2020

Visual object tracking for UAVs using deep reinforcement learning

Kyungtae Ko
Iowa State University

Visual object tracking for UAVs using deep reinforcement learning

by

Kyungtae Ko

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Iowa State University

Ames, Iowa

2020

DEDICATION

I would like to dedicate this thesis to the world that I’m living in. I really thank many researchers in the field of computer science who indirectly helped me to continue my research and inspired me from the start of my childhood. Finally, of course my family, my friends and lab members who supported me internally deserve this dedication.

TABLE OF CONTENTS

Page

LIST OF FIGURES iv

ABBREVIATIONS v

ACKNOWLEDGMENTS vi

ABSTRACT vii

CHAPTER 1. INTRODUCTION 1

CHAPTER 2. BACKGROUND 4
2.1 Object Detection 4
2.3 Reinforcement Learning 4

CHAPTER 3. METHODOLOGY 9
3.1 Introduction 9
3.2 Object Detection and Depth Prediction 10
3.3 Reinforcement Learning Architecture 12
3.3.1 Actions and Rewards 12
3.3.3 Convergence-based Exploration 13

CHAPTER 4. SYSTEM ARCHITECTURE 14
4.1 Hardware and Software Setup 14
4.2 Environment Setup 14

CHAPTER 5. RESULTS 16

CHAPTER 6. CONCLUSION AND FUTURE WORK 19

REFERENCES 20

LIST OF FIGURES

Page

Figure 1 Bellman Equation for updating q-value. 5

Figure 2 Q Learning and Deep Q Learning. 6

Figure 3 Architecture Setup of this work........ 9

Figure 4 Depth Prediction Images. 10

Figure 5 Images with Bounding Box. 11

Figure 6 Images with Depth Prediction and Bounding Box. 11

Figure 7 Input and Actions. 12

Figure 8 Simple (Left) and Hallway (Right) Environments for Training........ 15

Figure 9 Comparison of cumulative rewards per episode between epsilon-greedy and convergence-based exploration. 16

Figure 10 Shots from Simulation Video showing a drone follows a human successfully in Simple Environment. 17

Figure 11 Shots from Simulation Video showing a drone follows a human successfully in Hallway Environment. 18

ABBREVIATIONS

UAV	Unmanned Aerial Vehicle
GPU	Graphic Processing Unit
DRL	Deep Reinforcement Learning
API	Application Program Interface
HCI	Human Computer Interaction
CNN	Convolutional Neural Network
R-CNN	Region Fully Convolutional Network
YOLO	You Only Look Once
SSD	Single Shot Multi-Box Detector
CNN	Convolutional Neural Network

ACKNOWLEDGMENTS

I would like to thank my committee chair, Dr. Jannesari Ali, and my committee members, Dr. Jin Tian and Dr. Radkowski Rafael, for their guidance and support throughout the course of this research.

In addition, I would also like to thank computer science department program manager Carla Harris and Lewis Nicole for their assist throughout the program of graduate study.

ABSTRACT

Integration of artificial intelligence and unmanned aerial vehicles (UAVs) has been an active research topic in recent years, especially when UAVs are required to implement difficult tasks which cannot be done easily with human control. Usually UAVs use multiple sensors such as top-down camera or LiDAR sensor to gather full information of environments, and main processor machine calculates all necessary trajectory for UAVs. However, such environment-specific methodology is not applicable to real-world problems mostly due to the complexity which incapacitates proper sensor installation around space. This thesis proposes an approach which uses monocular on-board camera and reinforcement learning-based model to follow the detected object. This approach is more cost efficient and environment-adaptive compare to the previous approaches which tend to use multiple sensors and pre-calculated trajectory. Our model extended the previous Deep Double Q-network with dueling architecture (D3QN) model, altered an action table and a reward function, which enables 3-dimensional movements, and combined object detection using MobileNet, which adds bounding box information to image input of training network. In addition, the convergence-based exploration and exploitation is adaptively applied to D3QN network aligned with epsilon-greedy algorithm. The tests are done in different simulation environments with different complexity and difficulty. The Microsoft-supported quadrotor simulating API, 'Airsim' module is used for the testing. Results shows the model tracks the detected object, the human figure, without colliding with obstacle along the path, and trained faster with convergence-based exploration algorithm.

CHAPTER 1. INTRODUCTION

With introduction of deep learning and performance boost of graphic processing unit (GPU), the field of artificial intelligence has gained growth power and actively been researched to test the potential of machine learning. Especially when it comes to Human Computer Interaction (HCI), where real application of deep learning takes on place, intelligent agents are likely to be implemented on moving hardware such as a ground robot as ground unit and a drone as air unit, and asked to perform autonomous tasks such as navigation[7], aerial mapping[10], object delivery etc. In typical, processes of achieving such goal are done based on integration of information gathering through multiple sensors and interpretation of given data such as path calculation, obstacle prediction and collision avoidance. These navigation and collision avoidance are quite challenging processes since the robot agent must deal with information of surrounding every frame second and calculate the optimal decision within that limited time. Furthermore, real world environment is usually flexible and complex which interrupts installation of multiple sensors all around the space for gathering and mapping surrounding information the agent.

