Journal of Computer Engineering & Information Technology

Research Article

A SCITECHNOL JOURNAL

A Drone-Based Deep Learning Framework for Detecting and Tracking Objects

Muhammad Shoaib[1], Nasir Sayed[2]

Abstract

In recent years, integrating artificial intelligence and unmanned aerial vehicles (UAVs) has become a hot topic of study, especially where UAVs must conduct complex tasks that cannot be completed quickly under human control. Drones often use several sensors to gather full details about conditions, such as a top-down camera or LiDAR sensors, and the main processor measures all of the drone's trajectories. This paper proposes tracking a detected target that employs a monocular on-board camera and a reinforcement learning model. This system is more cost-effective and adaptable to the atmosphere using various sensors and pre-calculated trajectories than previous approaches. Our model added encompassing box details to the drive network picture input by extending the previous Deep Double Q network with the Duel Architecture Model (D3QN), modifying an action table and incentive feature, enabling 3-dimensional gestures and object recognition combined with MobileNet's support. The simulations are carried out in various simulation settings, each with its level of difficulty and sophistication. The "Airsim" application, a Microsoft-supported quadrotor simulation API, is used for research. The findings reveal that using a convergence-based exploration algorithm, the model approaches the observed object, a human figure, without reaching any barriers along the way and is moved faster.

Keywords

Deep Reinforcement Learning; Object Detection; Computer Vision and Unmanned Aerial Vehicles.

Introduction

Artificial intelligence has grown in strength and been extensively investigated to assess machine learning capability after the advent of in-depth learning and increased efficiency in the graphics processing unit (GPU). Intelligent agents are likely to be deployed on moving equipment such as a ground robot as a ground unit and a drone as an air unit, particularly in the case of human-machine interaction (HCI), where the actual implementation of a detailed understanding occurs. Furthermore, autonomous activities such as navigation [1], aerial mapping [2], item distribution, and so on are encouraged. In general, achieving such a target entails combining data from several sensors and analyzing it for trajectory calculation, obstacle prediction, and collision avoidance. Navigation and collision avoidance are complex tasks since the robot agent must handle the situation every second of the picture and make the best judgment possible in that short amount of time. Furthermore, since the physical world is generally fluid and dynamic, the construction of several sensors across space to capture and map the agent's data is disrupted.

The full implementation of many utilities to gather case-specific details and algorithms is ludicrous after realizing the real obstacles. As a result, even though it is complicated, a general-purpose algorithm that can cooperate with a simple hardware configuration has been highlighted and researched recently. Approaches with fewer sensors or only the camera are common because they are less expensive than other sensors, and visual input is usually more effective than auditory, gravity, and other sensors. The convolutional neural network (CNN) is used in the vision-based method to analyze vision and support AI agents in making decisions. However, since CNN is guided learning, it has a drawback in that the supervisor must first transmit case-specific training results. Consequently, an additional module called the Reinforcement Learning Algorithm (RL) is added to the learning agent, enabling it to become a general-purpose AI.

One of the most exciting reinforcements learning jobs with simple equipment and Xie from Oxford University [1] has produced the CNN network. In their thesis ''Towards obstacle avoidance based on monocular vision through deep-reinforcing learning', the AI agent is implemented on the Turtlebot Mobile, the 2-dimensional ground-based motor robot, using the only monocular camera above the hardware. The officer gradually finds ways to avoid obstacles and navigate the surrounding area as quickly as possible through the reinforcement learning algorithm. Their work's novelty lies in a two-tiered architecture where CNN comes first to extract information from the image and RL comes later to calculate the next movement, and the Double Q-network algorithm with Duel Architecture (D3QN), which improves RL performance.

This research focuses on the extension of 'Towards obstacle avoidance based on monocular vision through deep-reinforcing learning', sharing the D3QN algorithm as a nucleus, however, with a modified architecture to overcome more complex tasks. Given the power of RL, an AI agent must perform complex tasks other than simple two-dimensional navigation. Therefore, this work selects an uneasy air vehicle (UAV) or a drone as an AI agent ship to perform navigation and a 3-dimensional object following the object detection API with the CNN module. The integration of vision technology and drones has been a very popular research topic in recent years due to three-dimensional tasks such as delivering air parcels, scanning agricultural areas, etc. With an increasing focus on drone research, the industry has evolved rapidly, and there is plenty of user-friendly UAV software and hardware for research. The Microsoft Airsim API is used for this work because of its manoeuvrability and accessibility, and simulation environment.

