Faculty of Electrical Technology and Engineering

DESIGN AND IMPLEMENTATION AN OBJECT-FOLLOWING SYSTEM BY USING DJI TELLO DRONE

LAM SHU XUAN

Bachelor of Electrical Engineering Technology (Industrial Automation & Robotics) with Honours

2023

DESIGN AND IMPLEMENTATION AN OBJECT-FOLLOWING SYSTEM BY USING DJI TELLO DRONE

LAM SHU XUAN

A project report submitted
in partial fulfillment of the requirements for the degree of
Bachelor of Electrical Engineering Technology (Industrial Automation & Robotics)
with Honours

Faculty of Electrical Technology and Engineering

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2023

UNIVERSITI TEKNIKAL MALAYSIA MELAKA
FAKULTI TEKNOLOGI DAN KEJURUTERAAN ELEKTRIK

BORANG PENGESAHAN STATUS LAPORAN
PROJEK SARJANA MUDA II

Tajuk Projek : Design and implementation an object-following system by using DJI Tello drone

Sesi Pengajian : 2023

Saya LAM SHU XUAN mengaku membenarkan laporan Projek Sarjana Muda ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Laporan adalah hakmilik Universiti Teknikal Malaysia Melaka.
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan laporan ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. Sila tandakan (✓):

☐	**SULIT***	(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)
☐	**TERHAD***	(Mengandungi maklumat terhad yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)
✓	**TIDAK TERHAD**	

Disahkan oleh:

[illegible]

(TANDATANGAN PENULIS)

Alamat Tetap:
8, JALAN ANGGEROIK10A,
TAMAN PUCHONG PERDANA,
47100 PUCHONG, SELANGOR.

[illegible]

(COP DAN TANDATANGAN PENYELIA)

Ts. Aminurrashid Bin Noordin
Pensyarah Kanan
Jabatan Teknologi Kejuruteraan Elektrik
Fakulti Teknologi dan Kejuruteraan Elektrik
Universiti Teknikal Malaysia Melaka

Tarikh: 12/1/2024

Tarikh: 19.1.2024

*CATATAN: Jika laporan ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh laporan ini perlu dikelaskan sebagai SULIT atau TERHAD.

DECLARATION

I declare that this project report entitled “Design and implementation an object-following system by using DJI Tello drone” is the result of my own research except as cited in the references. The project report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature	:	[illegible]
Student Name	:	LAM SHU XUAN
Date	:	12/1/2024

APPROVAL

I hereby declare that I have checked this project report and in my opinion, this project report is adequate in terms of scope and quality for the award of the degree of Bachelor of Electrical Engineering Technology (Industrial Automation & Robotics) with Honours.

Signature	:	[illegible]
Supervisor Name	:	Ts Aminurrashid Bin Noordin
Date	:	19/1/2024

DEDICATION

To my beloved mother, I would like to dedicate this project to my mother who has been encouraging me by giving endless support throughout the completion of this project.

192.2024

ABSTRACT

Unmanned aerial vehicles (UAVs) have gained significant popularity across various industries and applications due to their versatility and accessibility. To facilitate these applications, this project aims to develop a system capable of tracking and following a moving object using the DJI Tello drone. The DJI Tello drone is a compact quadcopter equipped with a built-in 5-megapixel camera capable of capturing 720p video at 30 frames per second. The project leverages computer vision techniques, specifically, Convolutional Neural Networks (CNNs), to locate the target in real-time and adjust the drone's flight path to maintain visibility. The system's core components involve creating and testing an algorithm that evaluates video data from the DJI Tello drone's camera and sends flying commands to its flight controller. The control mechanism is crucial to ensuring the drone maintains a safe distance from the object and avoids collisions with obstacles. Python programming language is utilized to control the drone via Wi-Fi, providing commands for take-off, landing, movement, rotation, and other flight maneuvers. The completed system will undergo rigorous testing using real-world scenarios, such as tracking a moving vehicle or object. By combining the capabilities of the DJI Tello drone, computer vision algorithms, and the control mechanism, the system aims to achieve real-time object tracking and following, contributing to enhanced disaster management, emergency response, and search and rescue operations.

ABSTRAK

Kenderaan udara tanpa pemandu (UAV) telah mendapat populariti yang ketara dalam pelbagai industri dan aplikasi kerana kepelbagaian dan kebolehcapaiannya. Untuk memudahkan aplikasi ini, projek ini bertujuan untuk membangunkan sistem yang mampu menjejak dan mengikuti objek bergerak menggunakan drone DJI Tello. Drone DJI Tello ialah quadcopter kompak yang dilengkapi dengan kamera 5 megapiksel terbina dalam yang mampu merakam video 720p pada 30 bingkai sesaat. Projek ini memanfaatkan teknik penglihatan komputer, khususnya, Convolutional Neural Networks (CNN), untuk mencari sasaran dalam masa nyata dan melaraskan laluan penerbangan dron untuk mengekalkan keterlihatan. Komponen teras sistem melibatkan mencipta dan menguji algoritma yang menilai data video daripada kamera drone DJI Tello dan menghantar arahan terbang kepada pengawal penerbangannya. Mekanisme kawalan adalah penting untuk memastikan dron mengekalkan jarak selamat dari objek dan mengelakkan perlanggaran dengan halangan. Bahasa pengaturcaraan Python digunakan untuk mengawal dron melalui Wi-Fi, menyediakan arahan untuk berlepas, mendarat, pergerakan, putaran dan gerakan penerbangan lain. Sistem yang lengkap akan menjalani ujian yang ketat menggunakan senario dunia sebenar, seperti menjejak kenderaan atau objek yang bergerak. Dengan menggabungkan keupayaan dron DJI Tello, algoritma penglihatan komputer dan mekanisme kawalan, sistem ini bertujuan untuk mencapai penjejakan dan pengikut objek masa nyata, menyumbang kepada pengurusan bencana yang dipertingkatkan, tindak balas kecemasan dan operasi mencari dan menyelamat.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my gratitude to my supervisor, Ts. Aminurrashid Bin Noordin for his precious guidance, words of wisdom and patient throughout this project.

I am also indebted to Universiti Teknikal Malaysia Melaka (UTeM) for the financial support which enables me to accomplish the project.

My highest appreciation goes to my parents, and family members for their love and prayer during the period of my study. An honourable mention also goes to my family members for all the motivation and understanding.

Finally, I would like to thank all of my classmates, as well as other individuals who are not listed here for being co-operative and helpful.

TABLE OF CONTENTS

		PAGE
DECLARATION		
APPROVAL		
DEDICATIONS		
ABSTRACT		**i**
ABSTRAK		**ii**
ACKNOWLEDGEMENTS		**iii**
TABLE OF CONTENTS		**iv**
LIST OF TABLES		**vi**
LIST OF FIGURES		**vii**
LIST OF APPENDICES		**ix**
CHAPTER 1	**INTRODUCTION**	**10**
1.1	Background	10
1.2	Problem Statement	11
1.3	Problem Objective	12
1.4	Scope of Project	12
CHAPTER 2	**LITERATURE REVIEW**	**14**
2.1	Introduction	14
2.2	OpenCV	14
2.3	Convolutional Neural Networks (CNNs)	15
2.4	Recurrent Neural Networks (RNNs)	18
2.5	Deep Reinforcement Learning Networks (DRLN)	20
2.6	Generative Adversarial Networks (GANs)	23
2.7	Variational Autoencoders (VAEs)	24
2.8	Comparison of article paper	26
2.9	Summary	27
CHAPTER 3	**METHODOLOGY**	**28**
3.1	Introduction	28
3.2	Methodology	28
3.3	First Milestone	29
3.4	Second Milestone	31
3.5	Project Design	32
3.6	Third Milestone	35
3.7	Fourth Milestone	35

CHAPTER 4 RESULTS AND DISCUSSIONS 38
4.1 Introduction 38
4.2 Proper detection range Between Person and DJI Tello Drone 39
4.3 Exploration of Light Intensity for Human Motion Detection 41
4.4 Automatic tracking of target motion 46
4.5 Summary 49

CHAPTER 5 CONCLUSION AND RECOMMENDATIONS 50
5.1 Potential for Commercialization 50
5.2 Future Works 51

REFERENCES 52

APPENDICES 54

LIST OF TABLES

TABLE	TITLE	PAGE
Table 2.1 Comparison of article paper		26
Table 4.1 Results of detection of person in difference distances		41
Table 4.2 Results of detection motion in different environment		45

LIST OF FIGURES

FIGURE	TITLE	PAGE
Figure 1.1 Block Diagram		11
Figure 2.1 a) image stiching; b) simultaneous recognition and segmentation; c) real-time face detection		15
Figure 2.2 Convolutional Neural Networks (CNNs)		16
Figure 2.3 Two-stage approach		16
Figure 2.4 Architecture of R-CNN [8]		17
Figure 2.5 Architecture of Fast R-CNN [8]		17
Figure 2.6 Architecture of Faster R-CNN		17
Figure 2.7 Architecture of Yolo [9]		18
Figure 2.8 Architecture of SSD [10]		18
Figure 2.9 Architecture of RetinaNet [11]		18
Figure 2.10 Recurrent Neural Networks (RNNs)		19
Figure 2.11 Deep Reinforcement Learning Networks (DRLN)		21
Figure 2.12 Generative Adversarial Networks (GAN) [22]		23
Figure 2.13 Variational Autoencoders (VAEs)		24
Figure 3.1 Methodology Flowchart		29
Figure 3.2 Literature review flowchart		31
Figure 3.3 Designing Flowchart		32
Figure 3.4 Codes import from the libraries		33
Figure 3.5 Codes to perform PID		33
Figure 3.6 Codes for control left, right, up and, down motion		34
Figure 3.7 Codes for control forward and backward motion		34
Figure 3.8 Object-tracking system flowchart		35

Figure 3.9 Object-following system flowchart 36

Figure 3.10 Concept of safety distance between drone and person 37

Figure 4.1 Vision from a) drone; b) reality 38

Figure 4.2 Vision from a) drone; b) reality 39

Figure 4.3 Experimental area 39

Figure 4.4 Person in the range of 10m 40

Figure 4.5 Person surpass 10m 40

Figure 4.6 Indoor light intensity a) lamp-off ; b) lamp-on 41

Figure 4.7 Person a) moving forward; b) moving backward; c) move left; d) move right 42

Figure 4.8 Person a) moving forward; b) moving backward; c) move left; d) move right 42

Figure 4.9 Light intensity in a) morning ; b) afternoon; c) evening 43

Figure 4.10 Person a) moving forward; b) moving backward; c) move left; d) move right 43

Figure 4.11 Person a) moving forward; b) moving backward; c) move left; d) move right 44

Figure 4.12 Person a) moving forward; b) moving backward; c) move left; d) move right 44

Figure 4.13 Curve motion path 46

Figure 4.14 Sampling frames A1-A16 captured from recording video 47

Figure 4.15 Target detection when a new person comes closer than first person 48

Figure 4.16 Drone action when no person is in the frame 49

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
Appendix A	Programming Codes	54

Type text here

CHAPTER 1

INTRODUCTION

1.1 Background

Unmanned aerial vehicles (UAVs) are aircraft that function without the presence of a human pilot. They can fly autonomously or remotely utilizing pre-programmed flight plans, inbuilt sensors, and navigation systems. UAVs have grown in popularity and are becoming more common in a variety of industries and fields due to their versatility, accessibility, and wide range of applications.

One of the applications being used is in disaster management and emergency response. Their ability to quickly reach remote or hazardous areas and provide real-time aerial views assists in assessing damage, identifying survivors, and coordinating relief efforts. Additionally, UAVs have proven valuable in search and rescue missions, surveying large areas in a short time, and locating missing persons.

In order to accomplish the application mentioned above, a system allowing to track and follow a moving object by using DJI Tello drone will be developed. The DJI Tello drone is a mini quadcopter that weighs 80 grams and has a built-in 5 megapixel camera that can capture 720p video at 30 frames per second. Another computer vision techniques such as Convolution Neural Networks (CNNs) will be applied to locate the target in real time and adjust the drone's flight path to keep it visible. To enable real-time object tracking, creating and testing an algorithm that evaluates video data from the DJI Tello drone's camera and sends flying commands to the drone's flight controller is a must. The drone can be controlled via Wi-Fi using python programming language that provides commands for takeoff, landing, movement, rotation etc. Besides, a control mechanism is crucial to ensure that the drone maintains a safe distance from the object and prevents collisions with obstacles. The completed system will be tested using a real-world scenario, such as tracking a moving vehicle or object. Last but not least, the overall block diagram shows the clear complete flows of the object-following system. Figure 1.1 shows the block diagram of object-following system.

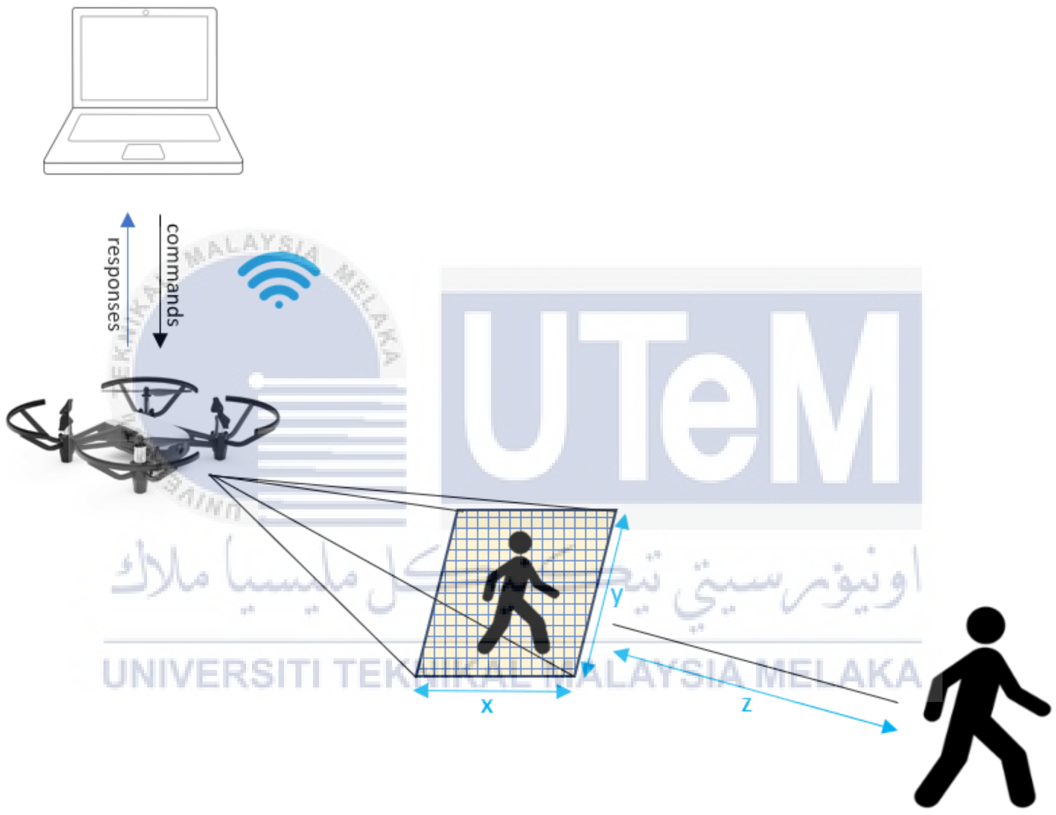


Figure 1.1 Block Diagram

1.2 Problem Statement

In recent years, drones have been applied in several fields due to drone versatility and accessibility. However, object-following systems can encounter a number of problems depending on the specific technology and circumstances in which they are used. Object-following systems that rely on visual cues can be sensitive to changes in lighting conditions.