Upon comprehension of realistic obstacles, complete installation of multiple utilities for information gathering and case-specific algorithm are rather absurd. Therefore, general-purpose algorithm which can cooperates with simple hardware setup has been highlighted and researched in recent years even though it’s challenging. Especially, approaches with less sensors or only with camera is popular since it is less expensive than other sensors and visual information is usually more powerful compare to auditory information, gravity information, and other etc. Vision-based approach tend to take advantage of Convolutional Neural Network (CNN), then analyze the vision, and help decision making of AI agents. However, only with CNN has a limitation since it is supervised learning where supervisor priorly must push case-specific training data. Therefore, Reinforcement Learning (RL) algorithm as an additional module is introduced which level up the learning agent to general-purpose AI.

One of the most interesting work of reinforcement learning with simple equipment and CNN network has done by Xie et al from University of Oxford (Xie et al, 2017). In their thesis 'Toward Monocular Vision based Obstacle Avoidance through Deep Reinforcement Learning', the A.I agent is implemented on mobile Turtlebot, the 2-dimensional ground motor robot, using only monocular camera on top of hardware the agent progressively figure out how to avoid obstacles and navigates a surrounding place as fast as it can by mean of reinforcement learning algorithm. The novelty of their work is on two level architecture where CNN comes first to extract information from image and RL comes later to calculate next move, and Double Q-network with Duel architecture (D3QN) algorithm which improves performance of RL.

This thesis is on the extension of 'Toward Monocular Vision based Obstacle Avoidance through Deep Reinforcement Learning' sharing D3QN algorithm as a core however with altered architecture to overcome more complex tasks and challenges. In consideration of powerfulness of RL, an AI agent should be able to achieve complex tasks other than just a simple 2-dimensional navigation. Therefore, this thesis chooses unmanned aerial vehicle (UAV), or drone as an AI agent vessel to perform a 3-dimensional navigation and object following using object detection API with CNN module. Integration of vision technique and UAV has been a very popular research topic in recent year due to functionality of 3-dimensional tasks such as aerial parcel delivery, farm area scanning, etc. With increasing attention to UAV research, the industry has rapidly evolved, and there exist many research-friendly UAV software and hardware. Microsoft Airsim API is used for this thesis due to handiness and accessibility as well as simulation environment provided by the same website.

In overview, the thesis consists of following: Chapter 2 introduces the review of literature which provides necessary knowledge background. Chapter 3 as a methodology section explains internal logic architecture of AI agents which contains CNN parts and RL parts. Overall flows between two modules will be described with examples as well as how knowledge from Chapter 2 is related. Chapter 4 is about utilities in terms of hardware and software, which is to implement Airsim API and simulation environment. Chapter 5 shows a result of AI agent's performance and corresponding analysis regarding average reward per episode.

CHAPTER 2. BACKGROUND

In this section, brief explanation on object detection and reinforcement learning is introduced as well as detailed principle of each if necessary.

2.1 Object Detection

Object detection has been an active research area mainly collaborated with computer vision and robotics where visual input determines tasks of a robot. Dealing with even a small size of image such as 32 by 32 as input was challenging a few years ago since every pixel of image can have 255^5 types of state which contains information of brightness and RGBA colors. As deep neural network introduced, working with image has become much easier and research regard to object detection, classification, and tracking started to boost. In consequences, researcher-friendly object detection module and API has been made in public often by IT companies such as Google. There are some popular object detection models which are Fast R-CNN[12], SSD[18], YOLO[19], and Retina-Net[20].

2.3 Reinforcement Learning

Reinforcement learning is a one of machine learning technique which it has strength in unsupervised learning however in some part supervised knowledge is required. The logic of the technique is action-reward based architecture where an agent takes actions first then changed environment and new observation calculate a new reward to the agent which updates preference factors for each action. Every action follows Markov decision process where action history doesn't affect the current decision making which means the agent only care about observation of current state.

2.3.1 Q-learning

Reinforcement learning has some variant depends on how an agent choose an action toward the observed environment, and how corresponding rewards update preference factors of the agent. This thesis uses Deep Reinforcement Learning (DRL) architecture which originates from Q-learning algorithm. Q-learning basically store, use, and update q-values which determine an agent's actions where the action with the highest q-value is chosen. Therefore q-values must be in form of action-state pair, and table is an appropriate frame for saving such values which is called 'Q-table'. In brief, at state $S_i \in S$, the agent takes action $A_i \in A$ which yield the best $Q(S_i, A_i)$, then the action affect the environment and the new observation gives a reward in consequence to update the Q-table with new q-value, $Q^{new}(S_i, A_i)$. The update is done by a Bellman equation as follow:

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

Figure 1 Bellman Equation for updating q-value.

2.3.2 Deep Reinforcement Learning (Deep Q Learning)

Q-learning, however, is effective only when states are discrete. Q-table stores q-value in state-action pair therefore infinitively many counts of state increase the size of Q-table, probably over the limit of storage which leads to waste of space. For example, if we define visual input as a state the number of state would be 256^3^(width of image * height of image) since every pixel of image has three channels and brightness between 0 and 256, which definitely demands over a million buffers for saving q-values. This has been a huge obstacle of reinforcement learning previously, however with introduction of deep neural network the algorithm could evolve to deep reinforcement learning. In 2013, the company DeepMind introduced the deep reinforcement learning with the well-known AlphaGo which takes game board image as an input, and process it into convolution neural network, which is a kind of deep neural network, and output decides for each frame of image which action yield the best q-values[14]. Originally output of convolution network indicates which category has the highest similarity value to a frame of image, for example with a cat image the ‘cat’ category has the highest similarity value among other categories such as ‘dog’, ‘rabbit’, and ‘human’. Instead in deep reinforcement learning, the output indicates which actions is the best strategy at this frame of image. In this state of algorithm, we call this new implemented convolution neural network as q-network since it stores variables that decides q-value for each action, therefore the entire algorithm is also called as ‘Deep Q Learning’.