Literature review

A brief overview of target recognition and reinforcement learning and the detailed principles of both are discussed in this portion.

Object Detection.

Object recognition has become a hot topic in recent years, with researchers collaborating mainly in the fields of computer vision

***Corresponding author:** Muhammad Shoaib, Department of IT and Emerging Sciences, University of Peshawar, Khyber Pakhtunkhwa, Pakistan, Tel: 03138841394; Email: shoaib1646@gmail.com

Received: April 16, 2021 **Accepted:** May 12, 2021 **Published:** May 19, 2021

and robotics, where sensory feedback dictates a robot's tasks. It was challenging to handle even limited picture sizes, such as 32 out of 32, a few years ago, because each pixel of the image may have up to 2555 types of states containing details regarding brightness and RGBA colors. Study with photos became even more available with the advent of deep neural networks, and research into the identification, recognition, and monitoring of artifacts started to evolve. Consequently, tech firms such as Google have often rendered the object detection module and researcher-friendly API available. Fast R-CNN [3], SSD [4], YOLO [5], and Retina-Net [6] are a few common object detection models.

Reinforcement Learning

In unsupervised learning, strengthening learning is a machine learning technique with force; however, supervised expertise is needed. The logic of the strategy is based on an activity-reward architecture, in which an agent acts first, then an evolving situation and a discovery calculate a new reward for the agent, which updates the preferred factors for each action. Each behaviour is driven by Markov's decision-making mechanism, in which stock experience has little bearing on recent decisions, implying that the agent is only concerned with observing the current state.

Q-learning

Reinforcement learning has some variations depending on how an agent chooses an action toward the observed environment and how the corresponding rewards update the agent's preferred factors. This framework uses the DRL (Deep Reinforcement Learning) architecture, which comes from the Q-learning algorithm. Q-learning stores use and updates the q values that determine an agent's actions where the action with the highest q value is chosen. Therefore, the q values must be in the form of an action-state pair, and the table is an appropriate frame for recording such values called the "Q table."

In summary, the agent takes action Ai ∈ A at state Si ∈S, which yields the best Q(Si, Ai), then the action affects the environment, and the new observation results in a reward as a result of updating the Q-table with a new q-value, Qnew(Si, Ai). The following Bellman equation is used to update the data Figure 1.

Deep Reinforcement Learning (Deep Q Learning)

Q-learning, on the other hand, works best when states are discrete. Because Q-table stores q-value in state-action pairs, infinitely many state counts increase Q-table size, likely exceeding the storage limit, resulting in waste of space. If we define visual input as a state, the state would be 256^3^(width of image * height of the image) because each pixel in the image has three channels and a brightness range of 0 to 256, requiring over a million buffers to save q-values. Previously, this was a significant barrier to reinforcement learning; however, with the addition of a deep neural network, the algorithm may evolve to deep reinforcement learning. Deep reinforcement learning was first introduced in 2013 by the company DeepMind with the well-known AlphaGo, which takes a game board image as an input and processes it through a convolution neural network, which is a type of deep neural network, and outputs which action yields the best q-values for each frame of the image [7]. The convolution network's original output shows which category has the highest similarity value to the image frame. The 'cat' category, for example, has the highest similarity value among the other types: 'dog,' 'rabbit,' and 'human.' Rather, the output of deep reinforcement learning indicates which actions are the best strategies at this frame of the image (Figure 2). We call this newly implemented convolution neural network a q-network in this state of the algorithm because it stores varia 2.2.3 Exploration and Exploitation bles that determine the qvalue for each action. As a result, the entire.

lgorithm is known as 'Deep Q Learning

The exploration and exploitation problem is a well-known dilemma problem in reinforcement learning. The algorithm forces an agent always to take the best action; however, the action may result in a local maximum rather than an overall maximum due to a lack of additional information or exploration. The q-network variables are initially set to random floats, and the learning process is carried out in an unsupervised manner, with the agent possibly repeating an action that provides an immediate short-term reward. The exploration strategy devises frequent random actions to avoid such learning limitations and provide more rich learning scenarios. This method employs the well-known epsilon-greedy exploration method, with the recently introduced convergent-based exploration method proving to be slightly more effective [8].

