Object Tracking Using Reinforcement Learning

DESIGN DOCUMENT

sdmay21-proj033
Swapp Lab / Iowa State University

Advisors
Deepak George Thomas
Dr. Ali Jannesari

Team Members/Roles
Karter Krueger (RL Team)
Joshua Kalyanapu (Sim Team)
Matthew Phipps (Sim Team)
Rithvik Menon (RL Team)
Ryan Howe (RL Team)
Thamir Al Harthy (Sim Team)
Zachary Mass (RL Team)

sdmay21-33@iastate.edu
https://sdmay21-33.sd.ece.iastate.edu/

Revised: October 4, 2020

Executive Summary

Development Standards & Practices Used

- Agile Development
- Test-Driven Development

Summary of Requirements

Functional requirements

Overall requirements

- The drone must be able to track intended object/person
- The drone must identify any obstacles to avoid
- The ability to control the camera during the object tracking

Drop requirements

- The drop must take off and land entirely autonomously
- Must be able to carry a microcontroller and GPU

Camera requirements

- Must be able to feed in the video frame sequentially
- The camera must capture everything during the flight

User Interface requirements

- The user should be able to easily find the power switch
- The user can accesses the camera data

Applicable Courses from Iowa State University Curriculum

- CS 575 (Computational Perception) - Computer Vision
- CS 518 (Computational Geometry) - Geometric Structures
- CS 577 (Applied Advanced Techniques for CS) - Coordinate Math
- CS 472 (Artificial Intelligence) - Decision Making

New Skills/Knowledge acquired that was not taught in courses

- Reinforcement Learning (Network Architectures, Reward Policy, etc)
- Unreal Engine (Simulation Environment)
- Airsim (Simulation Learning with Drone)

Table of Contents

List of figures/tables/symbols/definitions

1 Introduction

1.1 Acknowledgement

It is acknowledged that the SwAPP Lab, directed by advisor Dr. Ali Jannesari is the key source providing the GPU that the simulation is primarily training on, and the drone equipment that will be used for testing and executing the real-world code demo.

1.2 Problem and Project Statement

Our project seeks to build on and improve the results found in [4] and to develop an edge computing drone capable of AI application. That is, we seek to build a drone that uses reinforcement learning and computer vision to follow an object, e.g a human, while avoiding obstacles and staying in the air.

To solve this problem, we first create a simulation in order to cheaply test the reinforcement learning algorithm we have developed. Once the simulation testing provides the desired results, we move on to the creation of the drone and the loading of the reinforcement learning algorithm onto the drone in order to test this in the real world.

1.3 Operational Environment

This drone will be expected to follow an object, e.g a human, both indoors and outdoors. Outdoors, we will expect the drone to stay in the air and follow the appropriate object in the presence of wind and in most lighting conditions. We will not assume certain times of day nor weather affecting lighting conditions in the final product. However, this drone will not be required to follow an object in complete darkness.

1.4 Requirements

Functional requirements

Overall requirements

- The drone must be able to track intended object/person
- The drone must identify any obstacles to avoid
- The ability to control the camera during the object tracking

Drone requirements

- The drone must take off and land entirely autonomously
- Must be able to carry a microcontroller and GPU

Camera requirements

- Must be able to feed in the video frame sequentially
- The camera must capture everything during the flight

User Interface requirements

- The user should be able to easily find the power switch
- The user can accesses the camera data

1.5 Intended Users and Uses

With the nature of this being strictly a research project, we do not have a specific end user or use in mind. However, we know that the results of this project may be used in wildlife tracking. Overall, we expect that the primary users of this project will be other researchers who are seeking to build on our results.

1.6 Assumptions and Limitations

Assumptions:

- The real drone will fly in an open area outside where there aren't safety concerns.
- There will be a kill switch on the drone to take over if there is concern.
- The drone will fly in a day environment with proper lighting to see obstacles and persons, but not too bright to blind the camera.

Limitations:

- The drone cannot be flown indoors, so the weather must meet the right conditions.
- The simulation training takes a long time to train, so development can be slow.
- The outdoor environment must be similar to the simulation for easier transfer learning.
- The drone is limited to a short flight time outdoors due to the battery size.
- Computation power is limited by the Jetson TX2 and battery power, so networks must be small enough to run in real-time.

1.7 Expected End Product and Deliverables

Our end product shall be two separate entities: a drone capable of following an object in most environments, and a reinforcement learning algorithm that builds off of previous work.