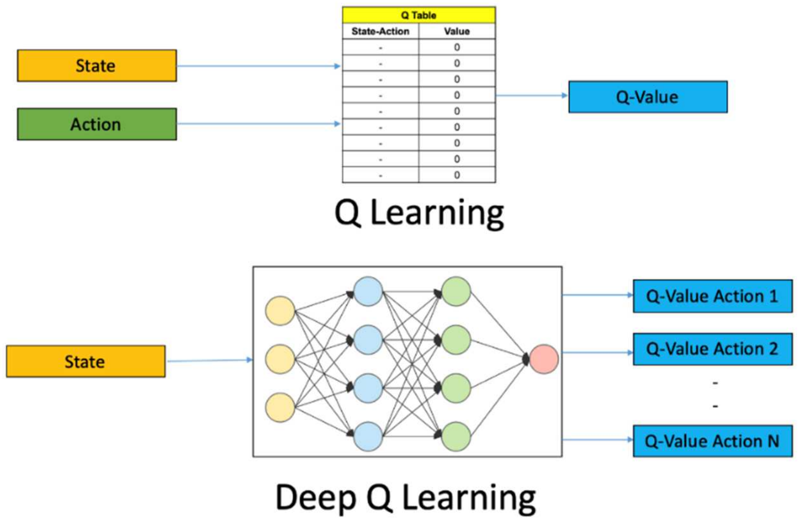

Figure 2 Q Learning and Deep Q Learning.

2.3.3 Exploration and Exploitation

There is famous dilemma problem in reinforcement learning which is exploration and exploitation problem. The algorithm forces an agent to always exploit the best action however there is a possibility of the action leads to local maximum not an overall maximum due to lack of additional information or exploration. Initially variables in q-network is set as random float and learning process is in unsupervised order where the agent possibly repeat an action that gives quick short-term reward. To prevent such limitation of learning and provide more rich learning scenarios, the strategy of exploration devises frequent random actions. This thesis chooses well-known epsilon-greedy exploration, and recently introduced convergent-based exploration proved to be work slightly better [1].

2.3.3.1 Epsilon-Greedy Exploration

Most important part of exploration and exploitation is how to balance time consumption between exploration and exploitation. If an agent explores too much, the learning speed decreases since exploitation contribute to overall convergence, however if exploration is discouraged the learning might be stuck in local convergence. In epsilon-greedy method (ε-greedy policy), a parameter ε is introduced as a probability of choosing random action, exploration rather than exploitation. In typical, ε is set to 0.1 therefore the agent chooses the best action in proportion of 0.9 and chooses random action in proportion of 0.1. Depends on circumstance, the parameter ε can change as training steps proceed usually in decreasing trend since exploration might not be needed at later stage of training.

2.3.4.2 Convergence-based Exploration

The issue with epsilon-greedy policy is that it relies on random factors which might randomly fall an agent into local maximum, and the agent repeat same action for short-term rewards. The best scenario would be when the agent faces an unfamiliar state it chooses to explore. Therefore, in work [5], a state-action familiarity, or a level of convergence is examined to decide between exploration and exploitation. Two parameter τ and ζ act as threshold of exploration time and minimum convergence error for exploitation. In total training time of T, the agent is forced to explore during first τ, then exploit in remaining time of T- τ. During exploration, the parameter ζ checks if convergence error of taking an action is less than ζ, which indicates the action for current state is converged enough. If the algorithm judges the action is converged, the other random action will be explored until the new action is not converged enough. Therefore, the agent can deal with unexplored states and untried actions more smoothly which is considered to achieve faster learning.

CHAPTER 3. METHODOLOGY

3.1 Introduction

The model of this thesis is basically the extension of 2-dimensional UAV navigation from the work [7]. It is mainly divided into two parts: object detection part and reinforcement learning part. The part of object detection processes cognition and pre-processing where visual data altered in a way that reinforcement learning module can deal with it easily. The process includes depth prediction, object detection, and drawing bounding box of the object which will be described in detail at following section. In reinforcement learning module, pre-processed visual data is transferred to input layer of q-network, and output layer decides which action to take. Another important module is the environment interpreter where an interaction between simulation environment, an UAV agent, and logic module is placed as well as how reward and penalty is treated. The interpreter will be introduced in the section of reinforcement learning.