Epsilon-Greedy Exploration

How to balance time consumption between exploration and exploitation is the most important aspect of exploration and exploitation. Exploration contributes to overall convergence, so if an agent explores too much, the learning speed slows down; however, if exploration is discouraged, the learning may become stuck in local convergence. A parameter is introduced as a probability of choosing random action, exploration rather than exploitation, in the epsilon-greedy method (-greedy policy). In most cases, it is set to 0.1; thus, the agent chooses the best action in proportion to 0.9 and random action in proportion to 0.1. The parameter can change as the training steps progress, usually decreasing because exploration may not be required at a later stage of training.

Convergence-based Exploration

The epsilon-greedy policy problem is based on random factors that may cause an agent to fall into a local maximum at random,

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)}^{\text{temporal difference}}$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{\text{new value (temporal difference target)}}$$

Figure 1: Bellman Equation for updating q-value.

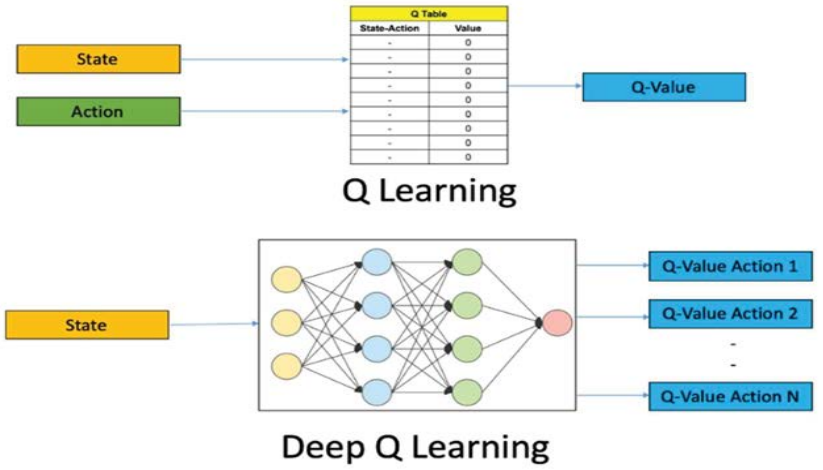

Figure 2: Q Learning and Deep Q Learning.

and the agent then repeats the action for short-term rewards. The best-case scenario is when the agent is confronted with an unfamiliar state and decides to investigate. As a result, a state-action familiarity or convergence level is examined in work [9] to choose between exploration and exploitation. Two parameters determine the exploration time threshold and minimum convergence error for exploitation, and. The agent is forced to explore during the first ten minutes of T's total training time, then exploit during the remaining ten minutes of T-. During exploration, the parameter checks if the action's convergence error is less than, indicating that the current state's action is sufficiently converged. If the algorithm determines that the action is converged, the algorithm will move on to the next random action until the new action is not sufficiently converged. As a result, the agent can deal more smoothly with new states and untried actions, referred to as faster learning.

Methodology

Introduction

The proposed model is an extension of work [1] on 2-dimensional UAV navigation. It is primarily divided into two sections: object detection and reinforcement learning. Object detection includes cognition and pre-processing, which alters visual data so that reinforcement learning modules can deal with it more easily. The process entails depth prediction, object detection, and the drawing of an object's bounding box, detailed in the following section. Pre-processed visual data is transferred to the input layer of the q-network in the reinforcement learning module, and the output layer determines which action to take. Another important module is the environment interpreter, which controls how reward and penalty are handled when interacting with the simulation environment, a UAV agent, and the logic module. In the section on reinforcement learning, the interpreter will be introduced in Figure 3.

Object Detection and Depth Prediction.

The entire architecture is designed to avoid collisions and force an agent to follow a target. As a result, depth prediction is required for collision avoidance, and object detection is required to obtain information about the target. The embedded depth prediction function of the Airsim API made depth prediction relatively simple. When it comes to real-world testing, it is a different storey. However, this article focuses solely on simulation training and testing, in which the depth prediction process is heavily reliant on camera and simulation setup related to the Airsim API. The Airsim API call 'DepthPerspective' returns camera input in black and white, with darker colours representing closer distance and brighter colours representing further distance. The q-learning network receives depth prediction images as state input, and the agent learns to prefer whiter space over black images; thus, collision avoidance is learned automatically. A study [1] has done the way of representing depth information as the state, and this research work has extended it.

