AI-Based Approach to Simulate the Drone Dynamics in Three-Dimensional Space

A Thesis Submitted for the Degree of Master of Computer Science

P.M.D.S Amarasooriya

University of Colombo School of Computing

2024

DECLARATION

Name of the student: P.M.S Shehan Amarasooriya
Registration number: 20440049
Name of the Degree Programme: Master of Computer Science
Project/Thesis title: AI-Based Approach to Simulate the Drone Dynamics in Three-Dimensional Space

1. The project/thesis is my original work and has not been submitted previously for a degree at this or any other University/Institute. To the best of my knowledge, it does not contain any material published or written by another person, except as acknowledged in the text.

2. I understand what plagiarism is, the various types of plagiarism, how to avoid it, what my resources are, who can help me if I am unsure about a research or plagiarism issue, as well as what the consequences are at University of Colombo School of Computing (UCSC) for plagiarism.

3. I understand that ignorance is not an excuse for plagiarism and that I am responsible for clarifying, asking questions and utilizing all available resources to educate myself and prevent myself from plagiarizing.

4. I am also aware of the dangers of using online plagiarism checkers and sites that offer essays for sale. I understand that if I use these resources, I am solely responsible for the consequences of my actions.

5. I assure that any work I submit with my name on it will reflect my own ideas and effort. I will properly cite all material that is not my own.

6. I understand that there is no acceptable excuse for committing plagiarism and that doing so is a violation of the Student Code of Conduct.

Signature of the Student	**Date (DD/MM/YYYY)**
[illegible]	27/10/2024

Certified by Supervisor(s)

This is to certify that this project/thesis is based on the work of the above-mentioned student under my/our supervision. The thesis has been prepared according to the format stipulated and is of an acceptable standard.

	Supervisor
Name	Dr. K.D Sandaruwan
Signature	[illegible]
Date	27.10.2024

I would like to dedicate this thesis to

my beloved family,

for their love, support, and encouragement.

&

academic and non-academic staff of

University of Colombo School of Computing

for their support and guidance given

to make this thesis a success.

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank the project supervisor, Dr. K D Sandaruwan, for his unrestricted assistance and direction in helping me complete this thesis. I also want to express my gratitude to the MCS3204 module coordinators for their constant advice during the project's duration. I would especially want to thank the University of Colombo School of Computing for allowing me to conduct this research study.

Finally, I would like to express my gratitude to everyone who wishes to remain anonymous. The assistance they gave me was invaluable and greatly appreciated.

ABSTRACT

Unmanned aerial vehicles, often known as drones, have unlocked new possibilities in a range of industries, including surveillance, transportation, construction, and agriculture. Analysis of drone behavior is challenging due to the complex interplay of several factors, including speed, altitude, orientation, and trajectory. Simulations of drone dynamics are an important requirement for many fields because simulations allow researchers to create and test drones in complex scenarios that might be challenging or unsafe. There are numerous proposed drone dynamic models on the base of Newtonian and fluid dynamics. These models include a variety of model parameters like force, gravity, propeller characteristics and air density. It is not feasible to examine the necessary model parameters to replicate a particular drone due to these parameters, but the suggested model can simulate any generic drone. An AI-based approach can be utilized to model drone dynamics simply compared to the Newtonian and fluid dynamics methodologies.

It involves an advanced AI model trained on huge datasets spanning multiple real-world flight scenarios. The datasets comprised a broad range of maneuvers, including the eight, circular, and lazy-eight patterns chosen to represent a variety of types of drone motions. Multiple approaches were employed in model creation, including multi-output regression, support vector machine, neural network, and convolutional neural network (CNN). Among these, the CNN model displayed the highest accuracy, achieving 78%. The quantitative validation approach was accomplished by comparing predicted maneuvers versus real-world drone maneuvers. Future work involves further development of a CNN-based drone dynamic model combined with a virtual reality environment.