For example, shadows or reflections can cause objects to appear differently, which can confuse the tracking algorithm. Besides that, in complex environments with many objects or where objects are moving in unpredictable ways, object tracking systems can struggle to distinguish the object of interest from other objects in the scene. Under these circumstances, an extra positioning technique is necessary, and since the majority of drones contain imaging cameras, positioning based on machine vision is one of the best options.

1.3 Problem Objective

The objectives of this project are as follows:

i. To design and implement an object following algorithm that processes video data from the DJI Tello drone's camera and sends flight commands to the drone's flight controller in real-time.
ii. To integrate a control system that ensures the drone maintains a safe distance from the object and avoids collisions with obstacles.
iii. To test the algorithm and control system in a real-world scenario and evaluate its performance in terms of accuracy, speed, and stability.

1.4 Scope of Project

By narrowing the needs for this project, a few guidelines are proposed to ensure that this project will achieve its objectives. The scopes covered for this project are:

i. Designing and developing an object-following algorithm that processes video data from the DJI Tello drone’s camera and sends flight commands to the drone’s flight controller.
ii. Implementing a control system that ensures the drone maintains a safe distance from the object and avoids obstacles.

iii. Integrate the algorithm and control system with the DJI Tello drone’s flight controller to enable real-time object following.
iv. Testing the system in a controlled environment, such as an indoor space with predefined paths, and in a real-world scenario.
v. Evaluating the system’s performance in terms of accuracy, speed, stability, and safety.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter reviews the relevant papers and journal articles. Previously, researchers at various institutions of higher education created comparable projects. The concepts and implementation of the earlier project's components, equipment, and programming language are covered here.

2.2 OpenCV

To understand OpenCV, let’s start by defining computer vision. The definition of computer vision is a branch of artificial intelligence (AI) that allows machines to make appropriate decisions based on the information they have learned from the given data. The data can be delivered to computers in the form of photos, videos, or any other visual input. Computer vision gives the computer the ability to analyze vision like human eyes. It makes it possible for them to be intelligent enough to recognize objects and distinguish between physical features [1].

OpenCV stands for Open-Source Computer Vision which is a library of programming functions mainly for real-time computer vision. It is the most widely used and well-documented computer vision library. Numerous computer vision algorithms are included in the open-source OpenCV library. OpenCV facilitates real-time applications and improves processing performance. One of the main objectives of OpenCV is to offer an open and user-friendly infrastructure for computer vision that enables anyone to create complex computer vision applications quickly. OpenCV is a powerful library and an effective tool for image processing and computer vision tasks. It is essential for real-time image processing and computer vision tasks in current applications. OpenCV makes use of NumPy, a highly optimized Python library for numerical computations. All OpenCV array structures are converted to and from NumPy arrays [2]. The applications for Open CV include stitching images shows in , face tracking, object detection and etc [3]. Figure 2.1 a) shows image stiching, while Figure2.1 b) shows simultaneous recognition and segmentation, and Figure 2.1 c) shows real-time face detection.

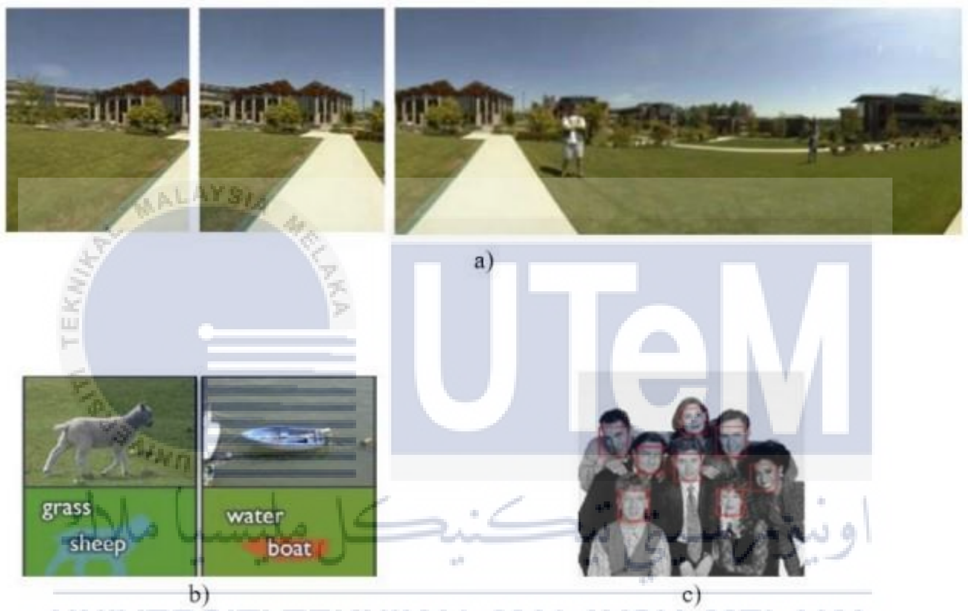


Figure 2.1 a) image stiching; b) simultaneous recognition and segmentation; c) real-time face detection

2.3 Convolutional Neural Networks (CNNs)

The foundation of Convolutional Neural Networks (CNNs) can be traced back to the discovery of Hubel and Wiesel in 1968 [4]. However, CNNs gained significant attention after the record-breaking performance of AlexNet in 2012 [5]. CNNs are a specific sort of multilayer neural network architecture built for spatial data. CNN architecture is inspired by real beings' visual perception, and they have gained popularity in domains [6]. Especially widely used in various computer vision tasks, such as image classification, object detection, and image segmentation. Figure 2.2 shows the image of Convolutional Neural Networks (CNNs).

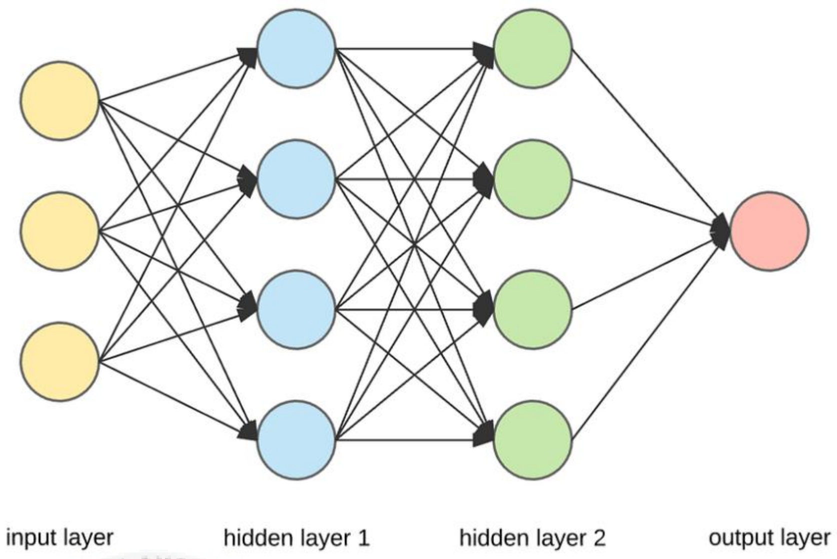


Figure 2.2 Convolutional Neural Networks (CNNs)

The study “A Review of Object Detection Models based on Convolutional Neural Network” is about object detection in computer vision, which involves identifying the class and location of objects within an image. The paper reviews different object detection models based on Convolutional Neural Networks (CNNs) [7]. These models are categorized into two different approaches:

Two-stage approach: This approach involves generating object proposals in the first stage and then classifying those proposals in the second stage. The models that follow this approach include R-CNN, Fast R-CNN, and Faster R-CNN. Figure 2.3 shows Two-stage approach, Figure 2.4 shows architecture of R-CNN, Figure 2.5 shows architecture of Fast R-CNN, and Figure 2.6 shows architecture of Faster R-CNN.

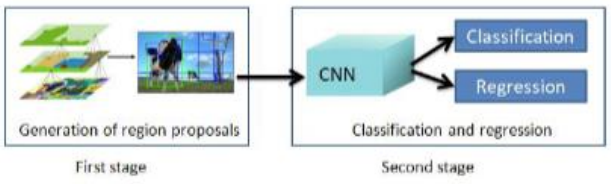


Figure 2.3 Two-stage approach

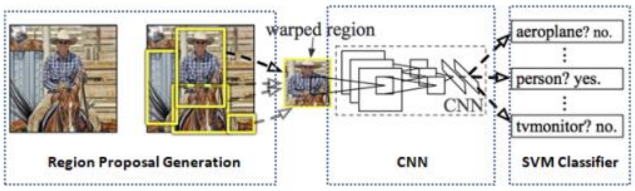


Figure 2.4 Architecture of R-CNN [8]

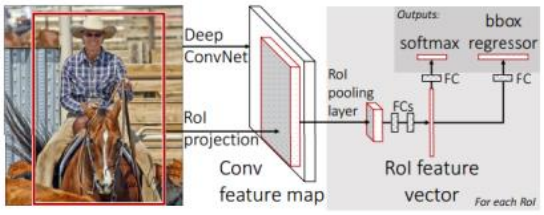


Figure 2.5 Architecture of Fast R-CNN [8]

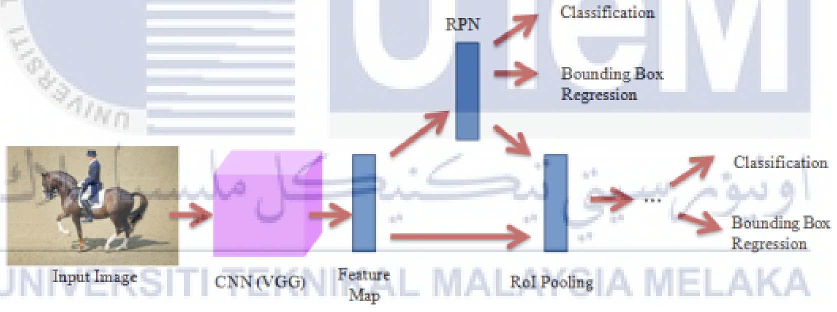


Figure 2.6 Architecture of Faster R-CNN

One-stage approach: This approach involves directly predicting the class and location of objects in a single stage. The models that follow this approach include YOLO, SSD, and RetinaNet. Figure 2.7 shows the architecture of Yolo, Figure 2.8 shows the architecture of SSD, and Figure 2.9 shows the arrchitecture of RetinaNet.

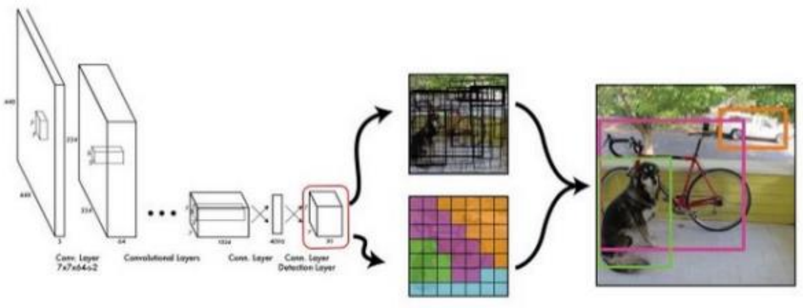


Figure 2.7 Architecture of Yolo [9]

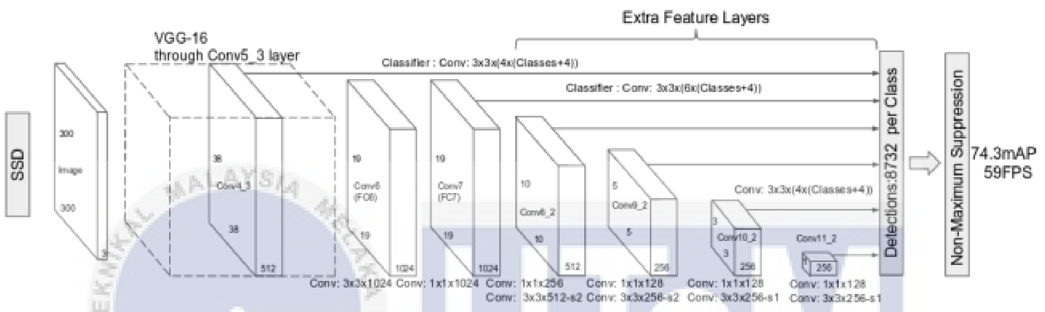

SSD
VGG-16 through Conv5_3 layer
Extra Feature Layers
Classifier : Conv: 3x3x(4x(Classes+4))
Classifier : Conv: 3x3x(6x(Classes+4))
Conv: 3x3x(4x(Classes+4))
Image
Conv4_3
Conv6 (FC6)
Conv7 (FC7)
Conv8_2
Conv9_2
Conv10_2
Conv11_2
Conv: 3x3x1024
Conv: 1x1x1024
Conv: 1x1x256 Conv: 3x3x512-s2
Conv: 1x1x128 Conv: 3x3x256-s2
Conv: 1x1x128 Conv: 3x3x256-s1
Conv: 1x1x128 Conv: 3x3x256-s1
Detections:8732 per Class
Non-Maximum Suppression
74.3mAP 59FPS

Figure 2.8 Architecture of SSD [10]

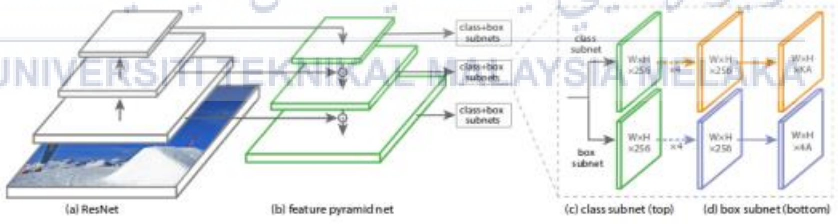


Figure 2.9 Architecture of RetinaNet [11]

2.4 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are primarily used for sequential data analysis, including natural language processing and speech recognition. RNNs have connections with feedback loops, allowing information to persist and influence future predictions. Figure 2.10 shows Recurrent Neural Network (RNNs).