When you consider the power of reinforcement learning, giving the learning agent a single task like collision avoidance may be underestimating its potential. As a result, this manuscript attempts to add a new task: detecting and following an object. To accomplish this, the object detection model must detect the person and process the data before feeding it into the learning algorithm. MobileNet SSD V3 was chosen for real-time object detection in this work because of its excellent performance in detecting speed despite its comparably low accuracy, as explained in the review session. The information about the position and size of the bounding box returns to the pre-process module once MobileNet detects the (Figure 4-6), which is a human.

As previously stated, the issue is how we define the state for the input layer of reinforcement learning, which was previously a depth-prediction image. To achieve a successful following, bounding box information must be reasonably harmonized with depth-prediction. It has been discovered that in depth-prediction based collision avoidance, an agent learns to prefer white space over dark space because brightness indicates open space while darkness indicates something approaching. This framework fills the bounding box with white colour based on the preference, so the agent is likely to learn to prefer bounding boxes.

Reinforcement Learning Architecture

Recent research has done much work on internal architecture reinforcement learning, and there are some well-known models like D3QN. This research framework aims to evaluate an existing model's

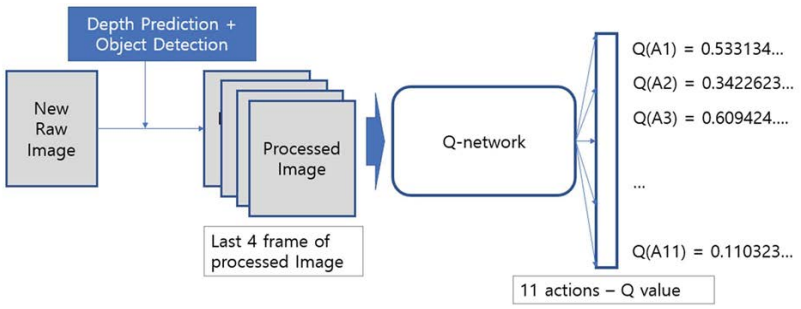

Figure 3: Architecture Setup of this work.

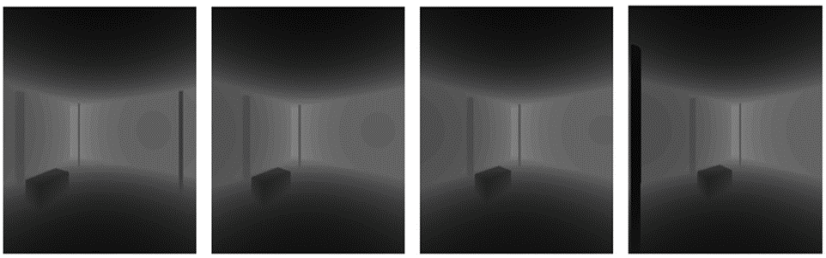

Figure 4: Depth Prediction Images.

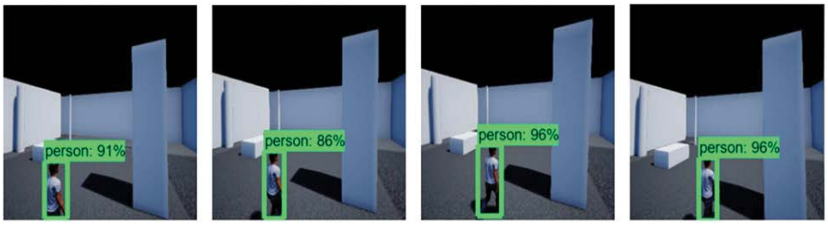

Figure 5: Images with Bounding Box.

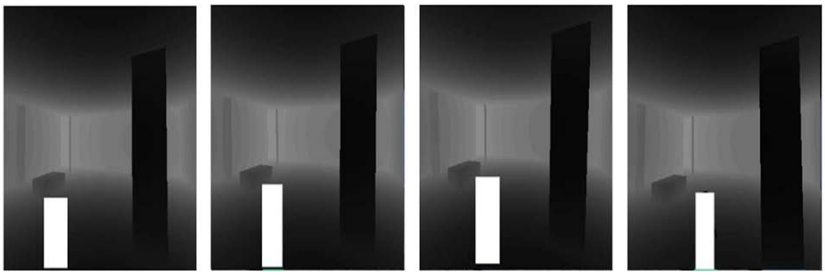

Figure 6: Images with Depth Prediction and Bounding Box.