The drone will be a physical drone that runs off of the Nvidia Jetson TX2 module, capable of following an object in most environments. This drone will be relatively unlikely to fail as the code running on it is based off of at least 1 year of testing in a simulation environment. This will be finished in late April 2021.

The reinforcement learning algorithm will be code written in Python that runs on the Nvidia Jetson TX2 module. This code is what controls the drone, keeps it in the air, and allows it to follow an object using computer vision. This code will be the culmination of over 1 year of effort and is expected to be finished in late April 2021.

2 Project Plan

2.1 Task Decomposition

The plan for this project is to train an unmanned aerial vehicle to follow a specified object. The goal is to use vision based deep reinforcement learning to implement the training process for the UAV. The UAV being used in development is in the form of a 4 propeller blade drone. Due to the nature of this type of drone we must implement the training in a virtual environment to ensure the safety of all involved including the drone. Our virtual environment will be implemented using Microsoft AirSim coupled Unreal Engine 4.

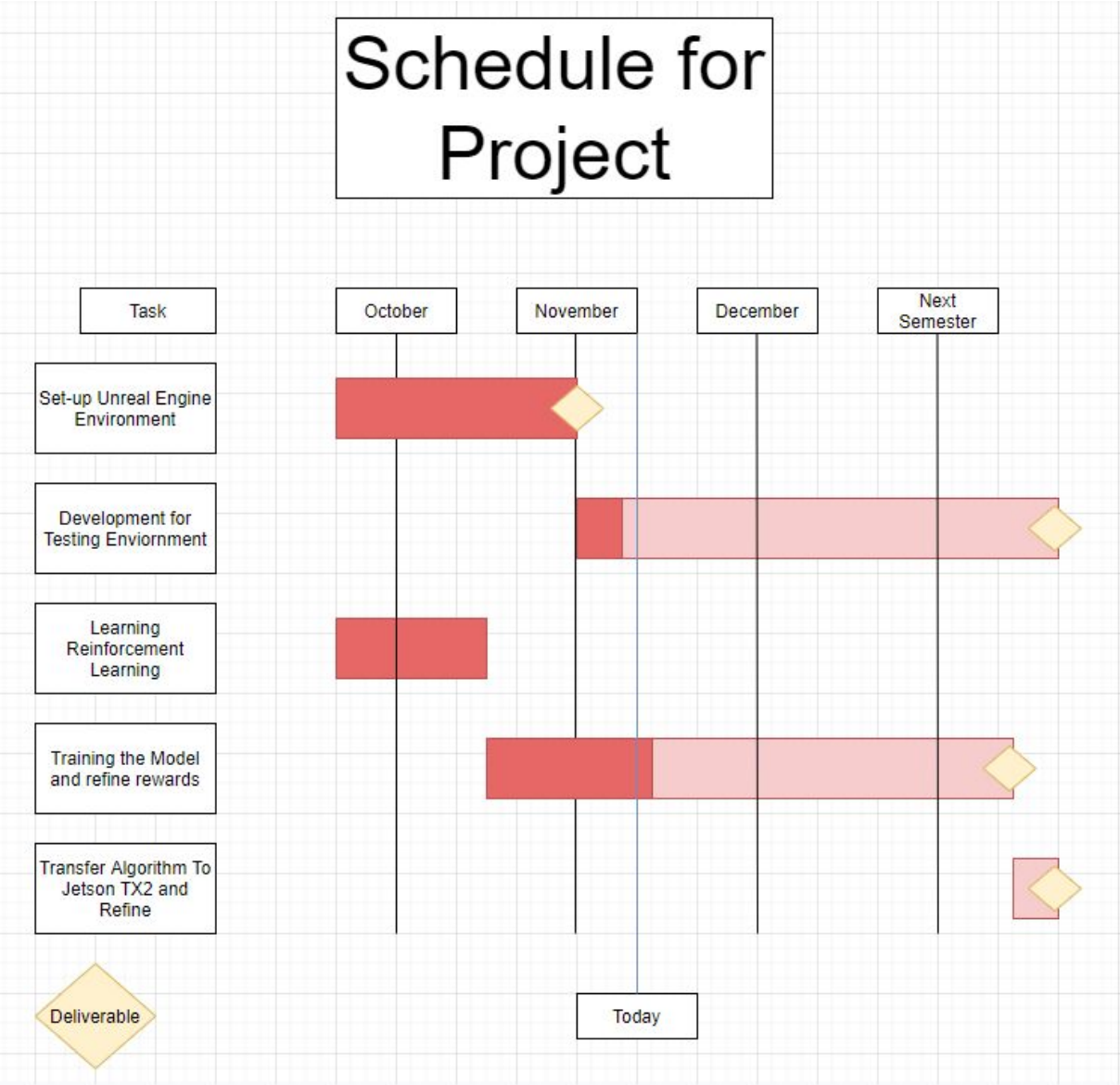
2.2 Risks And Risk Management/Mitigation