Keywords:

Artificial Intelligence, Conventional Neural Networks, Drone Dynamics, Simulation, Three-Dimensional Space

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

UAV	– Unmanned Aerial Vehicle
VTOL	- Vertical Takeoff and Landing
VANET	- Vehicular Ad-hoc Networks
GUI	– Graphical User Interface
UAVNet	- UAV networks
FPV	- First Person View
MAV	- Micro Aerial Vehicles
RC	- Radio Controller
FDM	- The Flight Dynamic
AIR	- Aerial Informatics and Robotics
ML	- Machine Learning
MAV	- Micro Air Vehicle
RTH	– Return to Home
CSV	- Comma Separated Values
OSD	- On-Screen Display
RC	- Remote Controller
FPV	- First-person view
MSE	- Mean Squared Error
NN	- Neural Networks
CNN	- Convolutional Neural Networks
RMSE	- Roort Squared Error
MAE	- Mean Absolute Error

CHAPTER 1: INTRODUCTION

1.1 Motivation

The goal of this research is to create an AI-based method for simulating drone dynamics in three dimensions. This is an important area of study because drones are becoming increasingly popular for various applications, including surveillance, delivery, agriculture, and search missions. However, modeling drone dynamics in virtual three dimensions is a difficult endeavor that requires more sophisticated models. The accuracy and effectiveness of drone simulations can be increased by creating an AI-based method, which can help cut down on the cost and time needed for testing and development by using simple model parameters. This strategy may also make it possible to simulate drone behavior in a variety of settings more accurately, enhancing drone reliability and safety. It will aim to improve drone simulators' effectiveness and accuracy by using an AI-based method, which helps reduce the expense and time needed for testing and development of drones.

1.2 Statement of the problem

Unmanned Aerial Vehicles (UAVs), regularly known as drones, have become popular over the past two decades because of their numerous uses in surveillance, disaster management(Geoawesomeness n.d.,2023), pollution monitoring, cinematography, archaeology (Drones Transform Archaeology,2023), pizza delivery and military reconnaissance (Drones and UAV News, 2023). The precise date of the first successful UAV flight is still unknown. Many sources indicate that the creation of UAVs can be traced back to 1849. The Austrians employed a fleet of 200 unmanned balloons in a tactical assault on the city of Venice during this period (Goodman et al., 2015). Key aspects pushing this progress are the built-in high maneuverability of UAVs, their vertical takeoff and landing (VTOL) capabilities, and their steady hovering skills (Fernando et al., 2013). These versatile drones fly without human pilots, altering industries by providing aerial perspectives in risky or remote situations. In terms of efficiency and innovation, drones are invaluable for filmmaking, protecting the environment, and disaster assistance. They improve reconnaissance and surveillance in the military, altering modern fighting. The development of drones begins a revolutionary period that alters the potential of hovering technology (Mairaj *et al.*, 2019a). On the other hand, the UAV market has rapidly grown during the previous five years. Figure 1 shows forecasted revenue through 2025 as well as the global commercial UAV revenue from 2016 to 2023 (Worldwide commercial drone market Statista, 2023).

The demand for knowledgeable and experienced human resources is critical as drone technology develops. For this technology to be used effectively, the workforce must possess both theoretical knowledge and practical abilities. Due to this, comprehensive training programs are necessary to cultivate a proficient workforce. (Ribeiro *et al.*, 2021) .

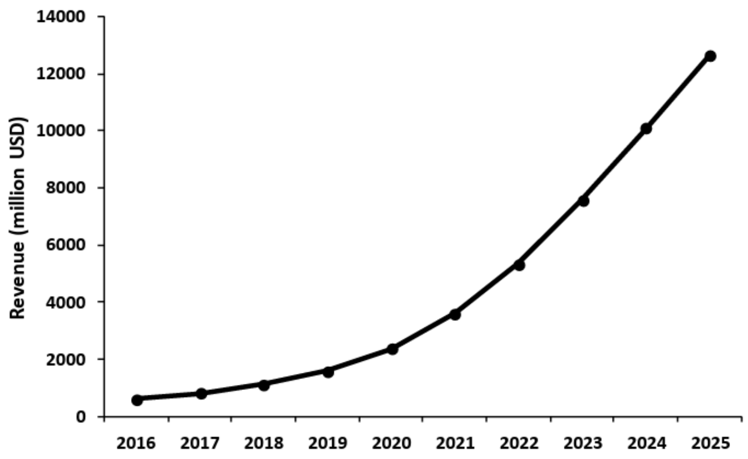


Figure 1: Commercial UAV revenue worldwide from 2016 to 2025

The use of drone simulators is a key component of this training program. These simulation programs provide a safe setting for aspiring drone pilots to improve their abilities and get comfortable with the specifics of flying drones. It is impossible to overstate the value of drone simulation, particularly since some drone activities are very dangerous. Before real-world deployment, professionals can evaluate and improve their abilities by simulating drone operations. One essential element that sticks out is drone simulation, which offers drone pilots significant training and hands-on experience. Additionally, by using simulators, costs and training durations can be minimized. Also, the risks and damages associated with improper operations can be decreased. There exist numerous justifications for employing simulation instead of conducting a real-world experiment on a given system. Three primary factors contribute to this phenomenon (Fritzson, 2004).

