1312.5602.
28. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-Level Control through Deep Reinforcement Learning. *Nature* **2015**, *518*, 529–533. https://doi.org/10.1038/nature14236.
29. Chen, C.; Ying, V.; Laird, D. Deep Q-Learning with Recurrent Neural Networks. https://cs229.stanford.edu/proj2016/report/ChenYingLaird-DeepQLearningWithRecurrentNeuralNetwords-report.pdf (Accessed on 29 May 2024) p. 6.
30. Hausknecht, M.; Stone, P. Deep Recurrent Q-Learning for Partially Observable MDPs. *arXiv* **2017**, arxiv:1507.06527. http://arxiv.org/abs/1507.06527.
31. Qi, H.; Yi, B.; Suresh, S.; Lambeta, M.; Ma, Y.; Calandra, R.; Malik, J. General In-Hand Object Rotation with Vision and Touch. In Proceedings of the Conference on Robot Learning, CoRL 2023, Atlanta, GA, USA, 6–9 November 2023.
32. Hafner, D.; Pasukonis, J.; Ba, J.; Lillicrap, T. Mastering Diverse Domains through World Models. *arXiv* **2023**, arxiv:2301.04104. http://arxiv.org/abs/2301.04104.
33. Kumar, A.; Fu, Z.; Pathak, D.; Malik, J. RMA: Rapid Motor Adaptation for Legged Robots. In Proceedings of the Robotics: Science and Systems XVII, Robotics: Science and Systems Foundation, Virtual Event, 12–16 July 2021. https://doi.org/10.15607/RSS.2021.XVII.011.
34. Lample, G.; Chaplot, D.S. Playing FPS Games with Deep Reinforcement Learning. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017.
35. Kempka, M.; Wydmuch, M.; Runc, G.; Toczek, J.; Jaśkowski, W. ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning. In Proceedings of the 2016 IEEE Conference on Computational Intelligence and Games (CIG), Santorini, Greece, 20–23 September 2016.
36. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. OpenAI Gym. *arXiv* **2016**, arXiv:1606.01540. http://arxiv.org/abs/arXiv:1606.01540.
37. Mirowski, P.; Pascanu, R.; Viola, F.; Soyer, H.; Ballard, A.J.; Banino, A.; Denil, M.; Goroshin, R.; Sifre, L.; Kavukcuoglu, K.; et al. Learning to Navigate in Complex Environments. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017,
38. Sadeghi, F.; Levine, S. CAD2RL: Real Single-Image Flight without a Single Real Image. In Proceedings of the Robotics: Science and Systems XIII, Massachusetts Institute of Technology, Cambridge, MA, USA,12–16 July 2017.
39. Vorbach, C.; Hasani, R.; Amini, A.; Lechner, M.; Rus, D. Causal Navigation by Continuous-time Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, virtual, 6–14 December 2021,
40. Kaufmann, E.; Bauersfeld, L.; Loquercio, A.; Müller, M.; Koltun, V.; Scaramuzza, D. Champion-Level Drone Racing Using Deep Reinforcement Learning. *Nature* **2023**, *620*, 982–987. https://doi.org/10.1038/s41586-023-06419-4.
41. Pham, H.X.; La, H.M.; Feil-Seifer, D.; Nefian, A. Cooperative and Distributed Reinforcement Learning of Drones for Field Coverage. *arXiv* **2018**, arXiv:1803.07250. http://arxiv.org/abs/1803.07250.
42. Muñoz, G.; Barrado, C.; Çetin, E.; Salami, E. Deep Reinforcement Learning for Drone Delivery. *Drones* **2019**, *3*, 72. https://doi.org/10.3390/drones3030072.
43. Akhloufi, M.A.; Arola, S.; Bonnet, A. Drones Chasing Drones: Reinforcement Learning and Deep Search Area Proposal. *Drones* **2019**, *3*, 58. https://doi.org/10.3390/drones3030058.
44. Morad, S.; Kortvelesy, R.; Bettini, M.; Liwicki, S.; Prorok, A. POPGym: Benchmarking Partially Observable Reinforcement Learning. *arXiv* **2023**, arXiv:2303.01859. http://arxiv.org/abs/2303.01859.
45. Heindl, C.; Brunner, L.; Zambal, S.; Scharinger, J. BlendTorch: A Real-Time, Adaptive Domain Randomization Library, In Proceedings of the Pattern Recognition, ICPR International Workshops and Challenges, Virtual Event, 10–15 January 2021.
46. Heindl, C.; Zambal, S.; Scharinger, J. Learning to Predict Robot Keypoints Using Artificially Generated Images. In Proceedings of the 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Zaragoza, Spain, 10–13 September 2019.
47. Towers, M.; Terry, J.K.; Kwiatkowski, A.; Balis, J.U.; Cola, G.d.; Deleu, T.; Goulão, M.; Kallinteris, A.; KG, A.; Krimmel, M.; et al. 2023. *Gymnasium (v0.28.1)*. Zenodo. https://doi.org/10.5281/zenodo.8127026.
48. Raffin, A.; Hill, A.; Gleave, A.; Kanervisto, A.; Ernestus, M.; Dormann, N. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *J. Mach. Learn. Res.* **2021**, *22*, 1–8.
49. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Proceedings of the Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, Vancouver, BC, Canada, 8–14 December 2019; p. 12.

50. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347. http://arxiv.org/abs/1707.06347.
51. Kosmatopoulos, E.; Polycarpou, M.; Christodoulou, M.; Ioannou, P. High-Order Neural Network Structures for Identification of Dynamical Systems. *IEEE Trans. Neural Netw.* **1995**, *6*, 422–431. https://doi.org/10.1109/72.363477.
52. Chow, T.; Fang, Y. A Recurrent Neural-Network-Based Real-Time Learning Control Strategy Applying to Nonlinear Systems with Unknown Dynamics. *IEEE Trans. Ind. Electron.* **1998**, *45*, 151–161. https://doi.org/10.1109/41.661316.
53. Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; Abbeel, P. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 23–30. https://doi.org/10.1109/IROS.2017.8202133.
54. Hasani, R.; Lechner, M.; Amini, A.; Rus, D.; Grosu, R. Liquid Time-constant Networks. *arXiv* **2020**, arXiv:2006.04439. http://arxiv.org/abs/2006.04439.
55. Team, D.I.A.; Abramson, J.; Ahuja, A.; Brussee, A.; Carnevale, F.; Cassin, M.; Fischer, F.; Georgiev, P.; Goldin, A.; Gupta, M.; et al. Creating Multimodal Interactive Agents with Imitation and Self-Supervised Learning. *arXiv* **2022**, arXiv:2112.03763. http://arxiv.org/abs/2112.03763.
56. Srigrarom, S.; Sie, N.J.L.; Cheng, H.; Chew, K.H.; Lee, M.; Ratsamee, P. Multi-Camera Multi-drone Detection, Tracking and Localization with Trajectory-based Re-identification. In Proceedings of the 2021 Second International Symposium on Instrumentation, Control, Artificial Intelligence, and Robotics (ICA-SYMP), Bangkok, Thailand, 20–22 January 2021; p. 6.
57. Koryttsev, I.; Sheiko, S.; Kartashov, V.; Zubkov, O.; Oleynikov, V.; Selieznov, I.; Anohin, M. Practical Aspects of Range Determination and Tracking of Small Drones by Their Video Observation. In Proceedings of the 2020 IEEE International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), Kharkiv, Ukraine, 6–9 October 2020; pp. 318–322. https://doi.org/10.1109/PICST51311.2020.9468047.
58. Loquercio, A.; Kaufmann, E.; Ranftl, R.; Dosovitskiy, A.; Koltun, V.; Scaramuzza, D. Deep Drone Racing: From Simulation to Reality with Domain Randomization. *IEEE Trans. Robot.* **2020**, *36*, 1–14, https://doi.org/10.1109/TRO.2019.2942989.
59. Tremblay, J.; Prakash, A.; Acuna, D.; Brophy, M.; Jampani, V.; Anil, C.; To, T.; Cameracci, E.; Boochoon, S.; Birchfield, S. Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization. *arXiv* **2018**, arXiv:1804.06516. http://arxiv.org/abs/1804.06516.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.



https://dspace.org/

Towards fully autonomous drone tracking by a reinforcement learning agent controlling a pan–tilt–zoom camera

Wisniewski, Mariusz; Rana, Zeeshan A.; Petrunin, Ivan; Holt, Alan; Harman, Stephen

2024-06-01
MDPI
https://doi.org/10.3390/drones8060235
Wisniewski M, Rana ZA, Petrunin I, et al., (2024) Towards fully autonomous drone tracking by a reinforcement learning agent controlling a pan–tilt–zoom camera. Drones, Volume 8, Issue 6, June 2024, Article number 235

Downloaded from a DSpace repository. https://dspace.org/