Current risks include drone collision, reward function not inadequacy, software bugs and software incompatibility. Drone collisions are expected during training, due to the nature of reinforcement learning, so training will always be done virtually in order to save money and effort. The probability of the reward function not being adequate is 0.5 and in this case a new or added reward function(s) will be researched and implemented. Since this projects virtualization software has been heavily tested, bugs and incompatibility have a low rating of 0.1

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

- Virtual environment setup and usable on all team members workstations
    - Clone AirSim
    - Clone Unreal Engine
    - Clone Project Repo
- Develop reward function to help drone learn efficiently
    - Research different reward functions for vision based RL
    - Implement new function(s)
    - Run and test until satisfactory results are obtained
    - Run in real world environment

2.4 Project Timeline/Schedule



Setting up Unreal Engine Environment	October 2020
Learning Reinforcement Learning Strategies	October 2020
Training the proposed algorithm on the environment and testing novel environments	November 2020 - May 2021
Transfer algorithm onto Jetson TX2	May 2021
Project Completion	May 2021

2.5 Project Tracking Procedures

Our team plans to use GitLab, Zoom and Group me to keep track and communicate during completion of the project. Git will be used as our version control system and as our wiki where we share findings and information. GitLab's issue and board feature will be used as a task manager. Zoom will be used for video-based conferencing with our project supervisor. Group me will be used for text communication among our team.

2.6 Personnel Effort Requirements

Task	Description	Estimated person-hours
Development of environment for testing	The simulation subteam of our team needs to bugtest our simulation environment	20
Deciding specific actions and rewards	The reinforcement learning subteam further modifies the already-implement learning algorithm looking for more optimalities	30
Training the proposed algorithm on the environment and testing novel environments	Running the python script on the simulation environment created in order to train the neural network. This will not require a significant amount of interaction, and will mostly be a computer executing the code hence the low amount of person-hours.	1
Transfer algorithm onto Jetson TX2	Transferring our reinforcement learning code onto the Jetson TX2. This will mostly involve porting our code to the platform and dealing with the peculiarities that arise.	30
Project Completion	Testing the drone in the real world and ensuring its effectiveness	10

2.7 Other Resource Requirements

This project additionally requires a VPS provisioned from ITS in order to run our simulation environment. Alternatively, a personal computer can be used if it has sufficient computing power, such as a GPU for training the neural networks with the simulation environment. Hard Drive space is also a concern at the full size of the project clocking in at about 130 Gigabytes.

2.8 Financial Requirements

The drone equipment and training GPU have been provided by our advisor's lab. There may be additional funds needed in the future to help acquire materials and designs for the Unreal Engine 4 environments.

3 Design

3.1 Previous Work And Literature

Our project makes use of work done by previous design students. This has had both benefits and shortcomings.

Benefits

- The technologies needed have already been decided upon
- A lot of the project has already been developed
    - The model is mostly done and just needs some troubleshooting / refining and more environments are needed to train future versions of the model

Shortcomings

- Uses deprecated libraries
    - Some of the libraries utilized in the project are using old versions of Unreal Engine which does not run on modern operating systems
- Need to invest time to understand the current codebase

The Project also makes use of some previous models that have had similar tasks that ours has. The main paper [4], uses an end-to-end network architecture to take in camera frames and output controls to the drone for tracking the person object. This is different from other approaches as it is all combined into a single network rather than several stages of control decision logic. The paper makes use of reinforcement-learning algorithms to avoid the tedious, expensive cost of labeling images manually for training. The key design they use in this paper is a ConvNet-LSTM. Our project will use a related approach to this architecture with customized networks and reward functions.

3.2 Design Thinking

Define Stage

This project was already started by our advisor's research lab and worked on by former year senior design groups, so there was not as much discussion for our defined stage since we inherited the previous semester's point-of-view. This point-of-view can be stated as "Design a model for a UAV that can follow a mobile object." To elaborate on this point-of-view the model design referred to is a Machine Learning model, the UAV can be any unmanned aerial vehicle, and the mobile object can be any object that can move. The UAV will be trained to follow the object in a safe manner through deep reinforcement learning.