- It is expensive to perform experiments on real systems.
- It is dangerous. The pilot can practice a dangerous maneuver before performing it on the plane.
- The system may not yet exist, the model will act as a prototype that is evaluated and tested.

Other positive features that simulation provides are:

- Variables not accessible in the real system can be observed in a simulation.
- It is easy to use and modify models to change parameters and perform new simulations.
- With system design optimization many variants can be evaluated.

Tools like the DJI drone simulator, Zephyr, and FPV Air 2 drone simulators can simulate real-time drone behavior. These simulators only imitate drones sold by manufacturers. On the other hand, it is difficult to simulate drone dynamics using simulation models based on fluid and Newtonian dynamics. It requires applying specific knowledge, conducting several trials under ideal environmental circumstances, analyzing practical challenges, and assessing the many parameters of the customized drone.

Compared to conventional techniques like fluid and Newtonian dynamics models, machine learning, a subfield of artificial intelligence, offers a simplified strategy for simulating drones. Machine learning uses fewer parameters than these traditional methods for accurately imitating drone behavior. When compared to the difficulties inherent in fluid and Newtonian dynamics models, machine learning models are a more effective option for generating realistic and accurate drone simulations. Therefore, in this dissertation proposed a method for the simulation of drone dynamics using machine learning.

1.3 Research Aims and Objectives

1.3.1 Aim

This study aims to develop an AI-based approach for simulating drone dynamics in three-dimensional space.

A simulation method known as an AI-based model makes use of artificial intelligence tools like machine learning and neural networks to faithfully imitate the motions of unmanned aerial vehicles. By incorporating inputs from the drone pilot via the drone's radio controller as well as the drone's shifting position and orientation over time, this AI-based model seeks to imitate UAV behaviors. The justification for using an AI-based model comes from its possible benefits:

- Data-Driven Learning: AI models can learn from information gathered during real world UAV operations, capturing complex correlations that could be difficult to represent mathematically.
- Adaptability: AI-based models can adjust to various circumstances and conditions, considering changes that occur in real-time and providing a more accurate picture of UAV behavior.
- Automation: These models improve accuracy and decrease the amount of time

required for parameter adjustments by automating the process.

- Generalization: AI-based models are more adaptable and relevant to many drone models because they may be taught in a variety of UAV kinds and settings.
- Validation against real-world data allows for a more comprehensive assessment of the correctness and efficiency of AI models.

Given these benefits, the AI-based simulation model put forward in this study attempts to overcome the drawbacks of non-AI methods like mathematical functions. By utilizing artificial intelligence's capabilities, it intends to faithfully imitate UAV motions while supplying a more flexible, automated, and data-driven framework for studying UAV behavior.

1.3.2 Objectives

The following are the specific objectives of the study:

- To conduct a comprehensive analysis of the current state-of-the-art techniques used for simulating UAV movements.
- To determine the existing difficulties and limitations associated with simulating drone dynamics using traditional methods, as well as the possible advantages of incorporating AI technologies.
- To gather necessary data on real flight scenarios for building the model.
- To develop an AI-based model that can accurately simulate the movements of UAVs.
- To validate the AI-based simulation model by comparing its predictions with real-world drone flight data.

1.4 Scope

In machine learning-based drone simulation, the quality and quantity of training data play a critical role in ensuring accurate model predictions. The training data used for drone simulation may include various flight-related data from real drones, such as sensor measurements, control inputs, and environmental conditions. Furthermore, the selection of appropriate model parameters is crucial to the accuracy and effectiveness of the machine learning-based drone simulation. Factors such as drone weight, battery capacity, motor specifications, propeller size, and control system parameters are a few examples. Additionally, environmental factors such as wind speed, temperature, and humidity can significantly affect the drone's flight characteristics and should be considered when selecting model parameters for the simulation. The few model parameters are as follows.