potential, the D3QN network, using various reward functions and multi-functional actions. The work [1], which asks a ground robot to navigate in two dimensions while avoiding collisions, was successful, but it was limited to single, simple tasks. As a result, this research creates a three-dimensional movable UAV and tests it for three-dimensional navigation, collision avoidance, and object tracking. Furthermore, due to the increased complexity of 3D states, the convergence-based algorithm from work [10] has been implemented for smarter reinforcement learning exploration

Actions and Rewards

The next state is determined by an agent's action, the drone, from the first-person view camera's perspective. There are eight action options available, each of which allows the agent to change the speed or direction of progress in three dimensions. Previously, rewards were based on an agent's linear speed, with a minus ten penalty when collided with other objects [1]. The reward is calculated using the linear velocity and angular velocity by * cos ( ). The reward function aims to achieve the fastest possible navigation without colliding.

This research framework adds a new reward based on bounding box information to achieve a successful following human figure. When the bounding box is in the centre of the visual input, it gives the highest reward. The bounding box reward is defined as √2/ 2 – / where √2/ 2 represents the maximum distance from the image centre and / represents the distance between the image centre and the bounding box centre

Convergence-based Exploration

Previous work [10] implemented a convergence-based q-learning exploration that had nothing to do with deep q-learning. This paper applies the approach to a deep neural network with near-infinite states, whereas the previous paper [10] tested the algorithm with discrete states. The main distinction between the two is how convergence error is calculated, which is the q-value difference between current and future state action. Our model fits q-network to get current q-values instead of using a q-table to refer to q-values.

System Architecture

This section describes the hardware and software setup and the simulation environment used to validate the new method and conduct experiments.

Hardware and Software Setup

This proposed model uses simulation-based experiments, which typically take a day to complete, and it involves the continuous repetition of 3-dimensional rendering, physics calculations, and code execution for each frame. As a result, high-quality GPU and CPUs are required, and we used NVIDIA Titan RTX for GPU and Intel i7 9700k for CPU. We brought additional devices with NVIDIA graphic cards for simultaneous simulation testing, in addition to the main testing machine, because each trial takes a long time. Aside from the simulation requirement, the algorithm's code includes a neural network in which we used TensorFlow API for parallel tensor computation. Tensor Flow 1.14 is used, along with compatible CUDA 10.0 and cuDNN 7.4 libraries [11-13].

Environment Setup

Because the Airsim API runs on the platform, environments for simulation testing are built on Unreal Engine. The 'Simple' and 'Hallway' environments are the two primary environments. The simple setting is shaped like a rectangle with a wall in the middle. A human character is programmed to move around the space, avoid obstacles, and go up and downstairs. The hallway environment begins with a vertical hallway where the human character tends to walk straight and return to the start point after reaching the end.

Each environment is designed to put an AI agent through its paces in 3-dimensional navigation, following, and collision avoidance. The simple environment is more specialized for collision avoidance testing, whereas the Hallway environment focuses on testing human figure following. In a Simple environment, the number of possible routes is limited. As a result, distinguishing between normal navigation and the next target can be difficult [14]. In contrast, the agent in the Hallway environment is challenged to choose its route wisely in order to keep the target in sight. The senior design team from a computer vision and robotics lab assisted in the creation of a 3-dimensional environment [15,16]. With their help, the foundation for editing Unreal Engine was laid, and they programmed the human figure's path.

Experimental Results

This section shows the visual and numerical results of a training experiment in two different simulations. Before the test, it was assumed that a drone would smoothly follow a target and re-route its next move if it missed the target. When it comes to tracking a target, our algorithm's reward function rewards us more when the target is in the centre of the camera. As a result, the overall performance is regarded as the cumulative reward per episode [17-20]. Each episode is one cycle of starting with lifting and ending with a collision; as the agent gets better at following the target and survives longer without colliding, the cumulative reward increases. Furthermore, when compared to the adaptive epsilon-greedy algorithm, the novelty of convergence-based discovery must be demonstrated. (Figure 7-11) depicts the two lines of cumulative reward per episode, as well as two methods of exploration.

As shown in the graph, a convergence-based exploration experiment performs better in general, and an agent is more likely to adapt to the environment faster. On the other hand, the adaptive epsilon-greedy algorithm is likely to struggle due to its inability to cope with a three-dimensional environment. As previously stated, a 3-dimensional environment is more challenging than a 2-dimensional environment due to a large number of states. By calculating the mean of convergence error, the convergence-based exploration appears to have figured out a way to distinguish necessary states.