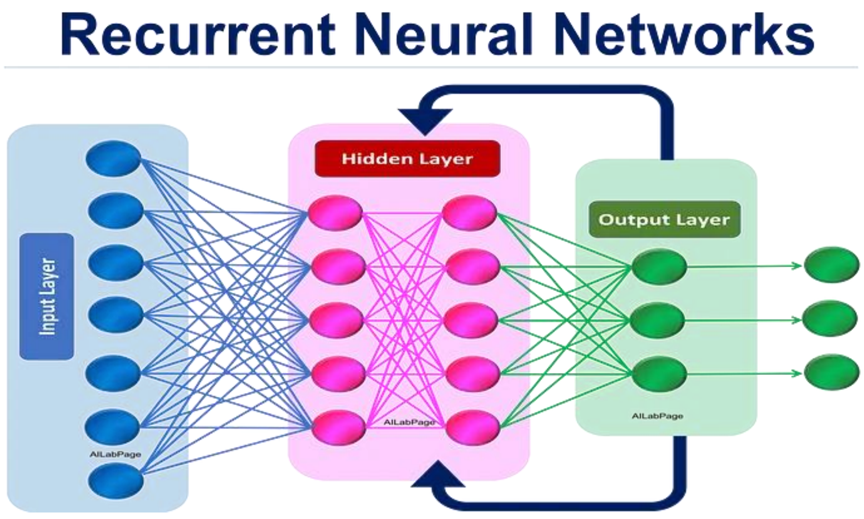


Figure 2.10 Recurrent Neural Networks (RNNs)

The fundamental principle behind applying RNNs is to improve their learning by repeating observations of a specific phenom or object, which is frequently coupled with a time-series collection. Long Short-Term Memory (LSTM) is a form of RNN that is now being used in a variety of applications.

The paper titled "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network" by Alex Sherstinsky [12] provides a comprehensive tutorial on the essential concepts of RNN and LSTM networks. The research uses Signal Processing ideas to explicitly construct the canonical RNN formulation from differential equations. It also presents and demonstrates a precise statement, from which the RNN unrolling approach is derived. The study examines the challenges of training the ordinary RNN and addresses them by changing the RNN into the "Vanilla LSTM" network using a series of logical arguments. The document includes all of the LSTM system's equations as well as extensive explanations of its constituent entities. It also discovers new ways to improve the LSTM system and incorporates these enhancements into the Vanilla LSTM network, resulting in the most broad LSTM version to date. The study is intended for readers who have experience with RNNs and LSTM networks and are open to a different pedagogical approach. It is also useful for Machine Learning practitioners who want to know how to deploy the new augmented LSTM model in software for experimentation and research.

Furthermore, in the field of remote sensing [13], RNN models have been used to deal with time series task analysis, with the goal of producing, for instance, land cover mapping. RNN models outperformed classical ML techniques in a pixel-based time series analysis aimed at discriminating classes of winter vegetation covering using SAR Sentinel-1 [14]. A recent method for accurate vegetation mapping [15] utilized multiscale CNN to extract spatial characteristics from UAV-RGB data, which was then input into an attention-based RNN to establish the sequential dependency between multitemporal features.

2.5 Deep Reinforcement Learning Networks (DRLN)

Deep Reinforcement Learning Networks (DRLN) have evolved as a powerful artificial intelligence solution that combines deep neural networks with reinforcement learning algorithms [16] . This fusion enables agents to learn difficult tasks by interacting with their environment and receiving feedback in the form of incentives or punishments. DRLN has attracted significant interest and achieved amazing success in a variety of fields, including robotics, gaming, and control systems, over the years. Figure 2.11 shows Deep Reinforcement Learning Networks (DRLN).

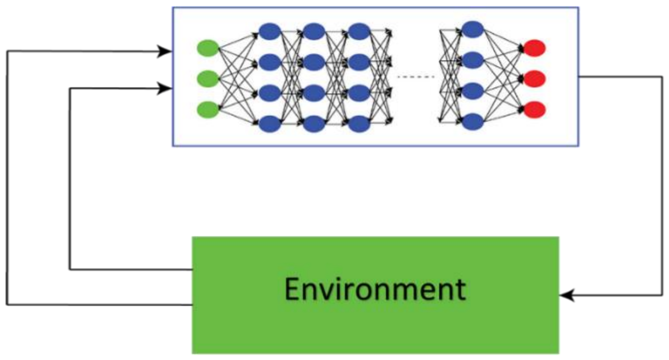


Figure 2.11 Deep Reinforcement Learning Networks (DRLN)

A research [17] by Patrik Reizinger and Marton Szemenyei introduces novel techniques for curiosity-driven exploration within the framework of Deep Reinforcement Learning. These methods leverage the attention mechanism to incentivize exploration and enhance generalization. The proposed approaches, namely AttA2C and RCM, are empirically evaluated on Atari games from OpenAI Gym, demonstrating encouraging outcomes. The authors summarize that incorporating attention-based curiosity-driven exploration can be highly effective for training agents in scenarios with limited rewards. Furthermore, this approach has the potential to improve overall performance and generalization capabilities of the agents.

The study in [18] introduces a novel deep reinforcement learning algorithm called Soft Actor-Critic (SAC) designed for continuous state and action spaces. SAC is built upon the maximum entropy reinforcement learning framework, where the actor's objective is to maximize both the expected reward and the entropy. By combining off-policy updates with a stable stochastic actor-critic formulation, SAC achieves impressive performance on various continuous control benchmark tasks, surpassing previous on-policy and off-policy approaches. The paper also investigates the significance of specific SAC components and examines the algorithm's sensitivity to hyperparameters such as reward scaling and target value update smoothing constant.

Moreover, DRLN has been successfully applied to complex tasks with high-dimensional state spaces. The Asynchronous Advantage Actor-Critic (A3C) algorithm, introduced by Mnih et al. [19] , utilized multiple agents that asynchronously interacted with separate instances of the environment, sharing the learned information periodically. This approach demonstrated excellent scalability and accelerated learning in challenging environments.

Another significant advancement in DRLN is the introduction of model-based approaches. Model-based methods leverage the use of learned models of the environment to plan and make more informed decisions. These methods aim to address the sample inefficiency problem often encountered in model-free approaches. For instance, the Model Predictive Control (MPC) algorithm in reference [20] combines a learned dynamics model with an optimization algorithm to plan actions that optimize long-term rewards. By incorporating learned models, these approaches have shown promise in achieving faster learning and improved data efficiency.

Furthermore, the combination of DRLN and deep generative models has sparked interest in domains such as unsupervised learning and exploration. DRLN has been related to Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) to enable generative modeling, allowing agents to learn complex representations and produce new examples. By learning diverse and representative state spaces, these techniques have the potential to address the difficulty of exploration in reinforcement learning.

Deep Reinforcement Learning Networks (DRLN) have revolutionised artificial intelligence by integrating deep neural networks with reinforcement learning techniques. DRLN has gained great success in a variety of sectors thanks to developments such as DQN, policy gradient methods, model-based approaches, and the integration of generative models.

These breakthroughs continue to fuel research and show great promise for solving complicated real-world challenges.

2.6 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) first introduced by Goodfellow I [21]. GAN is made up of two parts: a generative network that generates samples and a discriminator that identifies the source of the samples. When new samples created by the generative network and real-world samples are put into the discriminator, the discriminator will accurately distinguish between the two types of samples. They are commonly utilised in the production of images, videos, and voices. Figure 2.12 shows Generative Adversial Networks.

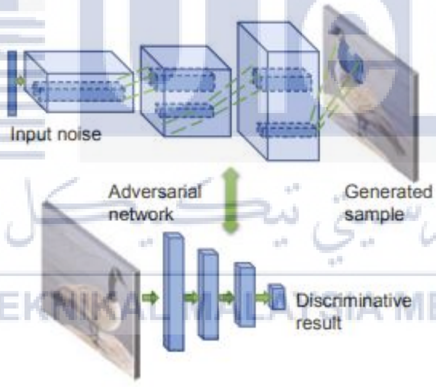


Figure 2.12 Generative Adversarial Networks (GAN) [22]

Since its first introduction, GAN has been extended into various applications. For instance, hybrid-augmented intelligence which is a new type of AI that integrates human cognitive capabilities or human-like cognitive models with machine learning methods [22].

Furthermore, Mirza et al. [23] propose the Conditional GAN which can be used to direct the data generation process by conditioning the model on additional information such as class labels or data from other modalities. Radford et al. [24] proposed a class of GAN results in stable training across a range of datasets and allow for training higher resolution and deeper generative models. Recently GAN has also been extended for generating images based on word descriptions [25] producing the aesthetic and architecture of natural indoor scene photos [26] , converting an image from one site to another [27] , and transform thermal face images into visible faces [28][29].

2.7 Variational Autoencoders (VAEs)

Variational Autoencoders (VAEs) have emerged as a prominent class of generative models in the field of deep learning. VAEs combine the power of neural networks with probabilistic modeling, enabling the generation of new data samples from learned latent representations. VAEs are frequently employed for generative modeling projects. Figure 2.13 shows Variational Autoencoders.

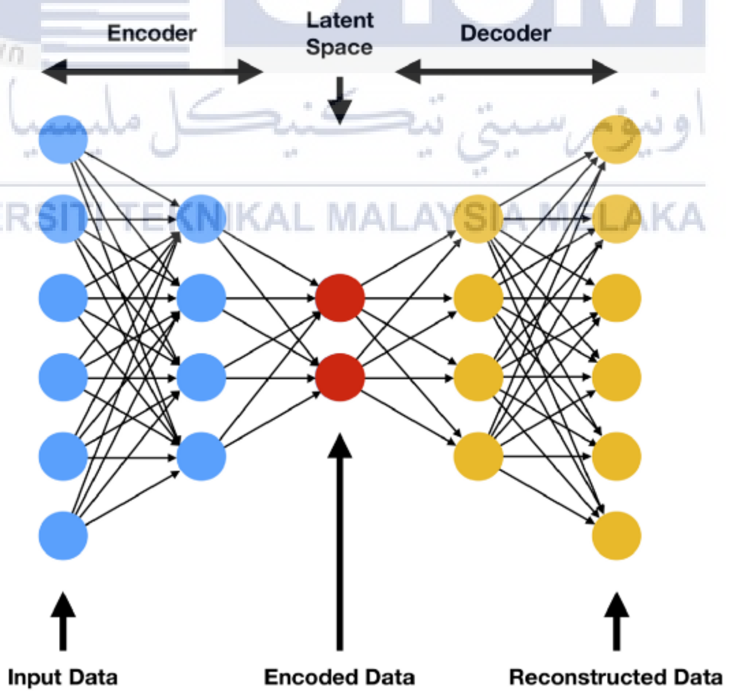


Figure 2.13 Variational Autoencoders (VAEs)

While variational autoencoders (VAEs) have made a significant impact in the field of deep generative models, there are still certain aspects of their underlying energy function that lack complete understanding. In particular, there is a prevailing belief that assuming Gaussian encoders and decoders limits the ability of VAEs to generate realistic samples. However, a comprehensive study titled "Diagnosing and Enhancing VAE Models" [30] challenges this notion and provides a detailed analysis of the VAE objective. The study demonstrates that the common perception of Gaussian encoder/decoder assumptions hindering the effectiveness of VAEs in generating realistic samples is not always accurate.

Moreover, the paper introduces a straightforward enhancement for Variational Autoencoders (VAEs) that does not require additional hyperparameters or intricate tuning. This enhancement brings about significant improvements, resulting in the generation of clear and sharp samples. It also achieves stable FID (Fréchet Inception Distance) scores, effectively narrowing the gap between VAEs and Generative Adversarial Network (GAN) models when employing a neutral architecture. Importantly, this enhancement preserves the desirable qualities of the original VAE architecture.

2.8 Comparison of article paper

Different article paper is study and the applications of each neural network are stated in the table 2.1. Table 2.1 shows comparison of article paper.

Table 2.1 Comparison of article paper

No.	Algorithm used	Applications	Paper
1	Convolutional Neural Networks (CNNs)	Primarily used for image processing tasks.	[4][5][6][7][8][9][10][11]
2	Recurrent Neural Networks (RNNs)	Suitable for sequential data.	[12][13]
3	Deep Reinforcement Learning Networks (DRLN)	Learn to make decisions and take actions based on rewards and penalties in a given environment.	[14][18][19]
4	Generative Adversarial Networks (GANs)	Utilize in production of images, videos, and voices.	[22][23][24][25][26][27][28][29]
5	Variational Autoencoders (VAEs)	Often used for generative modeling tasks.	[30]

2.9 Summary

In summary, based on the previous research, a suitable computer vision techniques for image processing tasks is selected. CNNs are suitable for image processing tasks due to their ability to automatically learn hierarchical patterns and spatial relationships within images. The convolutional layers perform localized operations by applying filters to extract features from different regions of the input image. Pooling layers downsample the feature maps, reducing their spatial dimensions. This hierarchical feature extraction allows CNNs to capture important visual characteristics at different scales. Additionally, weight sharing in CNNs enables parameter efficiency and translation invariance, making them highly effective in handling large image datasets. These characteristics make CNNs a powerful tool for tasks like image classification and object detection.

19.2.2024

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter will cover into the methods and processess employed throughout the project. The methodology includes components such as research methodologies, and experimental procedures that were aimed to direct the project towards its goals. Several flow charts have been included in this project to illustrate and explain the procedure. These flow charts are useful for describing and clarifying the sequential processes involved in project execution, as well as providing a visual depiction of the process and assisting in the understanding of the project's structure and dependencies. By following the approach and employing the flow charts, the researhcer can efficiently plan, execute, and monitor the progress of the project, ensuring that it stays on track and achieves its desired goals.

3.2 Methodology

To understand the research project, the project has been separated into 4 milestones. Each milestone will describe the activities that have been done. Figure 3.1 shows the flowchart of the methodology of this project.