- An important parameter is the drone's geometry, which includes its size, shape, and mass distribution. It has an impact on the aerodynamic forces and moments the drone experiences while in flight.
- Another crucial factor is the propulsion system, which includes the motor, propellers, and batteries. It has an impact on the drone's energy usage and thrust generation.
- When simulating the flight dynamics of the drone, aerodynamic coefficients such as lift, drag, and moment coefficients are essential inputs.
- When simulating the drone's flight dynamics, it is important to consider the wind conditions, including wind speed and direction. The drone's aerodynamic forces and moments are influenced by the wind, which might lead it to deviate from its planned trajectory.
- When simulating the drone's flight dynamics, fluid properties like density, viscosity, and temperature are essential inputs. These characteristics have an impact on the aerodynamic moments and forces that the drone experiences while in flight.

Without considering all these parameters, this study focuses exclusively on the input data from the radio controller of the drone and its position as model parameters. The battery level of the drone may influence its performance and potentially affect the model's outcomes. However, to minimize the impact of this variable, data collection for this study was conducted exclusively on fully charged batteries with a battery level ranging from 95% to 100%.

Furthermore, the influence of real-world wind flows, which may introduce turbulence and complexity to the experimental setup, was considered. To mitigate these effects, all experiments and tests were carried out in a calm outdoor environment with negligible wind influence. This approach ensures that the study's results are not confounded by extraneous variables and enhances the rigor and validity of the research outcomes.

1.5 Research Gap

Hwangbo developed a neural approach to simulate drone dynamics in a restricted environment measuring around 2m x 2m x 2m (Hwangbo *et al.*, 2017). Although effective in simulated and real-world settings, this technique did not properly anticipate drone dynamics to a reasonable degree. Sandaruwan conducted more research on the user's perception of realism in a machine learning-driven drone simulator (Sandaruwan *et al.*, n.d.). A neural network model was employed in this strategy. Trained on data from eight movement patterns on a two-dimensional axis, this model initially predicted the drone's position accurately but deviated more as time progressed. Prediction errors were greatly impacted by the RcRudder radio controller input, which controls yaw motion, resulting in the accuracy of the model being 75%.

This study suggests creating an enhanced model to simulate drone dynamics to overcome the current limitations and inaccuracies. The model used data gathered from three-dimensional maneuvering patterns and tested against actual drone maneuvering flight data.

1.6 Contribution to Computer Science Domain

- The researcher collected a large dataset consisting of various three-dimensional drone flight maneuvers, amounting to about one hundred thousand (10^5) data points. This expanded dataset has great potential for future research and progress in drone technology and simulation.
- This work introduces a novel approach for creating a dynamic drone simulation model using machine learning techniques.
- By presenting a basic technique to develop machine learning-based dynamic drone simulation models, the study tackles the difficulty of evaluating various model parameters, especially for customized drones. This streamlined process improves the accessibility and applicability of drone simulation technology.

1.7 Research Challenges

The major challenges faced during the study are as follows:

- Collecting a massive dataset of drone flight data.
- Finding skilled drone pilots capable of executing different movement patterns.
- Building a robust machine learning-based drone dynamic model.
- Access to a high-performance computer for processing large datasets.

1.8 Research Questions

- Q1: What are the techniques used to simulate UAV movements?
- Q2: What are the strengths and weaknesses of each technique in simulating UAV dynamics?
- Q3: What are the major challenges and limitations of non-AI-based methods when simulating drone dynamics?
- Q4: How could AI technologies assist in addressing these difficulties and improve simulation accuracy and efficiency?
- Q5: What factors and variables are necessary for accurately simulating real UAV flight scenarios?
- Q6: Which AI techniques and algorithms are most appropriate for creating a model to simulate UAV movements?
- Q7: How to validate the developed AI-based drone dynamics simulation model?

1.9 Structure of the Thesis

This chapter provides a comprehensive overview of the research background, scope, requirements, and objectives. This chapter briefly outlines the challenges, limitations of present systems, and other relevant factors.