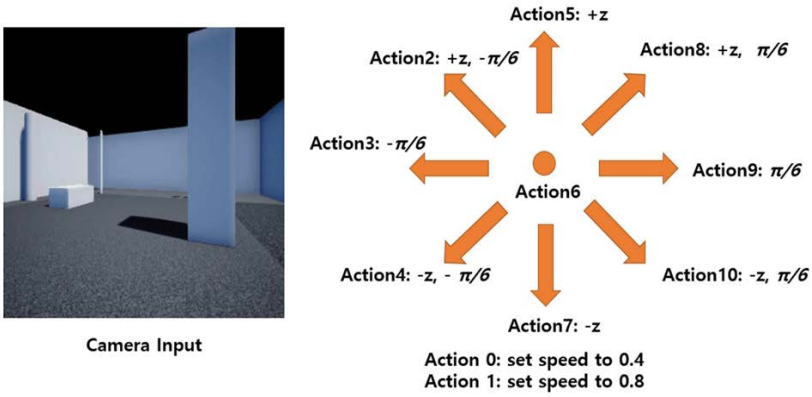

Figure 7: Input and Actions.

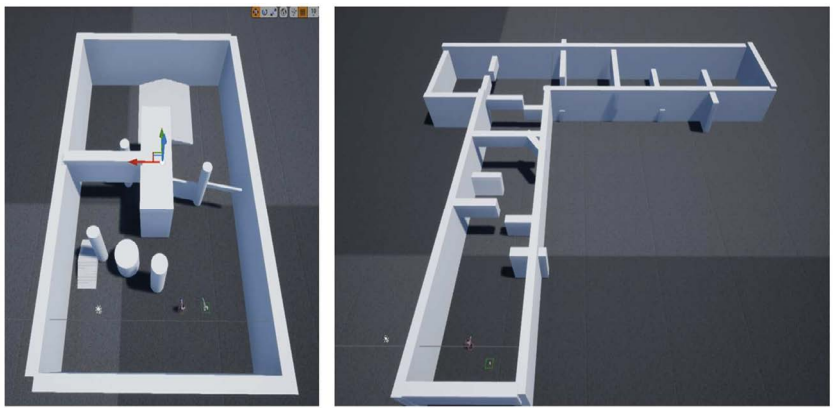

Figure 8: Simple (Left) and Hallway (Right) Environments for Training.

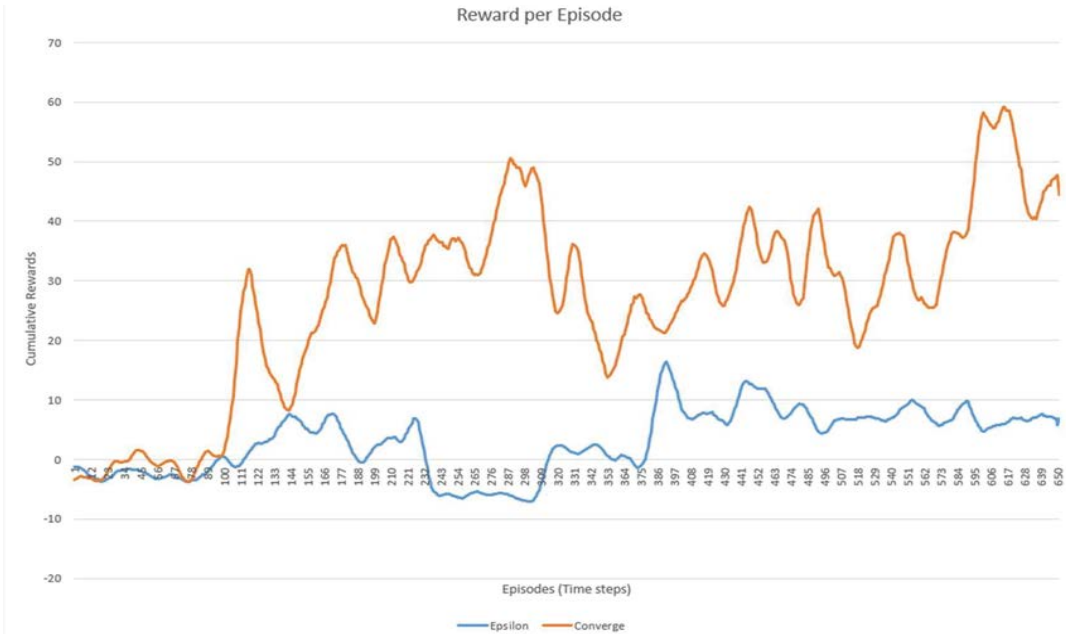

Figure 9: Comparing epsilon-greedy and convergence-based exploration in terms of cumulative rewards per episode.