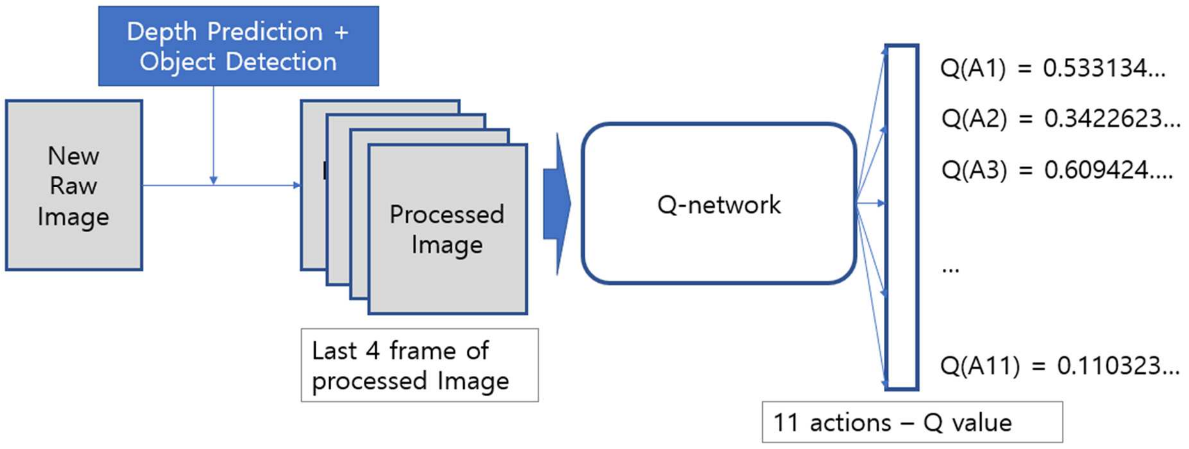

Figure 3 Architecture Setup of this work.

3.2 Object Detection and Depth Prediction

Purpose of entire architecture is to avoid collision and make an agent to follow a target. Therefore, depth prediction for the collision avoidance and object detection for getting information about the target is necessary. Depth prediction has been achieved comparably easily due to embedded depth prediction function of Airsim API. It would be the different problem when it comes to real world testing, however this thesis only focusses on simulation training and testing where the process of depth prediction heavily relies on camera and simulation setup related to Airsim API. The call 'DepthPerspective' from Airsim API returns camera input in black and white format where darker color represents close distance and brighter color represents far distance. Depth prediction images are transferred as state input of q-learning network and an agent learns to favor whiter space than black images therefore collision avoidance is automatically learned by the agent. The way of representing depth information as state has been done by study [7], which this thesis has extended it.

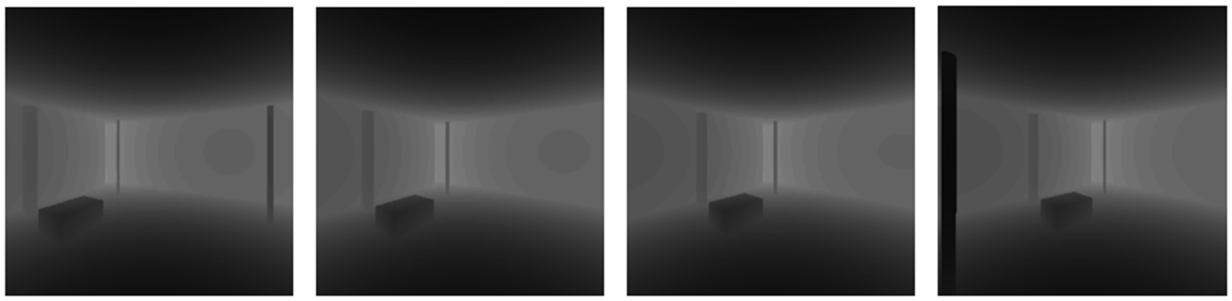

Figure 4 Depth Prediction Images.

Considering the power of reinforcement learning, asking a single task of collision avoidance to the learning agent might underrate the potential. Therefore, this thesis attempts add one more task which is detecting object and follow it. To achieve such goal, the object detection model needs to detect the person and process the information to input of learning algorithm. As explained in review session, MobileNet SSD V3 is selected for real time object detection in this thesis due to its fine performance in terms of detecting speed even though of its comparably low accuracy. Once MobileNet detects the figure, in this thesis human is the one, the information about position and size of bounding box returns to pre-process module.

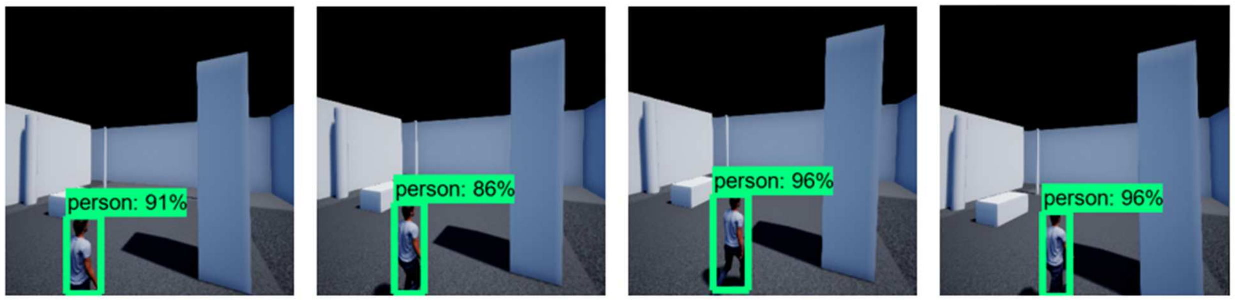

Figure 5 Images with Bounding Box.

As stated in above, the problem is how we define the state for the input layer of reinforcement learning, and it was depth-prediction image previously. To achieve successful following, information of bounding box should be fairly harmonized with depth-prediction. In depth-prediction based collision avoidance, it has been found that an agent learns to favor white space than dark space since brightness indicates open space while darkness indicates something approaching. Based on the preference, this thesis fill the bounding box with white color so it can be expected that the agent learns to favor bounding box.