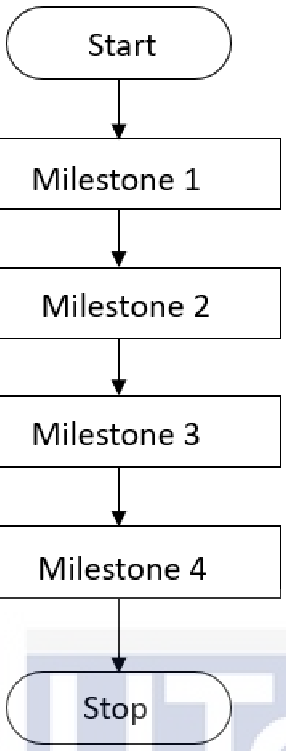


Figure 3.1 Methodology Flowchart

3.3 First Milestone

- Activity 1: Project Objectives

The project objectives were discussed with the supervisor to ensure they do not run out of the project scope. This project is to design and implement an object-following algorithm that processes video data from the DJI Tello drone's camera and sends flight commands to the drone's flight controller in real time.

To achive the objectives, the hardware used in this project is DJI Tello drone. This drone is programmable and supports the Tello SDK, which enables developers to create bespoke applications and functionalities. An infared sensor located on the bottom of the drone for precise hovering. This drone also support wireless connectivity and supports programming using various programming language.

Besides, it have maximum 13 minutes flight time, 720 HD transmission video, and 5 MP image sensor. Moreover, the software used in this project is Pycharm. PyCharm is an integrated development environment (IDE) used for programming in Python. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems, and supports web development with Django. OpenCV are intepret in Pychram, which is a powerful library and an effective tool for image processing and computer vision tasks.

- Activity 2: Literature Review

  To learn how to develop an object tracking system, it is required to read relevant topic research articles from a variety of sources. Project scopes direct researchers in the right direction in order to summarise research articles and provide a better knowledge of the project. Figure 3.2 shows the literature review flowchart of this project.

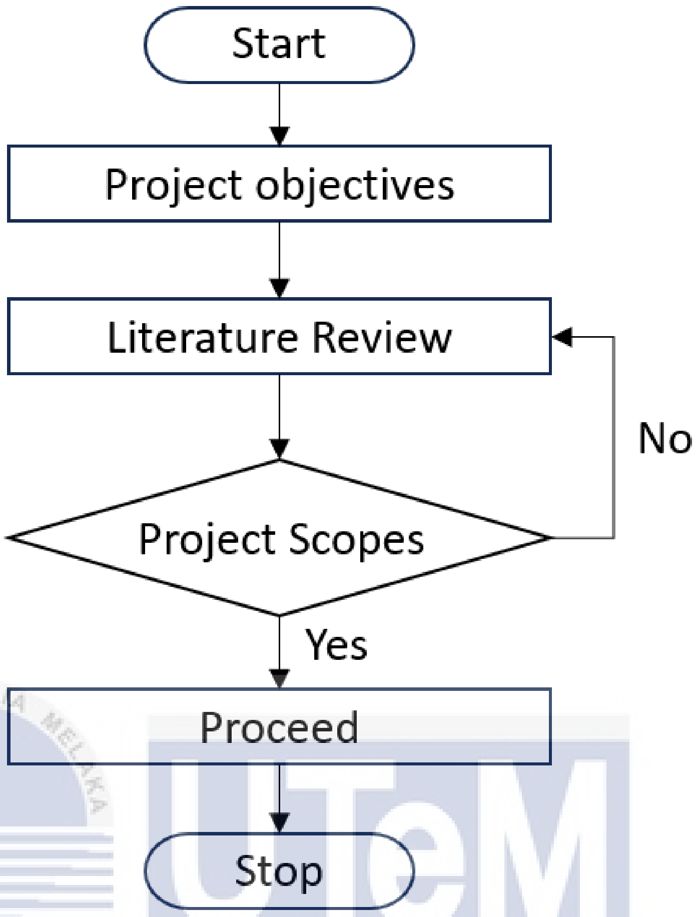


Figure 3.2 Literature review flowchart

3.4 Second Milestone

- <u>Activity 3: Design the code to program the drone</u>

The programming codes will be designed in this activity. The programming codes will be written in Pycharm with OpenCV library. The programming codes influence the performance of the drone. If any error occurs in this part, the programming codes need to be rewritten to get the expected results. The programming codes will be implemented later in the simulation part. Figure 3.3 shows the design flowchart of this project.

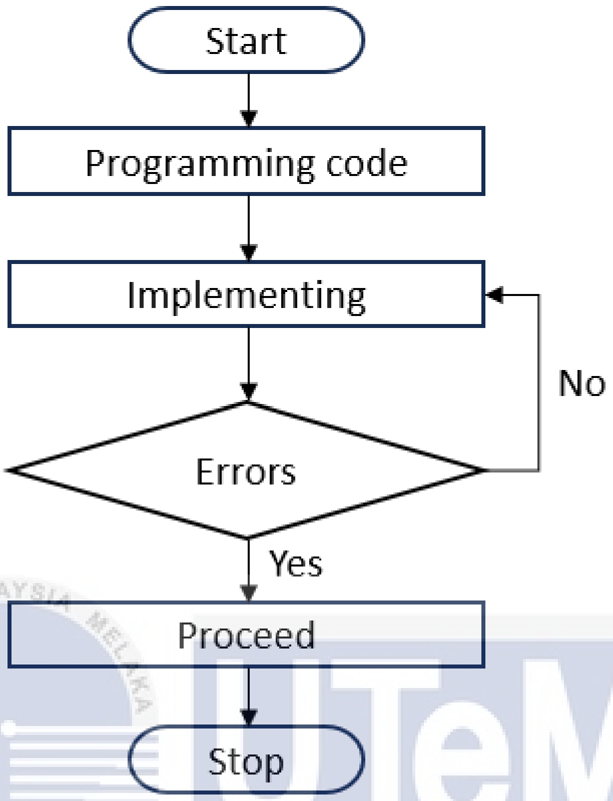
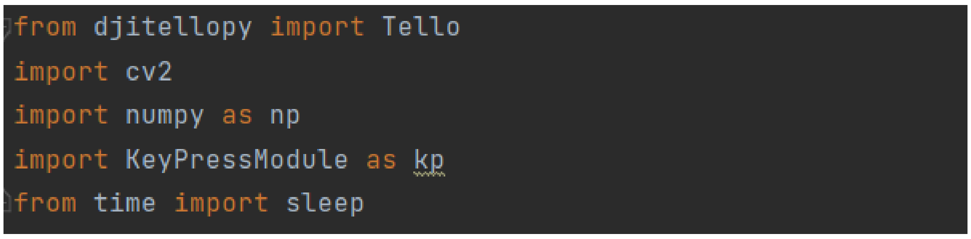


Figure 3.3 Designing Flowchart

3.5 Project Design

This part will be explaining the design of the project including programming codes, and safety distance to avoid collision. The programming codes is written using Python with OpenCV library. Some of the example of codes have shown in Figure 3.4, Figure 3.5, Figure 3.6 and Figure 3.7. Completed code can refer to the Appendix A.

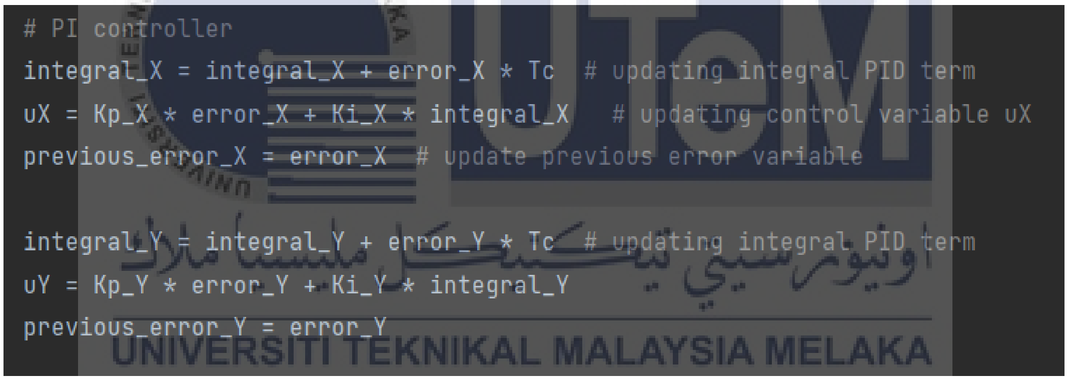
A library is a collection of existing functions that can be used in our code. The import keyword lets us import entire libraries or specific library functions into our code. The codes in Figure 3.4 shows the codes import from the library. 1.2.2024



```python
from djitellopy import Tello
import cv2
import numpy as np
import KeyPressModule as kp
from time import sleep
```

Figure 3.4 Codes import from the libraries

In order to make the object-following system achieve stable, accurately and safety state, a PID is a good control mechanism which can use. In this system only proportional controller (P-controller) and integral controller is used. The P-controller will make the drone stay in specific height. Whereas the I – controller accumulate the error overtime enable the drone more stable and precise hovering at the specific height. Figure 3.5 shows the codes to perform PID.

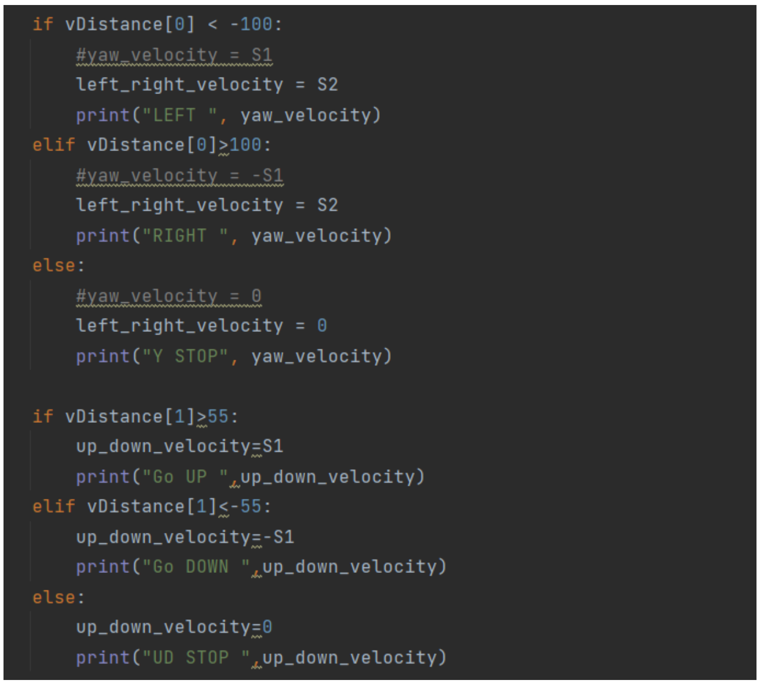


```python
# PI controller
integral_X = integral_X + error_X * Tc  # updating integral PID term
uX = Kp_X * error_X + Ki_X * integral_X   # updating control variable uX
previous_error_X = error_X  # update previous error variable

integral_Y = integral_Y + error_Y * Tc  # updating integral PID term
uY = Kp_Y * error_Y + Ki_Y * integral_Y
previous_error_Y = error_Y
```

Figure 3.5 Codes to perform PID

An algorithm is used to control the motion for the drone. In image processing the distance is count in pixels instead of meter or centimeter. Figure 3.6 shows the codes for control left, right, up and, down motion.

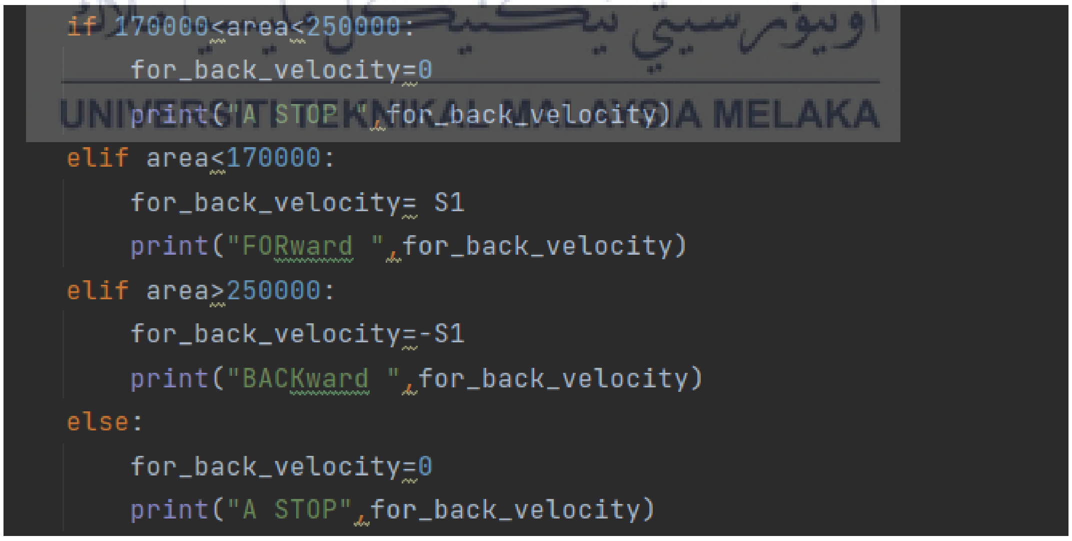


```
if vDistance[0] < -100:
    #yaw_velocity = S1
    left_right_velocity = S2
    print("LEFT ", yaw_velocity)
elif vDistance[0]>100:
    #yaw_velocity = -S1
    left_right_velocity = S2
    print("RIGHT ", yaw_velocity)
else:
    #yaw_velocity = 0
    left_right_velocity = 0
    print("Y STOP", yaw_velocity)

if vDistance[1]>55:
    up_down_velocity=S1
    print("Go UP ",up_down_velocity)
elif vDistance[1]<-55:
    up_down_velocity=-S1
    print("Go DOWN ",up_down_velocity)
else:
    up_down_velocity=0
    print("UD STOP ",up_down_velocity)
```

Figure 3.6 Codes for control left, right, up and, down motion

Next, the forward and backward motion of the drone is controlling by the area detected from the object in the image. Figure 3.7 shows the codes for control forward and backward motion.



```
if 170000<area<250000:
    for_back_velocity=0
    print("A STOP ",for_back_velocity)
elif area<170000:
    for_back_velocity= S1
    print("FORward ",for_back_velocity)
elif area>250000:
    for_back_velocity=-S1
    print("BACKward ",for_back_velocity)
else:
    for_back_velocity=0
    print("A STOP",for_back_velocity)
```

Figure 3.7 Codes for control forward and backward motion

3.6 Third Milestone

- Activity 4: Object tracking system based on camera view

  In this part, when the drone takes off without any error such as the motor being stuck or broken. If errors occur, the drone will land else the program will proceed to object detection. From the flowchart below, when the drone detects the object it will show the shape of the object in the processing view, when giving a command to stop it the drone will be landing and the whole process will stop. Figure 3.8 shows the Object-tracking system flowchart of this project.