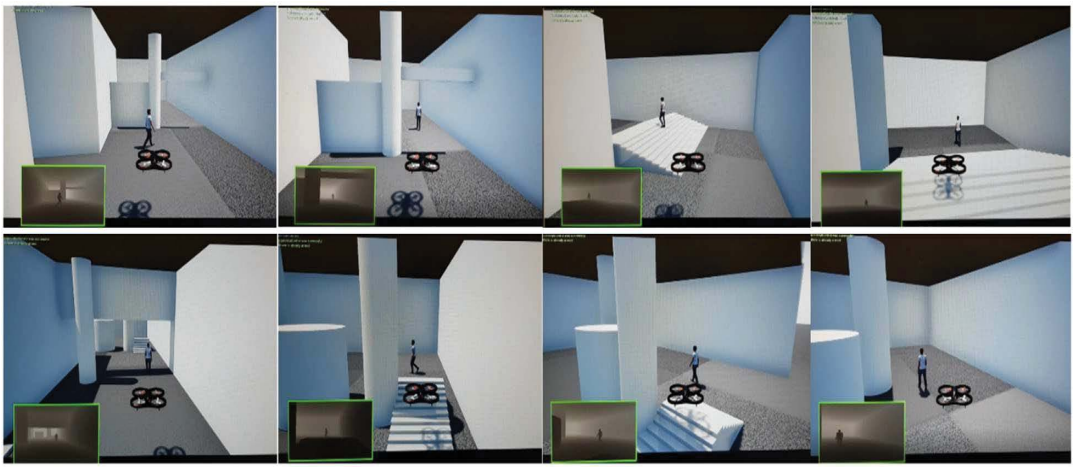

Figure 10: Simulation Screenshots In this video, a drone successfully follows a human in a simple environment.

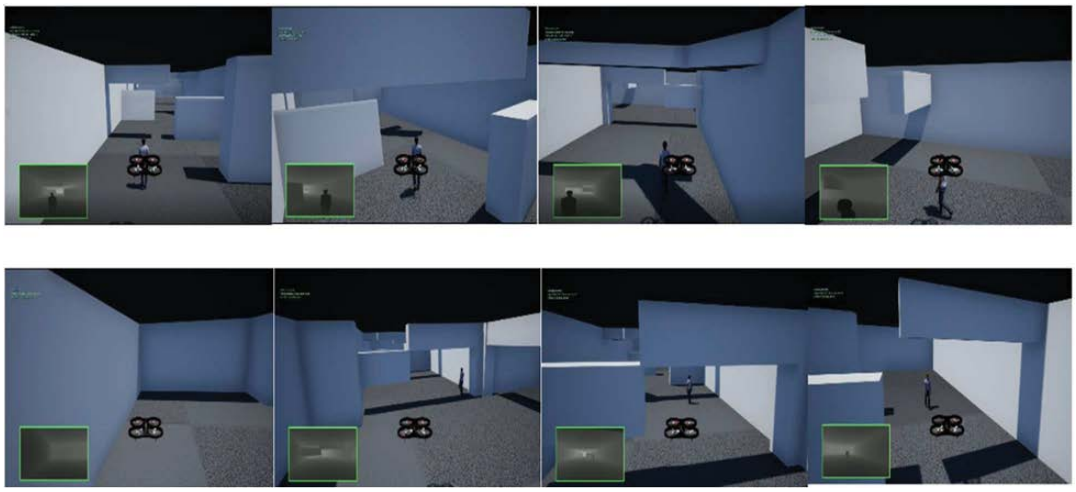

Figure 11: Simulation Screenshots In this video, a drone successfully follows a human in a hallway environment.

In terms of coping ability, the agent's performance is inconsistent. Because the reward function is based on the drone's speed when the target is not detected, when the target abruptly changes direction and disappears from the camera's view, the agent is likely to navigate as quickly as possible. The agent eventually re-detects the target and re-routes its trajectory, even though it may take some time. There is one drawback: when the target quickly turns back to face the drone, the agent tends to struggle and wander for longer. Because our algorithm prevents the drone from moving backwards, this is the case. Moving backwards in three dimensions would add nine additional actions, slowing training by more than twice.