Ideate Stage

Since our project is a hand-me-down from a previous design group we did not have say in what ideas we were to implement to solve the point-of-view statement. Ultimately though the previous design team decided upon deploying a deep learning model trained using reinforcement learning in a simulated environment to a UAV.

3.3 Proposed Design

The current design that the group is implementing is one that follows the following functional and non-functional requirements.

Functional Requirements

- Ability to follow another object
- Ability to fly
- Ability to carry a microcontroller and GPU

Non-functional Requirements

- Should be able to fly and avoid objects in the environment
- Model should be small enough to fit on our microcontroller

Model

Thus far our team has a model that was created using a network and reward function that were implemented with Google's Tensorflow library. The current architecture uses a D3QN network based on Deep-Q reinforcement learning. This uses two networks that are being run and trained simultaneously. There is also a convolutional neural network on top of this that uses a mobile-net SSDv3 detector to find the people in the frame and draw boxes around them. The location of the person is then used to reward the reinforcement learning networks to train it to keep a specified distance from the person in view while avoiding obstacles.

Training Environments

An environment is needed to train the model. For this our team is simulating environments using Unreal Engine paired with Microsoft Airsim to communicate with the model. The environments that our team designed are such that the model will encounter problems that one might expect it to encounter in the real world. This will help ensure the model is trained to properly operate outside of the simulated environments while also increasing the efficiency at which we can conduct training. Furthermore there is some degree of randomness in the simulation to help try to prevent overfitting.

3.4 Technology Considerations

There are a plethora of software for making and training machine learning models. Our team decided to use Tensorflow since it is one of the simplest to use machine learning frameworks available. Furthermore Tensorflow has very good documentation and an active community that can help ease the bug fixing process. Unreal Engine and Airsim were chosen as a pair because together they will make training and testing the model a very robust process. Since we can design environments for Unreal Engine we are not constrained like with it's alternatives, such as using a video game that involves flying. Airsim is also a great technology that gives our model access to an API that streamlines a lot of the problems that might arise from trying to develop our own software for a model to interface with a simulation environment.

3.5 Design Analysis

Thus far our team has run into many issues regarding design. Since we inherited this project from the senior design group of a previous semester there are a lot of issues that were left to be resolved along with parts of the project that have yet to be implemented.

Environment Issues

Since this project was inherited from a previous design group, they started developing using technology that was available to them at the time. However they did not update to newer versions of the software as it was released. This has caused trouble when trying to set up the simulation environments on our computers since we need to either update their code or revert our operating systems to earlier versions. The team is currently working on getting servers set up so that we can remote in to work on the environments since the team does not have the computers to do this ourselves.

Model Issues

An issue our team is currently working on resolving with respect to the model is trying to stop the UAV from flying too low to the ground or too high above the target object. Alongside this we are having trouble with the UAV colliding into obstacles, especially if it has trained for an extended period of time. The team believes that both these issues might be results of having an unrefined reward function and are trying to update it. Another possibility is the learning rate for training may be set at too high of a value since this would result in an overshooting of the minima of the cost function for our model.

3.6 Development Process

Our process will use agile methodologies and Test-Driven Development for our development process. We chose these both because they are quickly becoming industry standards and because of team-members familiarity with them. Test-Driven Development will be beneficial to us in ensuring the creation of high-quality code that works as expected in accordance with our functional requirements. Agile development practices will work well to organize the team's efforts around biweekly sprints in line with our biweekly reports. This will additionally facilitate stand up meetings twice a week, one with our project advisor and one without.

3.7 Design Plan

Our design plan is for a system where the drone is successfully able to fly around the environment and dodge obstacles to avoid collisions. Our architecture is best described by the illustration in Figures one and two in the appendix. Consider figure 1 below which illustrates a use-case diagram of how the user interacts with the drone through the automated system and through normal RC control.