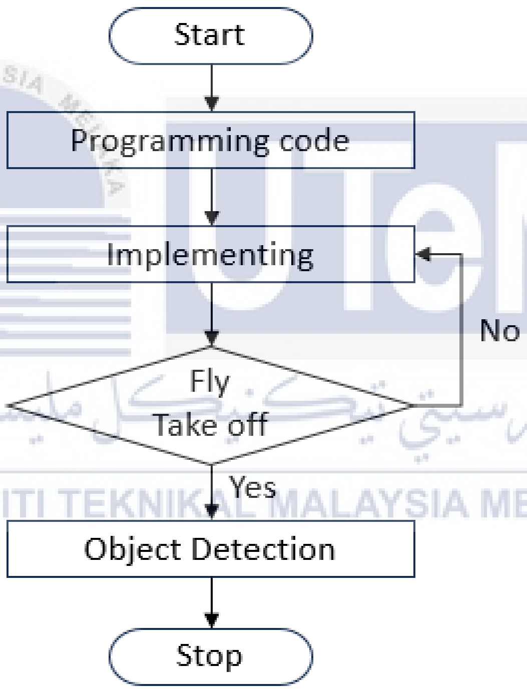


Figure 3.8 Object-tracking system flowchart

3.7 Fourth Milestone

- Activity 5: Object-following system based on positioning from the camera

  In this part, when the drone takes off without any error such as the motor being stuck or broken. In contrast, once errors occur, the drone will land else the program will proceed to object detection. From the flowchart below, when the drone scans the object on top of the view, the drone will rise. If the scanning positioning of the object is at the bottom, the drone will lower the level until finding the center position. If the drone’s scanning is at the left or right, the drone will be based on the position to slide it. Besides, the object is at the center of the view the drone will hold at the high. Futhermore, if the object is near to the drone, the drone will move backward. While the object is far away from drone, the drone will move forward. Figure 3.9 shows the Object-following system flowchart of this project.

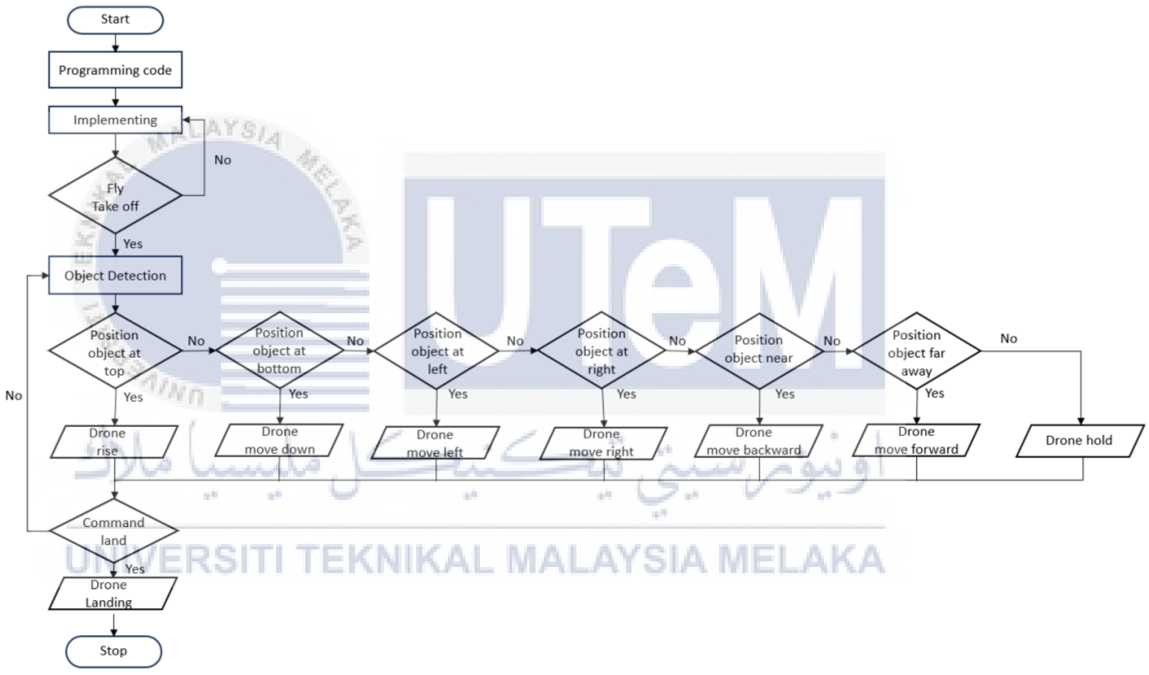


Figure 3.9 Object-following system flowchart

Additionally, to ensure that the drone and person is always in a safe distance. An algorithm has developed as illustrated in figure 3.10. During the tracking process, the drone is always higher than person and maintain a safety distance with the person where h denotes high whereas the d denotes distance. Figure 3.10 shows the concept of safety distance between drone and person.

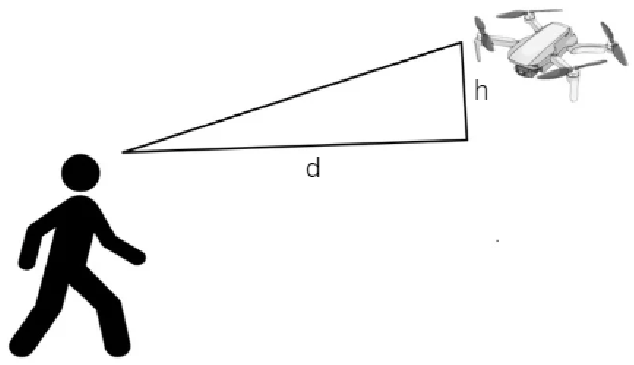


Figure 3.10 Concept of safety distance between drone and person

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Introduction

This chapter presents the results and analysis on the development of an object-following system by using DJI Tello drone. A few experiment has been studies and discuss in this chapter.

Refer to the algortihm illustrated in figure 3.10 concept of safety distance between drone and person. This algorithm has tested in indoor. The height of the person in figure 4.1 is 155 cm, wheares the height of the person in figure 4.2 is 162 cm. After applied the algorithm, we can see that the outputs of the height of drone is 165 cm and 172 cm respectively, which display on the left with purple color text.

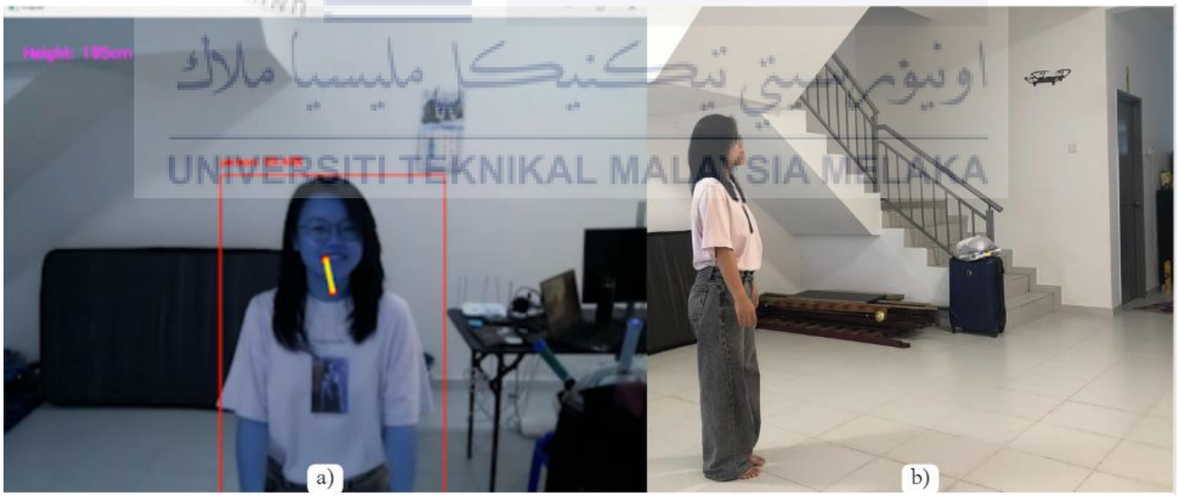


Figure 4.1 Vision from a) drone; b) reality

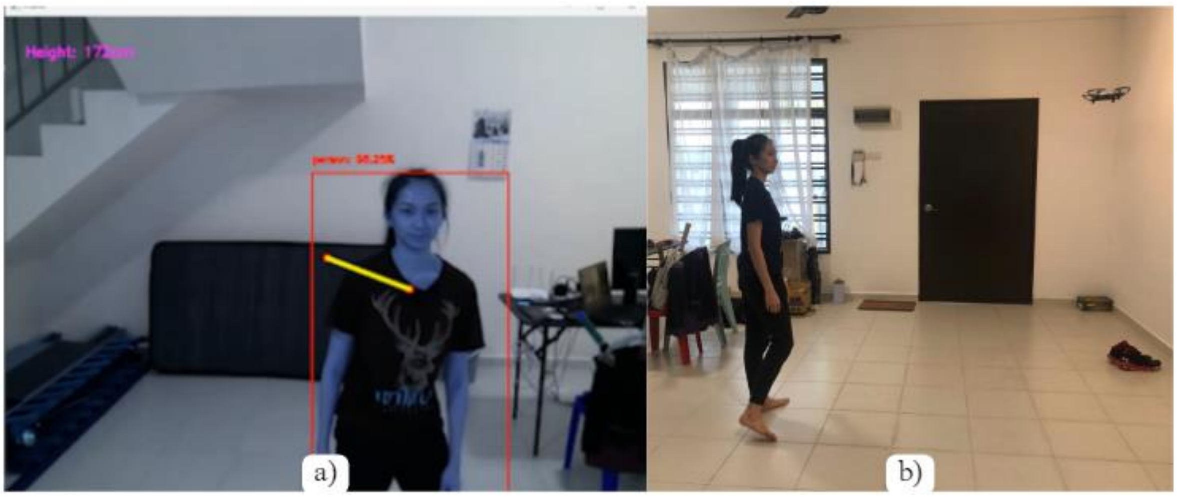


Figure 4.2 Vision from a) drone; b) reality

4.2 Proper detection range Between Person and DJI Tello Drone

This experiment aimed to determine the proper detection range between the person and the DJI Tello drone. The experimental areas were set outdoors. The drone and person initially are at the starting point, then the person will walk towards the ending points, while the drone will stay static at the starting point. The distance between the starting point and the ending point is 10 meters. Figure 4.3 shows the experimental area.

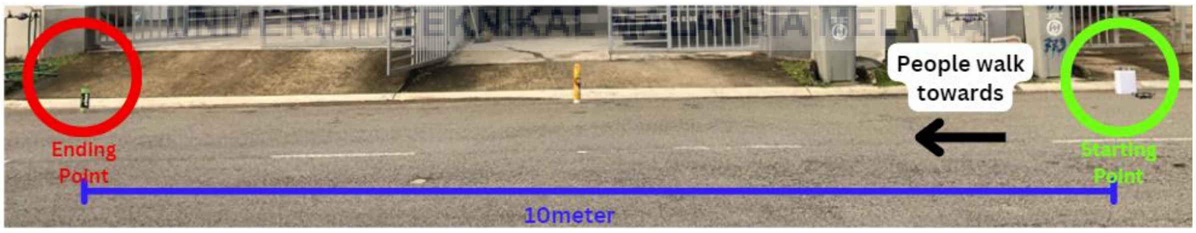


Figure 4.3 Experimental area

From figures 4.4 and figures 4.5 we can observe that when the person is still in the range of 10 meters, the person is still detectable. However, after surpassing the ending point, the person detection has lost.

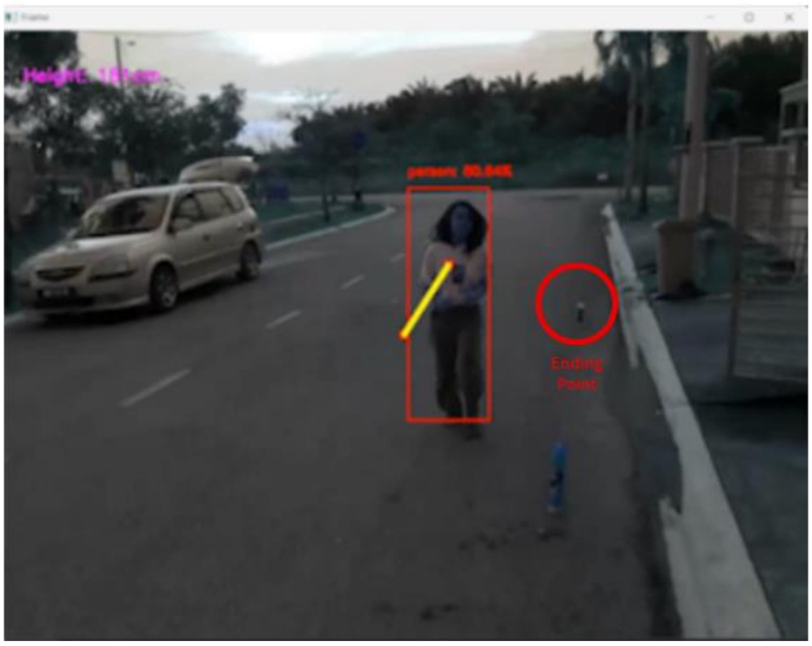


Figure 4.4 Person in the range of 10m

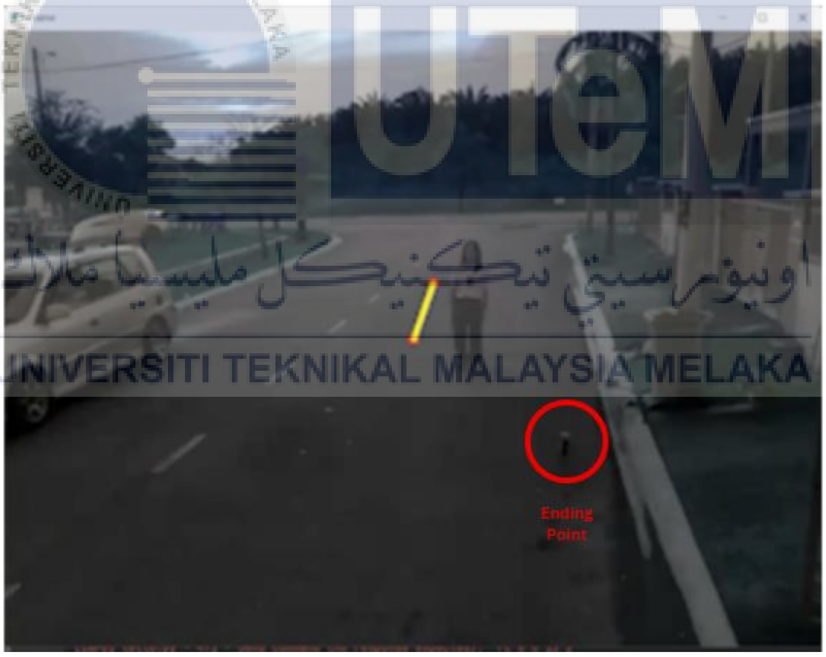


Figure 4.5 Person surpass 10m

The distance test in this experiment are 1 m, 5 m, 10 m and more than 10 m, where (/) indicates detected, (X) indicates no detection. This experiment has tested 5 round to make sure the data collected is accurate. Table 4.1 shows the results of detection of person in difference distances.