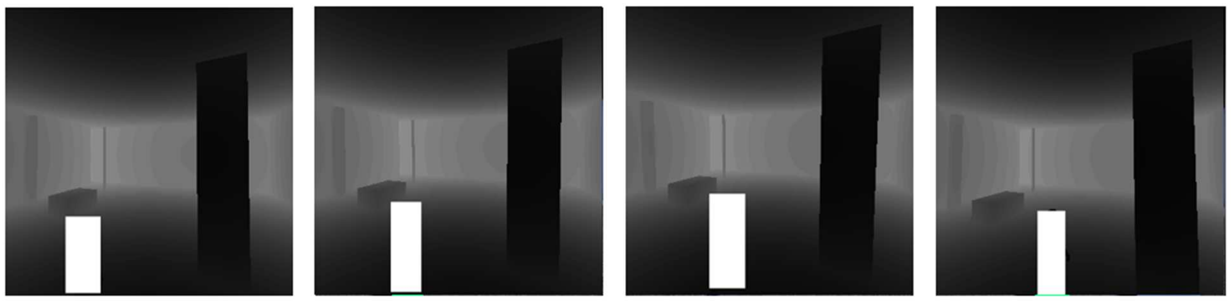

Figure 6 Images with Depth Prediction and Bounding Box.

3.3 Reinforcement Learning Architecture

Study of internal architecture reinforcement learning has been done numerously by recent research, and there exists some famous model such as D3QN. This thesis tries to test potential of an existing model, which is D3QN network, with different reward function and multi-functional actions equipped. The work [7], which asks a ground robot to perform 2-dimensional navigation with collision avoidance, was successful but limited to single and simple tasks. Therefore, this thesis set an agent as 3-dimensional movable UAV and test it to perform 3-dimensional navigation, collision avoidance, and following the object. Furthermore, due to increased complexity of 3-dimensional states, smarter exploration of reinforcement learning has been implemented, which is the convergence-based algorithm from the work [1].

3.3.1 Actions and Rewards

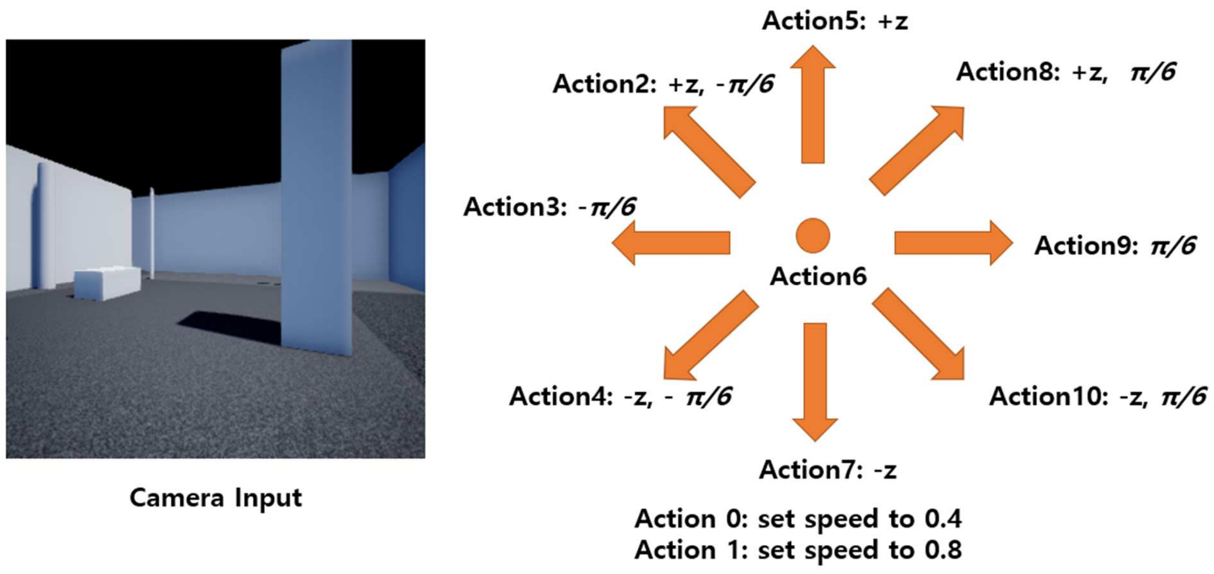

Figure 7 Input and Actions.

In perspective of the first-person view camera, the next state depends on an action taken by an agent, the drone. There are total eight options of action where the agent can increase or decrease the speed or change the direction of proceeding 3-dimensionally. Previously in work [], rewards were based on linear speed of an agent with the penalty of minus ten when it collides with other objects. With linear velocity v and angular velocity θ, reward r is calculated by $v * \cos(\theta)$. The aim of the reward function is to achieve as fast navigation as possible without any collision. This thesis adds a new reward regards to bounding box information to achieve successful following of human figure. It is to give a maximum reward when the bounding box is on the center of the visual input. The bounding box reward R_b is defined as $\sqrt{2}/\,2 - d/IMG_{size}$ where $\sqrt{2}/\,2$ is the maximum distance from the center of the image and d stands for distance between the center of image and the center of bounding box.

3.3.3 Convergence-based Exploration

Previous work [1] implemented a convergence-based exploration on q-learning, however it was not related to deep q-learning. This thesis applies the approach to deep neural network where states are close to infinite while the work [1] tested the algorithm with discrete states. The main difference between two are just how to calculate convergence error which is q-value difference between current state-action and future state-action. Instead of using q-table to refer q-values, our model fit q-network to get current q-values.