The thesis documents the research work with five main chapters. The literature review, which is the second chapter, provides a background analysis of the research topic by citing previously published research materials, papers, books, journals, internet articles, etc. An in-depth literature review was performed to identify the previous methodologies followed in similar research studies. The methodology which is the third chapter details the research approach and describes the technical aspects of the dataset, characteristics, design, and modeling. The fourth presents the evaluation of the results obtained through performing the research methodology. The study effort is concluded in the last chapter, which summarizes the findings as well as recommendations for future research and the project's limits.

CHAPTER 2: LITERATURE REVIEW

The literature review chapter will summarize the related information on this study. This chapter includes an in-depth analysis of the research, along with a critical assessment of related earlier publications. Also, this chapter includes an algorithmic study, an assessment of present approaches, and the methodologies that are in existence.

2.1 Literature Review

The process of making a drone model involves people from different fields. People are working together on it from different fields, which makes it an interesting topic that needs a thorough description so that everyone can understand it. Here are the main points this literature study makes.

1. Aims to examine the historical development of drone simulators, offering readers an overview of the beginnings of the UAV simulation industry and its subsequent expansion in recent years.
2. Provide a comprehensive examination of the requirements for drone simulators, aiming to enhance readers' understanding of these simulators from multiple viewpoints, including aerodynamics, game engineering, network engineering, and software development.
3. Provide an in-depth analysis of many simulators across various fields, along with a comparative evaluation.

2.2 UAV types and models

Rotorcraft drones, another name for rotary-wing drones, are a class of UAVs that use one or more rotor blades to rotate to fly. There are different types of rotary-wind drones. Following are a few instances of UAV and the various UAV types are depicted in Figure 2.

- Multirotor
    - The most popular kind of multirotor drones are quadcopters, which have four rotors.
    - Hexacopter and octocopter drones with six and eight rotors respectively offer more stability.
    - Tri copters: Drones with three rotors, which are less common but offer unique flight characteristics.

- Helicopters
    - Single-rotor helicopters: These unmanned aerial vehicles are like conventional helicopters, including a single, large rotor at the top and a stabilizing tail rotor.

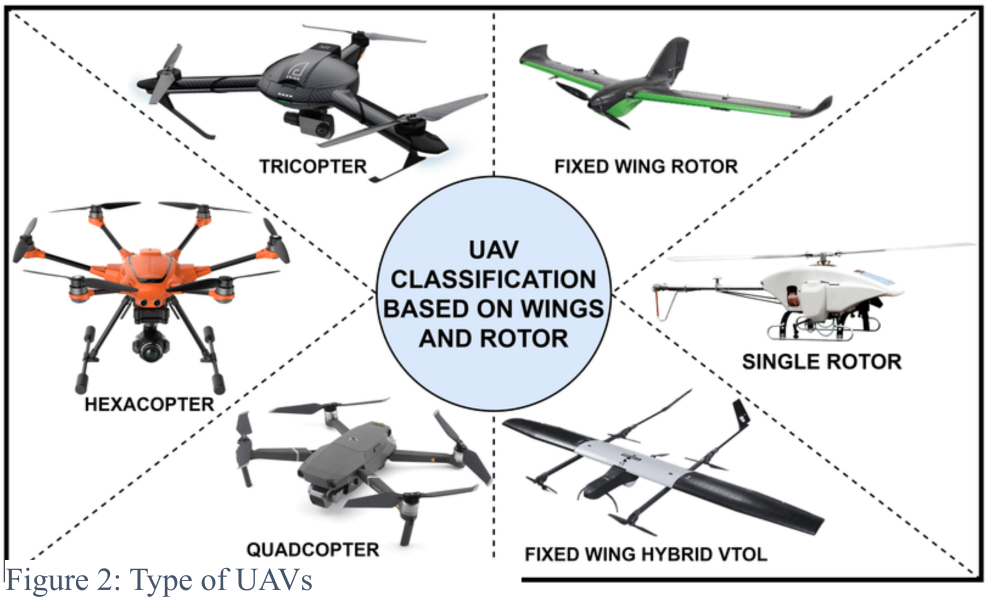


Figure 2: Type of UAVs

An effective simulator should include the capability to deal with a diverse range of drone types and models. The classification of UAVs lacks a universally accepted standard, nevertheless, many experts have proposed categorizing them according to various factors including size, range, and configuration. Table 1 presents a comprehensive summary of several types of UAVs, classified according to their size, endurance, and configuration (The Business of Drones, 2014).,(GEOG 892: Unmanned Aerial Systems, 2023)