Table 4.1 Results of detection of person in difference distances

Round	Distance(m)			
	1	5	10	<10
1	/	/	/	X
2	/	/	/	X
3	/	/	/	X
4	/	/	/	X
5	/	/	/	X

4.3 Exploration of Light Intensity for Human Motion Detection

This experiment is aimed at exploring light intensity for target motion detection. The experiment is divided into two parts, explored light intensity in an indoor environment and, explored light intensity in an outdoor environment. The Lux Light Meter Pro mobile application installed on the phone was used as an instrument to examine the light intensity. The indoor experiment is conducted in the daytime and separated into two parts, one with the lamp-on and one with lamp-off.

The light intensity range of the environment lamp-off is a maximum of 26 lux whereas the light intensity range of the situation with lamp-on is a maximum of 34 lux. Figure 4.6 shows indoor light intensity with lamp-off and lamp-on.

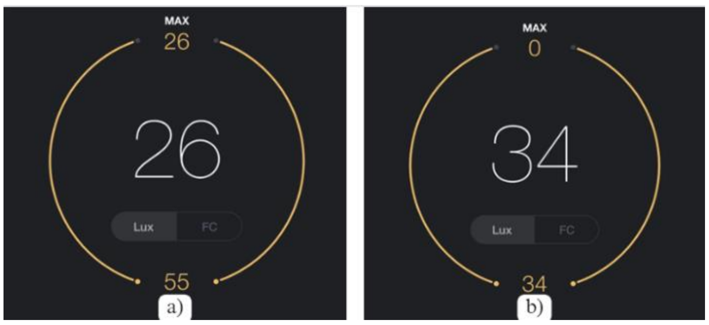


Figure 4.6 Indoor light intensity a) lamp-off ; b) lamp-on

Figure 4.7 shows the results of experiments conducted in a lamp-off environment.

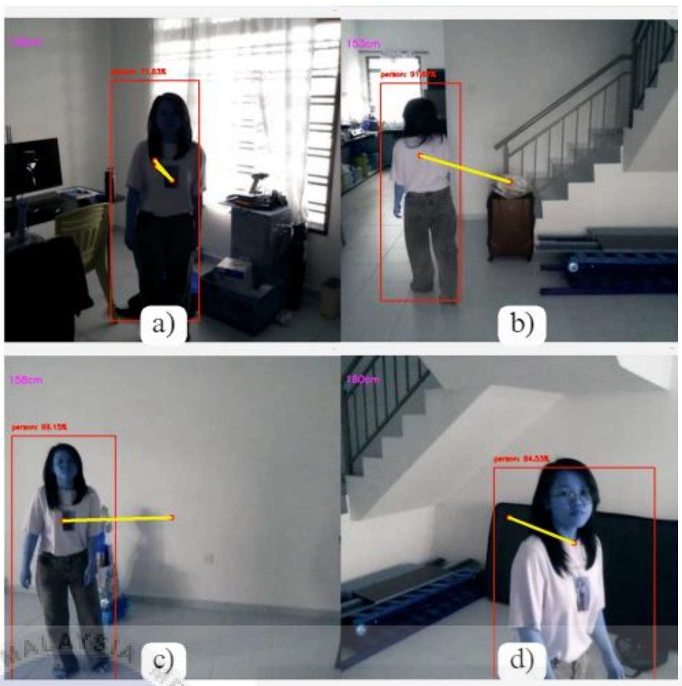


Figure 4.7 Person a) moving forward; b) moving backward; c) move left; d) move right

Figure 4.8 shows the results of experiments conducted in a lamp-on environment.

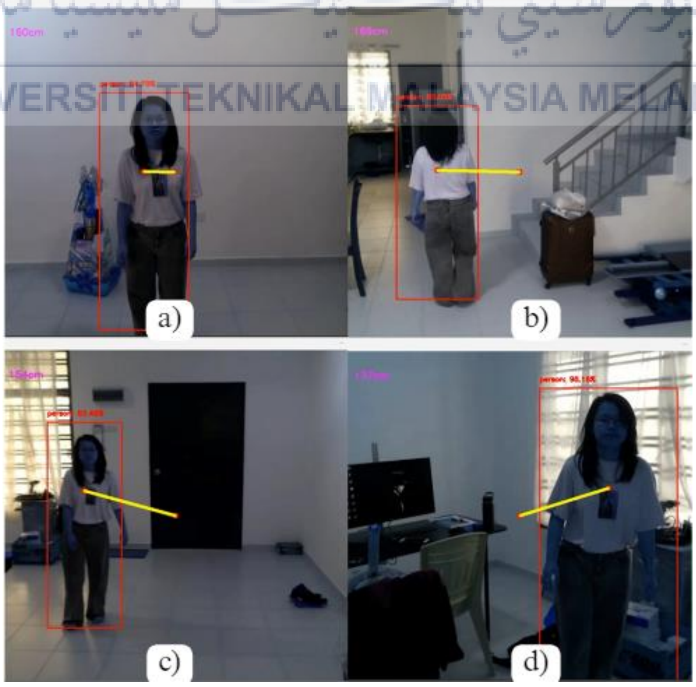


Figure 4.8 Person a) moving forward; b) moving backward; c) move left; d) move right

From figure 4.7 and 4.8 we can observe the visibility of the target is clear and enables detection of the motion of the target moving forward, moving backward, moving left, and moving right.

Next, the experiment conducted in an outdoor environment is divided into three time periods, which are morning (10.00-11.00), afternoon (14.00-15.00), and evening (18.00-19.00). The light intensity in the morning is 3085 lux, while the afternoon is 3442 lux and the evening is a maximum of 300 lux. Figure 4.9 shows the light intensity for mornig, afternoon, and night respectively.

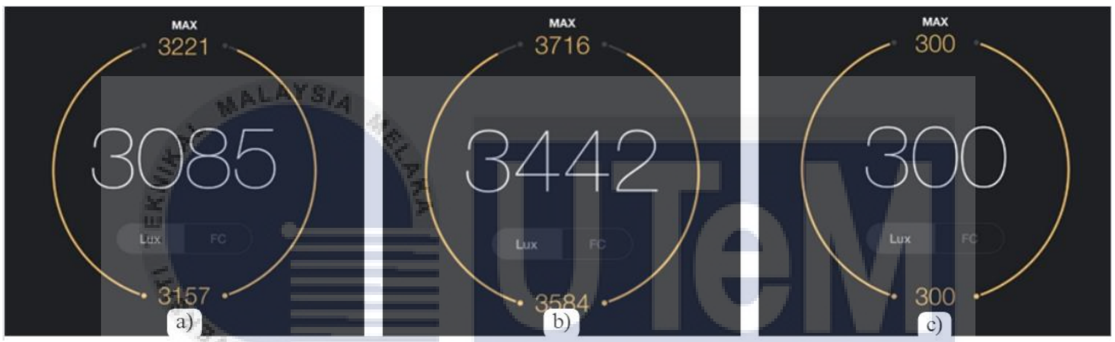


Figure 4.9 Light intensity in a) morning ; b) afternoon; c) evening

Figure 4.10 shows the results of experiments conducted in the morning.

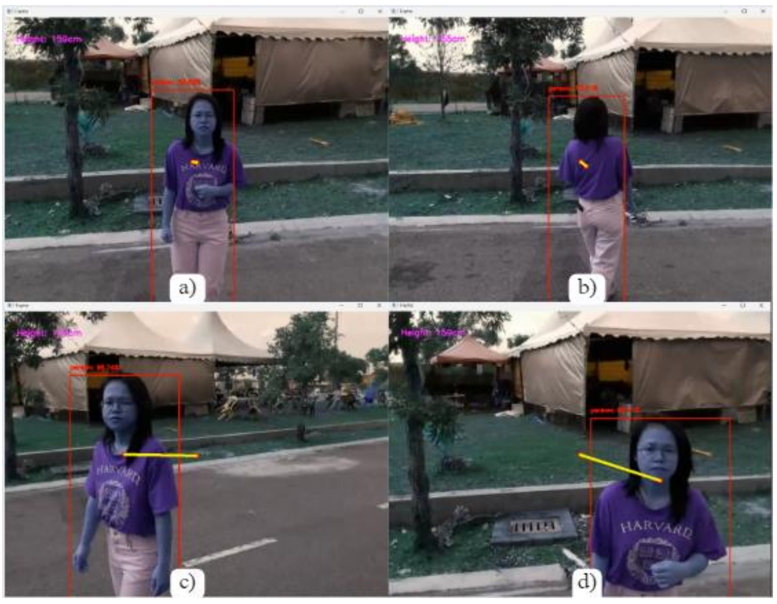


Figure 4.10 Person a) moving forward; b) moving backward; c) move left; d) move right

Figure 4.11 shows the results of experiments conducted in afternoon.

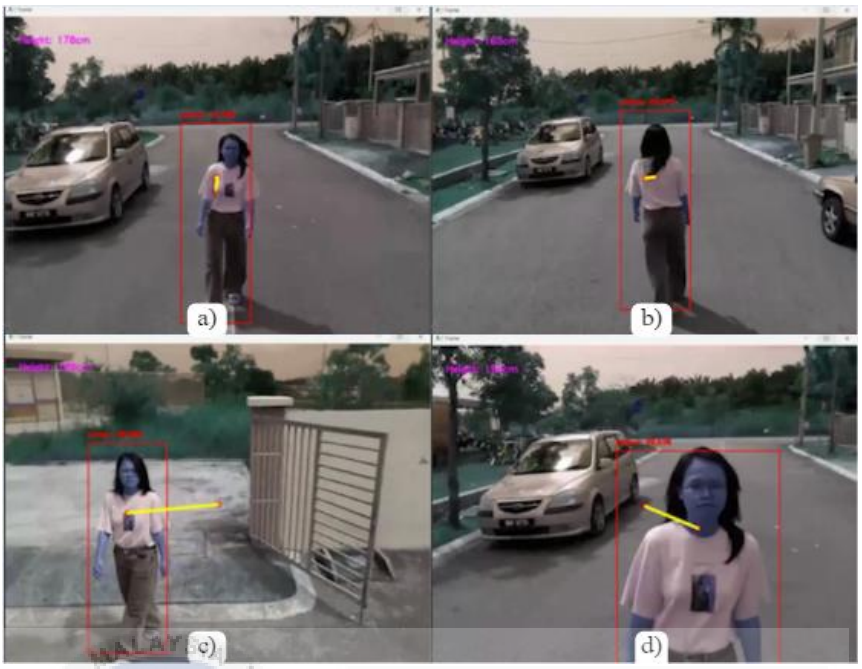


Figure 4.11 Person a) moving forward; b) moving backward; c) move left; d) move right

Figure 4.12 shows the results of experiments conducted in the evening.

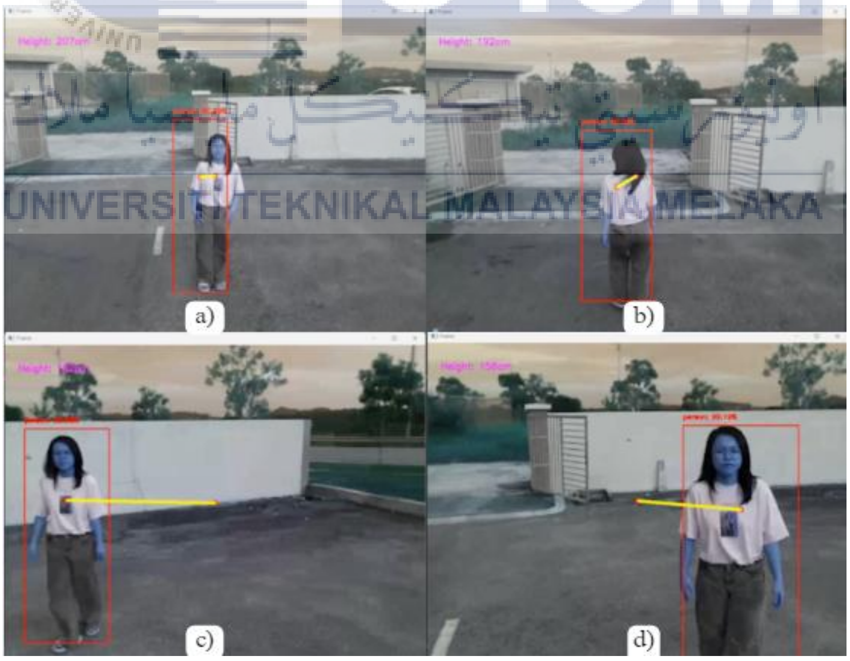


Figure 4.12 Person a) moving forward; b) moving backward; c) move left; d) move right

From figure 4.10, 4.11 and 4.12 we can observe the visibility of the target is clear and enables detection of the motion of the target moving forward, moving backward, moving left, and moving right.

Table 4.2 shows results of detection motion in different environment where (/) indicates detected, (X) indicates no detection

Table 4.2 Results of detection motion in different environment

Environment	Human motion			
	Move forward	Move backward	Move left	Move right
(Indoor) Lamp-off	/	/	/	/
(Indoor) Lamp-on	/	/	/	/
(Outdoor) Morning	/	/	/	/
(Outdoor) Afternoon	/	/	/	/
(Outdoor) Evening	/	/	/	/

4.4 Automatic tracking of target motion

This section demonstrates the automatic object-following system by using DJI Tello drone to follow the motion of the target. We will discuss three scenarios in this experiment. The first scenario is a drone following a person walking in curve motion. Secondly, the drone tracks a person moving in a multi-person scenario. Third scenario, the drone searches for a person when no target is tracked.