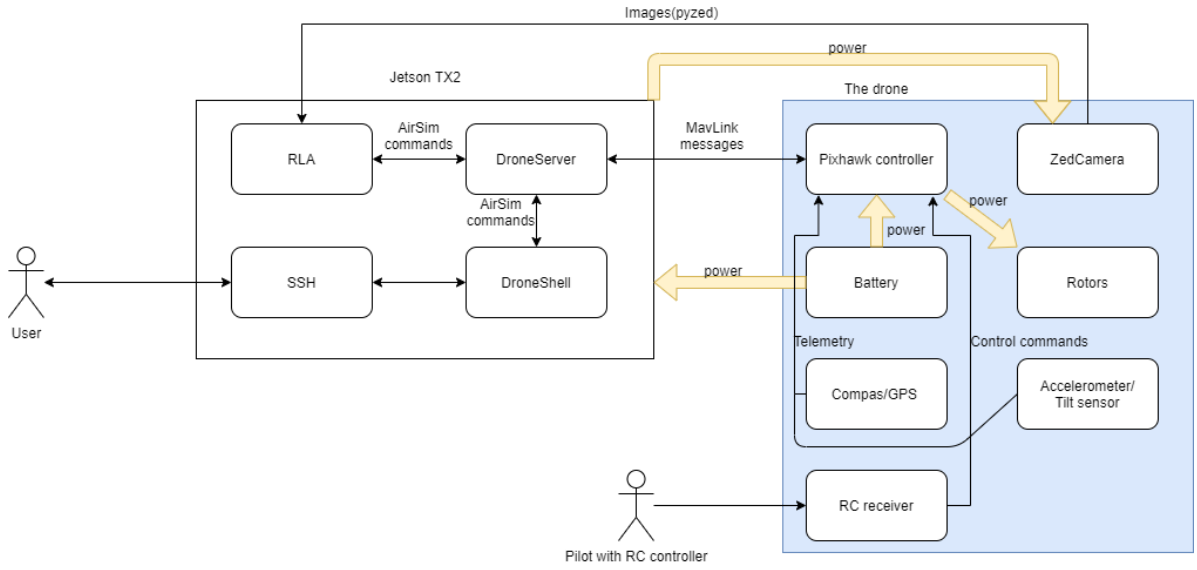


Figure 1. Diagram of the drone system

The pilot has access to an RC controller, which will send input to the RC receiver on the drone. This input will go to the Pixhawk controller which then sends a message to the droneserver application. This server application takes in commands from multiple locations, including another user using ssh to access the drone, the reinforcement learning algorithm output, and as previously mentioned the Pixhawk controller to send commands to the drone about movement. As shown in the diagram, the rotors on the drone are directly controlled by the PIxhawk controller. The part our team specifically is working on is the RLA, which takes in input from the ZedCamera mounted on the drone and makes decisions on where the drone should go accordingly. Once it makes the decision, it sends a signal to the DroneServer which then sends a signal to the Pixhawk controller which controls the rotors.

Figure 2, shown below, illustrates the architectural diagram of the project. It illustrates how the server handles the computation of images using the reinforcement learning model and turns them into action commands for the drone to execute. The drone itself faces the constraint of low processing power, so it instead uses remote computation in this design structure. This however faces its own challenges of potential latency between taking an image and executing the commands to "steer" the drone.

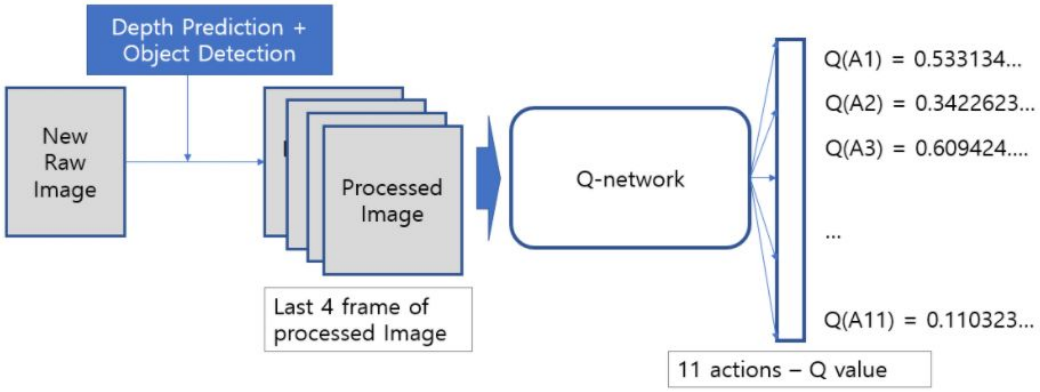


Figure 2. Diagram of the system architecture

4 Testing

4.1 Unit Testing

– The simulation environment can involve several aspects of testing independently of training the deep neural network models

– The camera and perception aspects of the sensors on the virtual drone can be tested to ensure the environment is working as it should

4.2 Interface Testing

The primary interface is between the simulation environment and the deep neural networks during the training phase. This interface includes communication in both directions for the reinforcement learning networks program to receive the sensory information such as the camera, depth sensor, and other drone speed items. Additionally, the program is sending commands the other direction to the simulator to tell the drone which way to move (left, right, forward, etc).