CHAPTER 4. SYSTEM ARCHITECTURE

This section provides information about hardware and software setup and simulation environment this thesis used and devised for experiments.

4.1 Hardware and Software Setup

This thesis operates simulation-based experiments which usually takes a day for one trial, and it is continuous repetition of 3-dimensional rendering, physics calculation, and running of code for every frame. Therefore, a fare quality of GPU and CPU must be equipped, and we used NVIDIA Titan RTX for GPU and intel i7 9700k for GPU. Except for the main testing machine, we brought additional machines that also equips NVIDIA graphic cards for simultaneous testing of simulation since each trial takes huge amount of time. Other than requirement for running simulation, the code for the algorithm contain neural network in which we used TensorFlow API for parallel computation of tensor. TensorFlow version of 1.14 as well as compatible CUDA 10.0 and cuDNN 7.4 library are used.

4.2 Environment Setup

Environments for testing in simulation are built on Unreal Engine since Airsim API runs on the platform. There are main two environments which we call 'Simple' and 'Hallway' environment. Simple environment is in shape of rectangle with a wall in the center. A human character is programmed to move around the space avoiding obstacles and passing through stairs.

Hallway environment starts with vertical hallway where human character tends to walk straight and come back to start point after it touches end of the hallway.

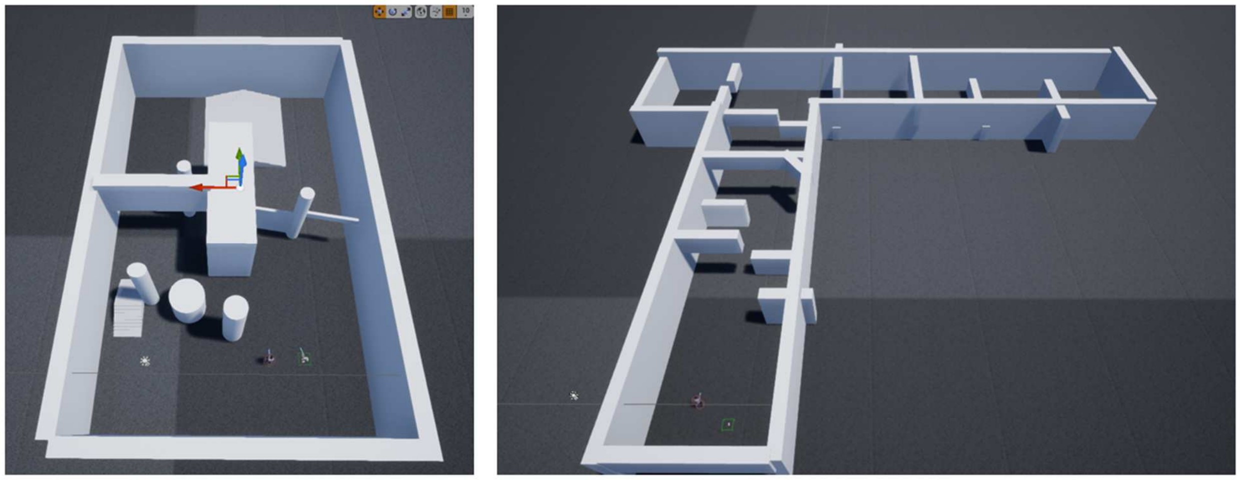

Figure 8 Simple (Left) and Hallway (Right) Environments for Training.

The purpose of each environment is to test various 3-dimensional navigation, following, and collision avoidance of an AI agent. Simple environment is more specialized for testing collision avoidance and Hallway environment is rather emphasized on testing following of human figure. All possible routes in Simple environment is limited therefore it is difficult to distinguish usual navigation and following target. In comparison, Hallway environment challenges the agent to choose its route wisely for not letting the target out of sight. The construction of 3-dimensional environment has been helped by senior design team from Dr.Ali’s lab. With their contribution, the base for editing Unreal Engine was prepared and they have programmed path of human figure.

CHAPTER 5. RESULTS

This section presents visual and quantitative results of experiment on training in two different simulations. In prior of experiment, it was expected that a drone follows a target smoothly and be able to re-route a next move when the drone misses the target. In terms of following a target, the reward function of our algorithm gives better reward when the target is at the center of camera. Therefore, the cumulative reward per episode, where each episode is one cycle of start with lifting and end with collision, is regarded to represent the performance of following in overall. The cumulative reward increases as the agent follows target better and survives longer without collision. Furthermore, novelty of convergence-based exploration must be shown by comparison with adaptive epsilon-greedy algorithm. Graph 5.1 shows the two lines of cumulative reward per episode with two exploration methods.