In the first scenario, the curve motion path is set as figure 4.13. The person will move from starting point to ending point. When the drone detects person it will following the person along path. Figure 4.13 shows the curve motion path.

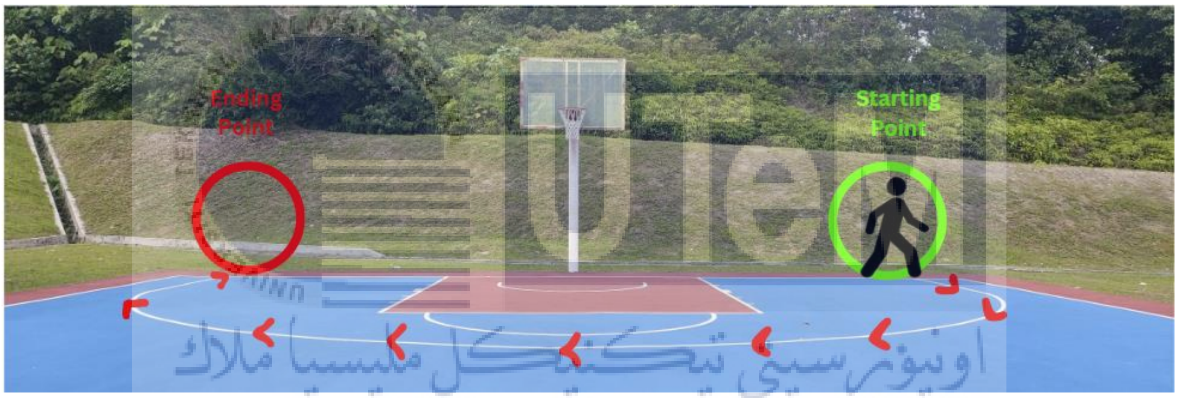


Figure 4.13 Curve motion path

The motion of person walking along path detected by drone is recorded. Total 16 sampling frames is selected from the recorded video. The number A1 denotes the staring point while number A16 denotes the ending point as illustrated in figure 4.14. Figure 4.14 shows the sampling frames A1-A16 captured from the recording video.

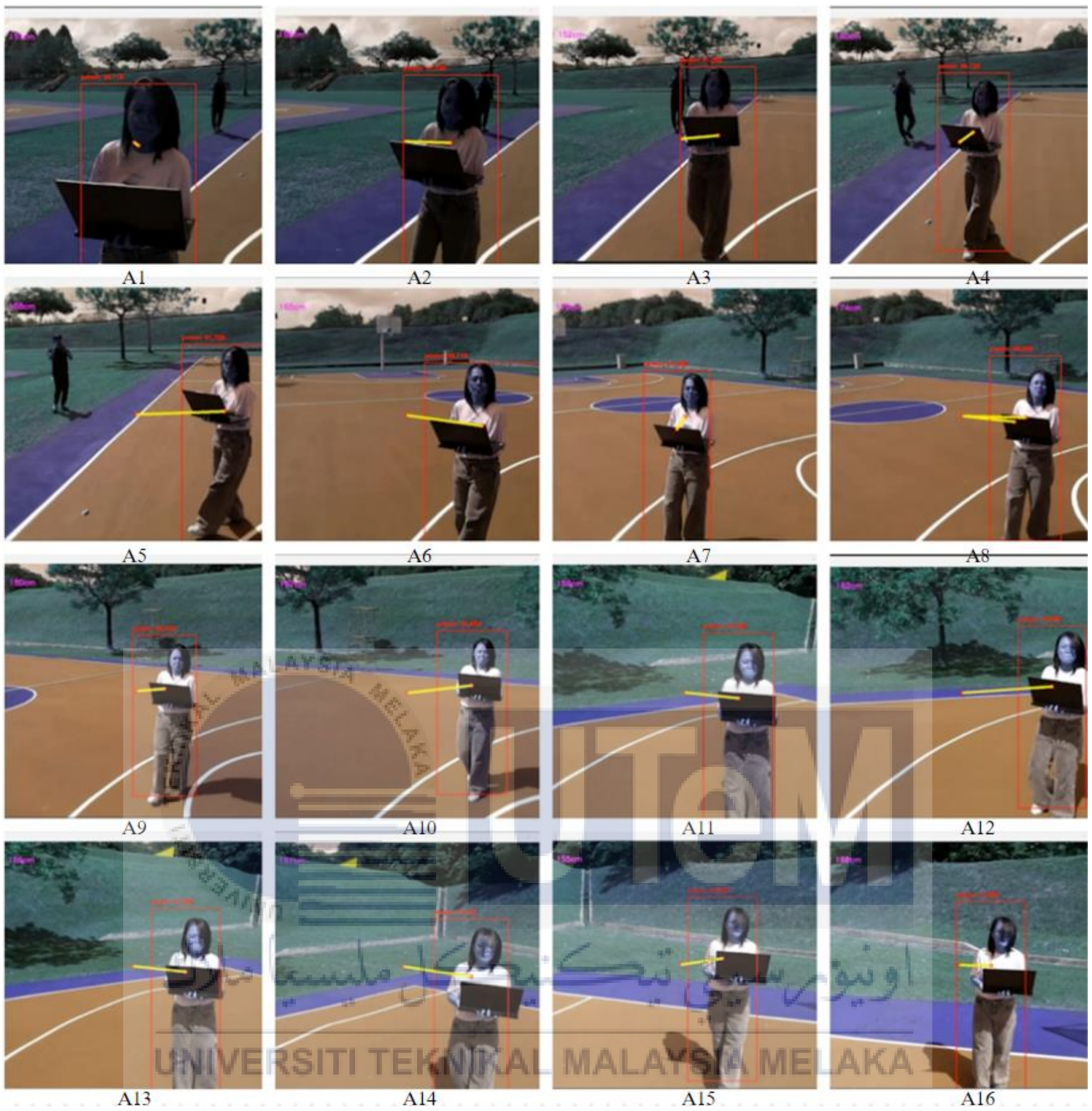


Figure 4.14 Sampling frames A1-A16 captured from recording video

In second scneario, when there is more than one person in the tracking frame, the object-following system will tend to detect the person who is closest to the drone. Initially, there is only one person in the frame, a new person comes closer to the drone, then the drone will track the new person instead of the first person, as illustrated in B1-B16 of figure 4.15. Figure 4.15 shows the target detection when a new person comes closer than first person.

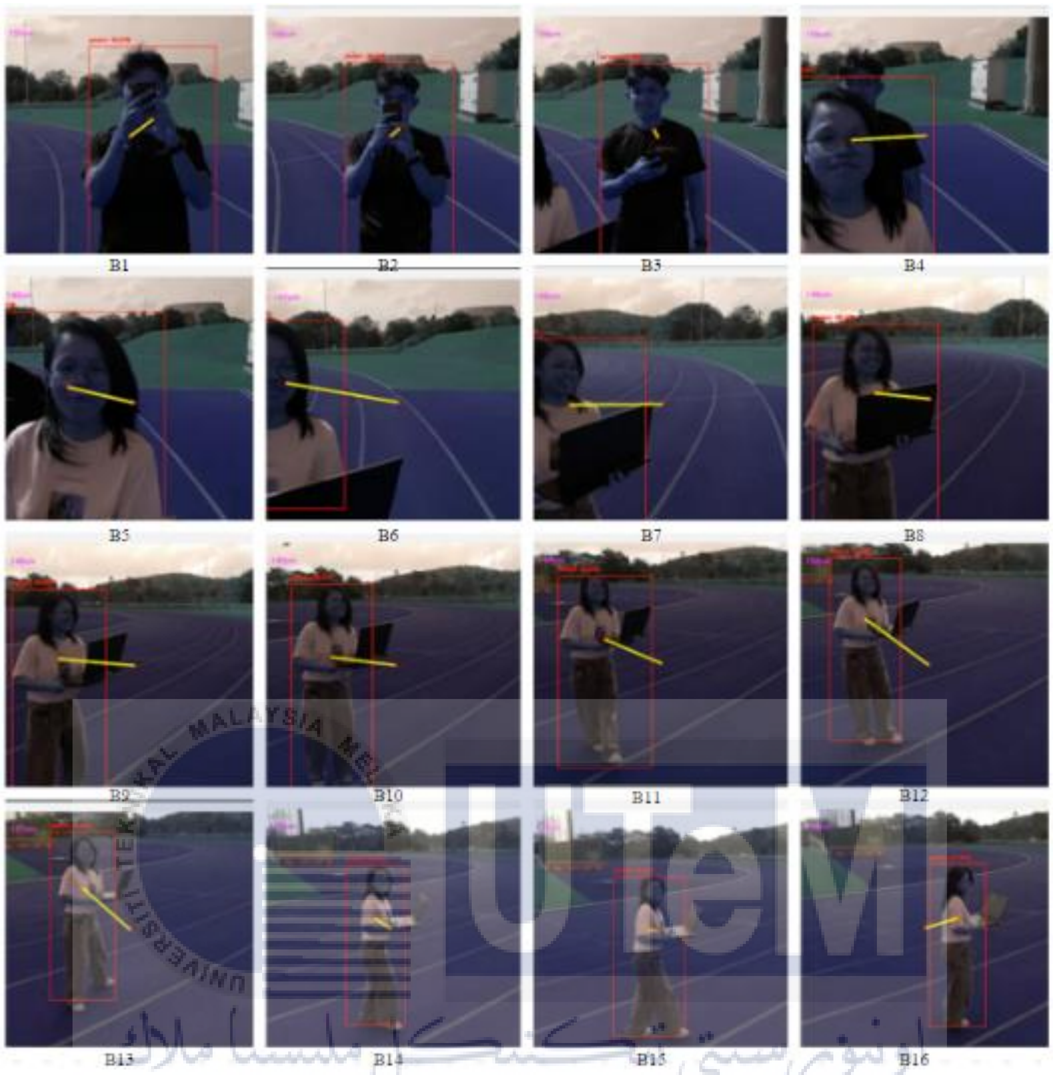


Figure 4.15 Target detection when a new person comes closer than first person

Third scenario, during the tracking process, if the person is out of frame, the system will start searching, the drone will rotate from the previous location where the target was lost until a person is detected, as illustrated in C1-C16 of figure 4.16. Figure 4.16 shows the drone action when no person is in the frame.

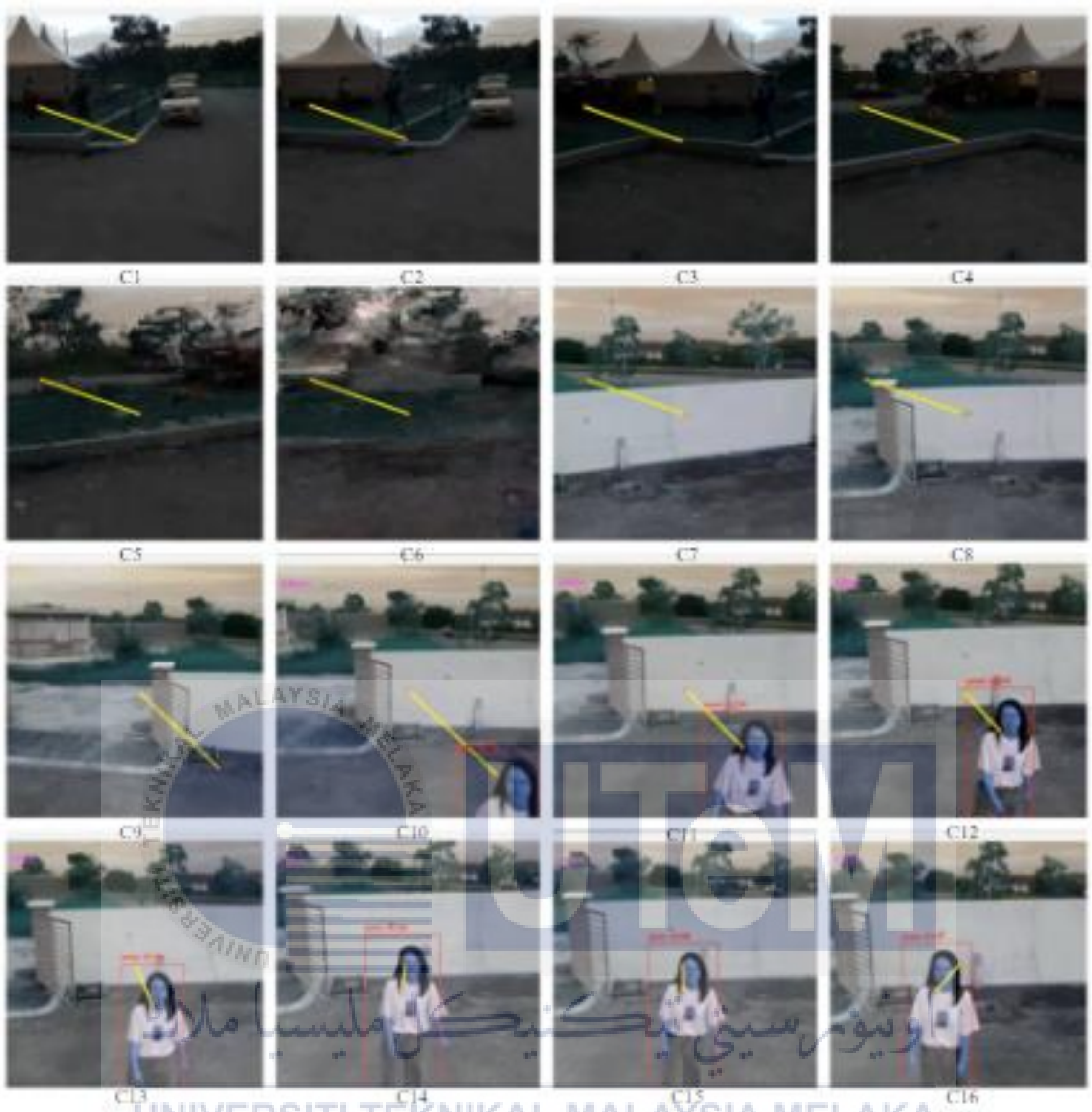


Figure 4.16 Drone action when no person is in the frame

4.5 Summary

This chapter presented case studies to demonstrate applicability of the object-following system. From the experiment, the proper detection range between the person and the DJI Tello drone is between 1 m to 10 m. Moreover, based on the results of light intensity experiment conducted in indoor and outdoors environments, we can conclude that the object-following system can work in most of the environments. The automatic tracking of target motion are successful and three different scenario has discussed.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

In conclusion, computer vision techniques play a crucial role in the design and implementation of an object-following system using the DJI Tello drone. In this project, Convolutional Neural Networks (CNNs), a powerful computer vision algorithm, will be employed for real-time target detection. A DJI Tello drone will be control via Wi-Fi using the Python programming language, and a program will be coded to transmit flying commands to its flight controller to follow an object. Initially, some article paper from previous research related to object tracking algorithms has been study. These papers provided valuable insights and served as a foundation for the project. Throughout the project, the primary objective was to develop a comprehensive algorithm that would enable the drone to effectively track object. Several iterations of the algorithm were designed, implemented, and evaluated to ensure optimal performance. A PID has integrated in the system to ensures the drone maintains a safe distance from the object and avoids collisions with obstacles. Last but not least, a few experiment have conducted to prove the performance of the object-following system.Overall, the algorithm have been fully developed and fine-tuned, empowering the DJI Tello drone with the ability to autonomously follow objects while ensuring a safe distance is maintained at all times.