4.3 Acceptance Testing

The ability to receive sensor information and video frames from the environment will be a required test to demonstrate that the drone should also be able to operate in a real-world setting without as opposed to just in the simulation. The other direction of information with drone commands will also be an important test to ensure the program will be able to control the real drone.

4.4 Results

In the virtual training environment, the drone is focused on obstacle avoidance. So far, the drone is having problems properly avoiding the wall obstacles, furthermore it also sometimes crashes into the floor. As part of this iterative development process, we are revising the reward policy for the reinforcement learning and looking to add a better network for determining the altitude of the drone to avoid hitting the ground.

5 Implementation

Our current implementation is focused on continuing the development of the reinforcement learning architecture and its associated reward function as well as developing more environments for training the drone in. The additional environments will help the drone to generalize for better performance on future unseen environments and conditions. This will be necessary especially once the transfer learning happens next semester with going from the simulation to the real world.

Based on the target timeline, next semester will be primarily focused on continuing the work of this semester as well as performing transfer learning from the simulation to the real-world drone to fly in outdoor environments instead of a virtual simulation environment. The transition to the real world will require porting the networks over to run efficiently on a Jetson TX2 edge GPU computing device that will be on-board the drone. The physical drone makes use of the same style of stereo depth camera that is utilized in the simulation, so the model should port to the physical drone relatively seamlessly when it comes to perceiving the real world. The depth information will pass through the neural networks and the drone will perform the decisions of moving left, right, forward, et cetera, based on the output of the neural network. It is possible that we will then work to adapt it to track additional objects other than just people. These objects could include cars, other drones, and other moving objects. This additional tracking capability would need to be subjected to the same work-flow as the current model, that being trained in a virtual environment before being deployed to the TX2.

6 Closing Material

6.1 Conclusion

Currently the simulation team has completed setting up the server workstations for Unreal Engine and Airsim, while the reinforcement learning team has read through the current algorithm to further understand what is causing collision issues in the virtual environment. The reinforcement learning team is actively reading multiple research papers to learn more about reward function implementations to better solve this crashing issue. Our upcoming plans are split into two for the different sub teams. The simulation team will be working on creating more and a high variety of different environments to train the model on. An example of one environment is a “garden environment” that should be analogous to the greenery of the campanile grounds. The reinforcement learning team will be working on the reward function implementation and then work on the reinforcement learning algorithm to ensure the drone maintains a set distance for all obstacles and tracks the person from a safe distance.

6.2 References

[1]S. Bhagat and P. Sujit, "UAV Target Tracking in Urban Environments Using Deep Reinforcement Learning", *arXiv preprint*, 2020. Available: 2007.10934 [Accessed 4 October 2020].

[2]K. Ko, "Visual Object Tracking for UAVs Using Deep Reinforcement Learning", 2020. [Accessed 4 October 2020].

[3]K. Lee, B. Vlahov, J. Gibson, J. Rehg and E. Theodorou, "Approximate Inverse Reinforcement Learning From Vision-Based Imitation Learning", *arXiv preprint*, 2020. Available: 2004.08051 [Accessed 4 October 2020].

[4]W. Luo, P. Sun, F. Zhong, W. Liu, T. Zhang and Y. Wang, "End-to-End Active Object Tracking and Its Real-World Deployment via Reinforcement Learning", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 6, pp. 1317-1332, 2020. Available: 10.1109/tpami.2019.2899570.

[5]J. Michels, A. Saxena and A. Ng, "High Speed Obstacle Avoidance using Monocular Vision and Reinforcement Learning", *Proceedings of the 22nd International Conference on Machine Learning*, pp. 593-600, 2005. [Accessed 4 October 2020].

[6]L. Xie, S. Wang, A. Markham and N. Trigoni, "Towards Monocular Vision based Obstacle Avoidance through Deep Reinforcement Learning", *arXiv preprint*, 2017. Available: 1706.09829 [Accessed 4 October 2020].