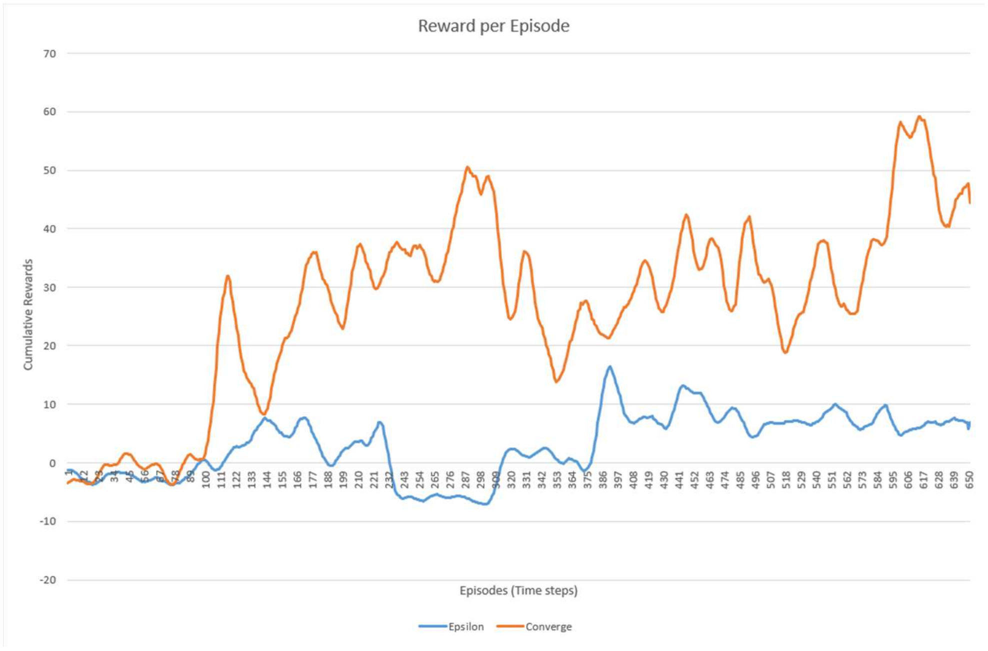

Figure 9 Comparison of cumulative rewards per episode between epsilon-greedy and convergence-based exploration.

As graph shows, an experiment with convergence-based exploration performs better in overall and an agent is likely to adjust to environment faster. In contrast, adaptive epsilon-greedy algorithm is likely to struggle being not able to cope with 3-dimensional environment. As explained earlier, 3-dimensional environment is challenging due to many amounts of states compare to 2-dimensional environment. The convergence-based exploration seems to figure out the way to distinguish necessary states by the mean of convergence error.

In terms of coping ability, the agent shows unstable performance. When the target suddenly changes its direction and get out of the sight of camera, the agent is likely to navigate as fast as possible since the reward function is based on the speed of the drone when the target is not detected. Even though it might take time, eventually the agent successfully re-detects the target and re-route its trajectory. There is a one limitation: when the target turns back quickly and head toward to the drone, the agent tend to struggle wandering longer. It is because our algorithm doesn't allow the drone to move backward. Moving backward in 3-dimensional would add nine more action which will slow down training probably more than twice.

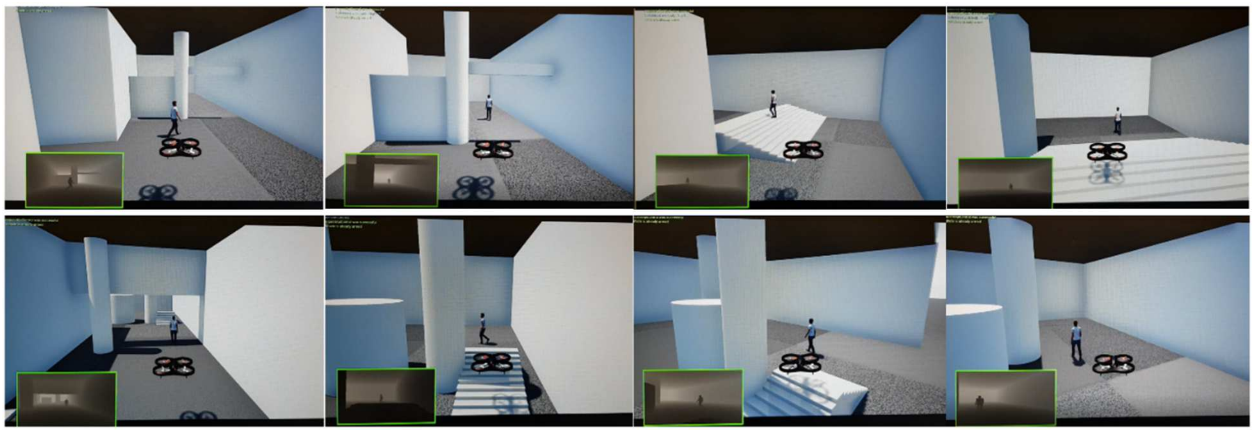

Figure 10 Shots from Simulation Video showing a drone follows a human successfully in Simple Environment.