5.1 Potential for Commercialization

The object-following system by using DJI Tello drone project has significant potential for commercialization, across a wide range of sectors. Surveillance and monitoring are important areas where these drones can be used to provide effective security measures.

Their capacity to travel particular regions autonomously, track movements, and offer live video feeds without continual human control improves surveillance capabilities, making them invaluable for protecting important infrastructure or private properties. Autonomous tracked drones provide a lifeline in search and rescue operations by quickly reaching isolated or hazardous regions and delivering real-time aerial pictures that aid in the identification of survivors. Furthermore, the adaptability of drones extends to aerial photography and filmmaking, allowing experts to record amazing sights from altitudes inaccessible by people.

5.2 Future Works

There are several potential future improvements that can be added to this object-following system to enhance their capabilities. In current stage, the object-following system will change the target based on the algorithm implemented while following, therefore a techniques can be developed and implement to locked a specific target during the tracking process in future. Additionally, improving energy efficiency or adopting more advanced battery technologies can increase the operational time of the drones. This would allow drone to travel further or stay in the field for longer periods of time without having to recharge. Futhermore, advanced obstacle avoidance can also enhance this object-following system. Using improved obstacle detection and avoidance algorithms can help the drone navigate complex environments. This may involve utilising advanced sensors, computer vision, and machine learning algorithms to detect and respond to dynamic barriers in real time.

REFERENCES

[1] K. Pulli, A. Baksheev, K. Kornyakov, and V. Eruhimov, "Realtime Computer Vision with OpenCV," *Queue*, vol. 10, no. 4, pp. 40–56, Apr. 2012, doi: 10.1145/2181796.2206309.

[2] "What is OpenCV? - An Introduction Guide - Python Geeks." Accessed: May 04, 2023. [Online]. Available: https://pythongeeks.org/what-is-opencv/

[3] R. Szeliski, "Computer Vision: Algorithms and Applications," 2010. [Online]. Available: http://szeliski.org/Book/.

[4] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *J Physiol*, vol. 195, no. 1, pp. 215–243, Mar. 1968, doi: 10.1113/jphysiol.1968.sp008455.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," 2012. [Online]. Available: http://code.google.com/p/cuda-convnet/

[6] A. Ghosh, A. Sufian, F. Sultana, A. Chakrabarti, and D. De, "Fundamental concepts of convolutional neural network," in *Intelligent Systems Reference Library*, vol. 172, Springer, 2019, pp. 519–567. doi: 10.1007/978-3-030-32644-9_36.

[7] F. Sultana, A. Sufian, and P. Dutta, "A Review of Object Detection Models based on Convolutional Neural Network," May 2019, doi: 10.1007/978-981-15-4288-6_1.

[8] R. Girshick, "Fast R-CNN", Accessed: Jun. 13, 2023. [Online]. Available: https://github.com/rbgirshick/

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 779–788, Jun. 2015, doi: 10.1109/CVPR.2016.91.

[10] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9905 LNCS, pp. 21–37, Dec. 2015, doi: 10.1007/978-3-319-46448-0_2.

[11] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal Loss for Dense Object Detection," *IEEE Trans Pattern Anal Mach Intell*, vol. 42, no. 2, pp. 318–327, Aug. 2017, doi: 10.1109/TPAMI.2018.2858826.

[12] A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network," Aug. 2018, doi: 10.1016/j.physd.2019.132306.

[13] L. P. Osco *et al.*, "A review on deep learning in UAV remote sensing," *International Journal of Applied Earth Observation and Geoinformation*, vol. 102, p. 102456, Oct. 2021, doi: 10.1016/J.JAG.2021.102456.

[14] D. H. T. Minh *et al.*, "Deep Recurrent Neural Networks for mapping winter vegetation quality coverage via multi-temporal SAR Sentinel-1," Aug. 2017, [Online]. Available: http://arxiv.org/abs/1708.03694

[15] Q. Feng *et al.*, "Multi-temporal unmanned aerial vehicle remote sensing for vegetable mapping using an attention-based recurrent convolutional neural network," *Remote Sens (Basel)*, vol. 12, no. 10, May 2020, doi: 10.3390/rs12101668.

[16] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Foundations and Trends in Machine Learning*, vol. 11, no. 3–4, pp. 219–354, Dec. 2018, doi: 10.1561/2200000071.

[17] P. Reizinger and M. Szemenyei, "Attention-based Curiosity-driven Exploration in Deep Reinforcement Learning," Oct. 2019, doi: 10.1109/ICASSP40776.2020.9054546.
[18] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," Jan. 2018, [Online]. Available: http://arxiv.org/abs/1801.01290
[19] V. Mnih *et al.*, "Asynchronous Methods for Deep Reinforcement Learning," Feb. 2016, [Online]. Available: http://arxiv.org/abs/1602.01783
[20] B. Ding, M. T. Cychowski, Y. Xi, W. Cai, and B. Huang, "Model predictive control," *Journal of Control Science and Engineering*, vol. 2012, 2012, doi: 10.1155/2012/240898.
[21] I. J. Goodfellow *et al.*, "Generative Adversarial Networks," Jun. 2014, [Online]. Available: http://arxiv.org/abs/1406.2661
[22] N. ning Zheng *et al.*, "Hybrid-augmented intelligence: collaboration and cognition," *Frontiers of Information Technology and Electronic Engineering*, vol. 18, no. 2. Zhejiang University, pp. 153–179, Feb. 01, 2017. doi: 10.1631/FITEE.1700053.
[23] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," Nov. 2014, [Online]. Available: http://arxiv.org/abs/1411.1784
[24] A. Radford, L. Metz, and S. Chintala, "UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS," 2016.
[25] H. Zhang *et al.*, "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks," Dec. 2016, [Online]. Available: http://arxiv.org/abs/1612.03242
[26] X. Wang and A. Gupta, "Generative Image Modeling using Style and Structure Adversarial Networks," 2016.
[27] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," Nov. 2018, [Online]. Available: http://arxiv.org/abs/1611.07004
[28] H. Zhang, V. M. Patel, B. S. Riggan, and S. Hu, "Generative Adversarial Network-based Synthesis of Visible Faces from Polarimetric Thermal Faces," 2017.
[29] T. Zhang, A. Wiliem, S. Yang, and B. C. Lovell, "TV-GAN: Generative Adversarial Network Based Thermal to Visible Face Recognition," Dec. 2017, [Online]. Available: http://arxiv.org/abs/1712.02514
[30] B. Dai and D. Wipf, "Diagnosing and Enhancing VAE Models," Mar. 2019, [Online]. Available: http://arxiv.org/abs/1903.05789

APPENDICES

Appendix A   Programming Codes

```
import cv2
import numpy as np
import time
from djitellopy import Tello

# set points (center of the frame coordinates in pixels)
rifX = 960 / 2
rifY = 720 / 2

# PI constant
Kp_X = 0.1
Ki_X = 0.0
Kp_Y = 0.2
Ki_Y = 0.0

S1 = 30
S2 = 10
S3 = 10

UDOffset = 150
dimensions=(960,720)
cWidth=int(dimensions[0]/2)
cHeight=int(dimensions[1]/2)

# Loop time
Tc = 0.05

# PI terms initialized
integral_X = 0
error_X = 0
previous_error_X = 0
integral_Y = 0
error_Y = 0
previous_error_Y = 0

centroX_pre = rifX
centroY_pre = rifY

# neural networkq
model_config_file_path =
r"C:\Users\suen\PycharmProjects\pythonProject2\.idea\MobileNetSSD_deploy.
prototxt.txt"
model_weights_file_path =
r"C:\Users\suen\PycharmProjects\pythonProject2\.idea\MobileNetSSD_deploy.
caffemodel"

net =
cv2.dnn.readNetFromCaffe(model_config_file_path,model_weights_file_path)
# modify with the NN path
CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
           "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
           "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
           "sofa", "train", "tvmonitor"]
```

```
#colors = np.random.uniform(0, 255, size=(len(CLASSES), 3))

drone = Tello()  # declaring drone object`789
for_back_velocity=0
left_right_velocity=0
up_down_velocity=0
yaw_velocity=0
speed=10

time.sleep(2.0)  # waiting 2 seconds
print("Connecting...")
drone.connect()
print("BATTERY: ")
print(drone.get_battery())
time.sleep(1.0)
print("Loading...")
drone.streamon()  # start camera streaming
print("Takeoff...")
drone.takeoff()  # drone takeoff
drone.send_rc_control(0,10,0,0)
time.sleep(3.0)

while True:
    start = time.time()
    frame = drone.get_frame_read().frame

    cv2.circle(frame, (int(rifX), int(rifY)), 1, (0, 0, 255), 10)

    h, w, channels = frame.shape

    blob = cv2.dnn.blobFromImage(frame,
                                 0.007843, (180, 180), (0, 0, 0), True,
crop=False)

    net.setInput(blob)
    detections = net.forward()

    for i in np.arange(0, detections.shape[2]):

        idx = int(detections[0, 0, i, 1])
        confidence = detections[0, 0, i, 2]

        if CLASSES[idx] == "person" and confidence > 0.5:

            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")

            label = "{}: {:.2f}%".format(CLASSES[idx],
                                          confidence * 100)
            cv2.rectangle(frame, (startX, startY), (endX, endY),
                          (0, 0, 255), 2)
            # draw the center of the person detected
            centroX = (startX + endX) / 2
            centroY = (2 * startY + endY) / 3

            centroX_pre = centroX
            centroY_pre = centroY

            h = int(endY-startY)
            w = int(endX - startX)
            area = h * w

            vtr = np.array((cWidth,cHeight,22000))
            vtg = np.array((centroX,centroY,area))
            vDistance = vtr - vtg


            cv2.circle(frame, (int(centroX), int(centroY)), 1, (0, 0,
255), 10)

            error_X = -(rifX - centroX)
            error_Y = rifY - centroY

            cv2.line(frame, (int(rifX), int(rifY)), (int(centroX),
int(centroY)), (0, 255, 255), 5)

            y = startY - 15 if startY - 15 > 15 else startY + 15
            cv2.putText(frame, label, (startX, y),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)

            text = "Height: {}cm".format(drone.get_distance_tof())
            cv2.putText(frame, text, (int(rifX) - 450, int(rifY) - 300),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 0, 255), 2)

            # PI controller
            integral_X = integral_X + error_X * Tc  # updating integral
PID term

            uX = Kp_X * error_X + Ki_X * integral_X   # updating control
variable uX

            previous_error_X = error_X  # update previous error variable

            integral_Y = integral_Y + error_Y * Tc  # updating integral
PID term

            uY = Kp_Y * error_Y + Ki_Y * integral_Y
            previous_error_Y = error_Y


            print("Area: ", area)
            print("Distance: ", vDistance)

            if vDistance[0] < -100:
                #yaw_velocity = S1
                left_right_velocity = S2
                print("LEFT ", yaw_velocity)
            elif vDistance[0]>100:
                #yaw_velocity = -S1
                left_right_velocity = S2
                print("RIGHT ", yaw_velocity)
            else:
                #yaw_velocity = 0
                left_right_velocity = 0
                print("Y STOP", yaw_velocity)

            if vDistance[1]>55:
                up_down_velocity=S1
                print("Go UP ",up_down_velocity)
            elif vDistance[1]<-55:
                up_down_velocity=-S1
                print("Go DOWN ",up_down_velocity)
            else:
                up_down_velocity=0
                print("UD STOP ",up_down_velocity)

            if 170000<area<250000:
                for_back_velocity=0
                print("A STOP ",for_back_velocity)
            elif area<170000:
                for_back_velocity= S1
                print("FORward ",for_back_velocity)
            elif area>250000:
                for_back_velocity=-S1
                print("BACKward ",for_back_velocity)
            else:
                for_back_velocity=0
                print("A STOP",for_back_velocity)

            drone.send_rc_control(left_right_velocity, for_back_velocity,
up_down_velocity, round(uX))
            # break when a person is recognized

            break


        else:  # if nobody is recognized take as reference centerX and
centerY of the previous frame
            centroX = centroX_pre
            centroY = centroY_pre
            cv2.circle(frame, (int(centroX), int(centroY)), 1, (0, 0,
255), 10)

            error_X = -(rifX - centroX)
            error_Y = rifY - centroY

            cv2.line(frame, (int(rifX), int(rifY)), (int(centroX),
int(centroY)), (0, 255, 255), 5)

            integral_X = integral_X + error_X * Tc  # updating integral
PID term
            uX = Kp_X * error_X + Ki_X * integral_X  # updating control
variable uX
            previous_error_X = error_X  # update previous error variable

            integral_Y = integral_Y + error_Y * Tc  # updating integral
PID term
            uY = Kp_Y * error_Y + Ki_Y * integral_Y
            previous_error_Y = error_Y

            drone.send_rc_control(0, 0, round(uY), round(uX))

            continue

    cv2.imshow("Frame", frame)

    end = time.time()
    elapsed = end - start
    if Tc - elapsed > 0:
        time.sleep(Tc - elapsed)
    end_ = time.time()
    elapsed_ = end_ - start
    fps = 1 / elapsed_
    print("FPS: ", fps)
    if detections.all():
        seconds = int(elapsed % 60)
        times = str(seconds)
        cv2.putText(frame, times, (int(rifX) - 450, int(rifY) - 280),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)

    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

drone.streamoff()
cv2.destroyAllWindows()
drone.land()
print("Landing...")
print("BATTERY: ")
print(drone.get_battery())
drone.end()
```