Conclusion

Deep reinforcement learning is a powerful tool for training task-purpose robots, and unmanned aerial vehicles (UAVs) are one of the most rapidly growing robotics industries. In other words, the potential of using deep reinforcement learning to train UAVs is enormous, and it is worthwhile to try a variety of experiments with two. Collision avoidance, autonomous landing, and data collection with UAVs have all been studied in the hopes of improving performance. This manuscript is the first to use convergence-based exploration to test following a target without collision avoidance. Despite the difficulties of a three-dimensional environment, this paper can be said to have successfully trained UAVs to follow a target with better performance using convergence-based exploration.

Because this manuscript is limited to a simulation environment, future work on this project should be done with real-world UAVs. Because the Airsim API is compatible with Pixhawk/PX4, which runs the drones' real hardware, our proposed algorithm should be simple to implement in a real-world setting. A better exploration method adaptable to 3-dimensional space can improve internal architecture performance in addition to real-world testing. Convergence-based exploration was first tested in a two-dimensional environment with a limited number of states. It does an excellent job of reducing redundant exploration, but it is not designed for three-dimensional testing. As a result, we will compare several different exploration methods in future work and put them to the test in real-world scenarios.

References

1. Xie L, Wang S, Markham A, Trigoni N (2017) Towards Monocular Vision based Obstacle Avoidance through Deep Reinforcement Learning. RSS workshop on New Frontiers for Deep Learning in Robotics.
2. Polvara R, Patacchiola M, Sharm S, Wan J, Manning A, et al. (2018) Toward End-to-End Control for UAV Autonomous Landing via Deep Reinforcement Learning, International Conference on Unmanned Aircraft Systems.
3. Girshick R (2015) Fast R-CNN, IEEE International Conference on Computer Vision.
4. Liu W, (2016) SSD: Single shot multibox detector. European Conference on Computer Vision, 21-37.
5. Redmon J (2016) You only look once: Unified, real-time object detection. Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition.
6. Tsung-Yi Lin (2020) Focal Loss for Dense Object Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence.
7. Volodymyr M, Koray K, David S, Alex G, Ioannis A, et al. (2013) Playing Atari with Deep Reinforcement Learning. arXive e-prints.
8. Madawalagama, S (2016) Low Cost Aerial Mapping with Consumer Grade Drones. 37th Asian Conference on Remote Sensing.
9. Iro Laina (2016) Deeper Depth Prediction with Fully Convolutional Residual Networks, IEEE International Conference on 3D vision.
10. Masadeh A, Wang Z, Ahmed E, (2018) Convergence-Based Exploration Algorithm for Reinforcement Learning. Electrical and Computer Engineering Technical Reports and White Papers.
11. Ala'Eddin M, Zhengda W, Ahmed E (2018) Reinforcement Learning Exploration Algorithms for Energy Harvesting Communications Systems. IEEE.
12. Cameron F (2017) Autonomous Driving with a Simulation Trained Convolutional Neural Network. University of the Pacific, Thesis.
13. Hasselt H, Guez A, David S (2016) Deep Reinforcement Learning with Double Q-Learning. Proceeding of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI16).
14. Jemin H, Inkyu S, Roland S, Marco H (2017) Control of a Quadrotor with Reinforcement Learning. IEEE Robotics and Automation Letter.
15. Mnish (2015) Human-level control through deep reinforcement learning. Nature, 518.
16. Risto K, Nora E, Alejandro H, Victor M, (2018) Evaluation of Deep Reinforcement Learning Methods for Modular Robots. Workshop track – ICLR.

17. Sarmad R, Mujdat S (2015) Effects of UAV Mobility Patterns on Data Collection in Wireless Sensor Networks. IEEE.

18. Wulfe, Black. UAV Collision Avoidance Policy Optimization with Deep Reinforcement Learning, Stanford University.

19. Xiaodan L, Tairui W, Luona Y, Eric P (2018) CIRL: Controllable Imitative Reinforcement Learning for Vision-based Self-driving.

20. Ziyu Wang (2016) Dueling Network Architectures for Deep Reinforcement Learning. International Conference on Machine Learning.

Author Affiliation

Top

[1]*Department of IT and Emerging Sciences, University of Peshawar, Khyber Pakhtunkhwa, Pakistan*

[2]*Islamia College Peshawar, Peshawar, Khyber Pakhtunkhwa, Pakistan*