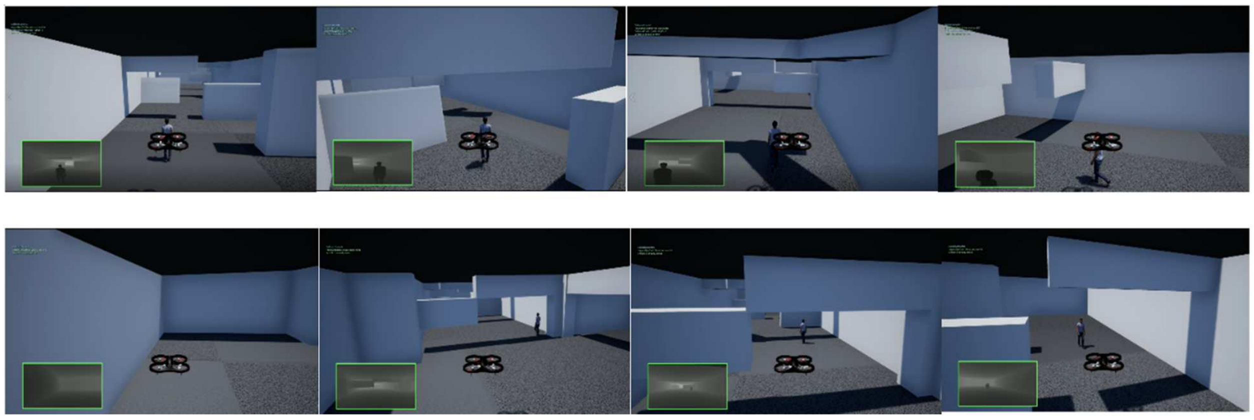

Figure 11 Shots from Simulation Video showing a drone follows a human successfully in Hallway Environment.

CHAPTER 6. CONCLUSION AND FUTURE WORK

Deep reinforcement learning is a power tool for training task-purpose robots and UAV is the one of rising robotics industry. In other words, the potential of training UAV with deep reinforcement learning is huge and it is worth trying various and different experiments with two. There has been researches which implements collision avoidance, autonomous landing, data gathering with UAV aiming for better performance. This thesis is yet the first paper which tests following a target without collision avoidance and uses convergence-based exploration. Even though there are challenges of 3-dimensional environment, it can be said that this thesis successfully trained UAV to follow a target with better performance with convergence-based exploration.

Future work of this work should be done with real world UAV since this thesis is limited to simulation environment. Airsim API is compatible with Pixhawk/PX4 which runs real hardware of drone therefore it is expected that in future work the algorithm of this thesis is easily implemented in real world environment. Other than real word testing, performance of internal architecture can be increase with better exploration method adaptive to 3-dimensional space. Convergence-based exploration is originally tested in 2-dimesional with discrete number of states. It performs well in terms of reducing redundant exploration however is not devised in regard of 3-dimesional testing. Therefore, we will try and compare several different exploration method and implement it in real world testing in future work.

REFERENCES

[1] Ala'Eddin Masadeh, Zhengda Wang, and Ahmed E. Kamal. *Convergence-Based Exploration Algorithm for Reinforcement Learning. Electrical and Computer Engineering Technical Reports and White Papers, 2018*

[2] Ala'Eddin Masadeh, Zhengda Wang, and Ahmed E. Kamal. *Reinforcement Learning Exploration Algorithms for Energy Harvesting Communications Systems. IEEE, 2018*

[3] Cameron Franke. *Autonomous Driving with a Simulation Trained Convolutional Neural Network. University of the Pacific, Thesis, 2017.*

[4] Hado van Hasselt, Arthur Guez, and David Silver. *Deep Reinforcement Learning with Double Q-Learning. Proceeding of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16), 2016.*

[5] Iro Laina et al. *Deeper Depth Prediction with Fully Convolutional Residual Networks, IEEE International Conference on 3D Vision, 2016.*

[6] Jemin Hwangbo, Inkyu Sa, Roland Siegwart, and Marco Hutter. *Control of a Quadrotor with Reinforcement Learning. IEEE Robotics and Automation Letter, 2017.*

[7] Linhai Xie, Sen Wang, Andrew Markham, and Niki Trigoni. *Towards Monocular Vision based Obstacle Avoidance through Deep Reinforcement Learning. RSS workshop on New Frontiers for Deep Learning in Robotics, 2017*

[8] Madawalagama, S. *Low Cost Aerial Mapping with Consumer Grade Drones. 37th Asian Conference on Remote Sensing,2016.*

[9] Mnish et al. *Human-level control through deep reinforcement learning. Nature, VOL 518, 2015*

[10] Riccardo Polvara et al. *Toward End-to-End Control for UAV Autonomous Landing via Deep Reinforcement Learning, International Conference on Unmanned Aircraft Systems, 2018*

[11] Risto Kojcev, Nora Etxezarreta, Alejandro Hernandez and Victor Mayoral. *Evaluation of Deep Reinforcement Learning Methods for Modular Robots. Workshop track – ICLR, 2018.*

[12] Ross Girshick. *Fast R-CNN, IEEE International Conference on Computer Vision, 2015.*

[13] Sarmad Rashed, and Mujdat Soyturk. *Effects of UAV Mobility Patterns on Data Collection in Wireless Sensor Networks. IEEE, 2015.*

[14] Volodymyr Mnih, Koray Kavukcugolu, David Silver, Alex Graves, Ioannis Antonoglou, Dann Wierstra, and Martin Riedmiller. *Playing Atari with Deep Reinforcement Learning. arXive e-prints, 2013.*

[15] Wulfe, Black. *UAV Collision Avoidance Policy Optimization with Deep Reinforcement Learning, Stanford University.*

[16] Xiaodan Liang, Tairui Wang, Luona Yang, and Eric P. Xing. *CIRL: Controllable Imitative Reinforcement Learning for Vision-based Self-driving, 2018*

[17] Ziyu Wang et al. *Dueling Network Architectures for Deep Reinforcement Learning. International Conference on Machine Learning, 2016.*

[18] Wei Liu. *SSD: Single shot multibox detector. European Conference on Computer Vision, 2016*

[19] Joseph Redmon. *You only look once: Unified, real-time object detection. Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition. 2016*

[20] Tsung-Yi Lin. *Focal Loss for Dense Object Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence. 2020*