Table 1 : Classification of UAVs

Category	*Subtype(s)*	*Properties*
Size	Large-scale UAVs: Fly either autonomously or through an operator.	Size (10m)
	Medium-scale UAVs: Heavy to be carried by a person; however, lighter than the large-scale UAVs.	Size (5m)
	Small-scale UAVs: Mostly fixed wing.	Size (50cm-2m), Weight (2-10kg)
	Mini UAVs: Most used UAVs, cheap, easy to maintain, and smaller in size.	Size (1-1.4m)
	Micro/Nano UAVs: Used for surveillance and academic research.	Weight (100g)
	Very low-cost and close-range UAVs: Include Mini, Nano, and Micro UAVs.	Endurance (0.5-1 hour), Range (5km)
	Close Range UAVs: Primarily used in surveillance, recon, traffic monitoring, etc.	Endurance (1-6 hours), Range (100 km)

Endurance	Mid-Range UAVs: Very high speed, used for surveillance, and gathering meteorological data.	Endurance (24 hours), Range (650km)
	Long-Endurance UAVs: Used for long-range surveillance missions.	Endurance (36 hours) and Maximum Height (9100m)
Configuration	Fixed wing: Capable of flying over longer distances, carrying a heavy payload with low power consumption. Modeling and structure are simpler than rotary wings.	Need a runway for takeoff and landing.
	Rotary wing: Quite maneuverable, has less endurance and is noisy. Further classified into the single rotor, quadcopter, hexacopter, and octa copter.	The biggest advantage is the ability to land and take off vertically.
	Flapping wing: Flight dynamics is bio-inspired by insects and birds and is more complicated than other UAVs because of the nature of flight.	Bridge the gap between fixed-wing and rotary-wing flights. Less noisy compared to rotary wing UAVs.

2.3 UAV Simulator Requirements

UAV simulators provide three primary purposes, including the assessment of novel technologies, cost-effective training, and research and development (R&D) endeavors(Mairaj et al., 2019). Modeling and simulating a UAV is not an easy task, it requires the calculation of many parameters. Unlike the mobility of Vehicular Ad-hoc Networks (VANETs), the movement of a UAV is not confined to 2D space (Capello *et al.*, 2009a). The use of various aerodynamic coefficients in the development of a three-dimensional mobility model for UAVs presents extra challenges(Vogeltanz, 2016).

Also, when constructing a simulator, the designers and engineers need to consider the aerodynamic characteristics, three-dimensional flight capabilities, and network communication aspects of drones. An effective UAV simulator requires the addition of several key attributes: user-friendliness, compatibility with commonly available technology, reduced complexity, and an easy-to-use graphical user interface (GUI). A detailed description of some of the important requirements for a good UAV simulator is provided below (Capello et al., 2009).

- **Flight dynamics model**: Understanding the aerodynamic forces produced during flight is crucial for efficient aircraft design and pilot control. Flight simulator designers can apply well-known equations related to aerodynamics to the dynamics modeling of drones, as the airfoil generates lift for flight.

- **System model**: The reliability of a drone simulator is dependent upon the presence of an accurate mathematical representation of the UAV system. The models that have been developed can be categorized into two main types: those that are grounded on an in-depth understanding of the underlying physics governing the entire system, and those that are generated from the utilization of dependable real-world data collected from UAVs. The objective is to replicate the actions of an actual UAV within a simulation environment, ensuring that the input-output relationships correspond between the physical vehicle and its simulated equivalent. In other words, if a specific set of inputs produces a particular set of outputs from the drone, the same inputs should result in identical outputs in the simulator.
- **Graphical model**: The graphical representations used by UAVs are characterized by a restricted color scheme, lack of shading, the use of vector graphics, integration of alphanumeric text, and incorporation of a small number of relevant symbols. The refresh rate of the displays should be a minimum of 20 times per second, and the display functions are implemented through the utilization of 2D graphics methods. The display is typically represented using a Cartesian coordinate system, wherein the horizontal axis is denoted as the x-axis and the vertical axis is denoted as the y-axis. Sometimes, differences in convention occur depending on whether the display mode is landscape or portrait. Conventions of this nature are typically established within a graphics library, so enabling users to create graphics software that is not constrained by specific hardware restrictions, as depicted in figure 3.

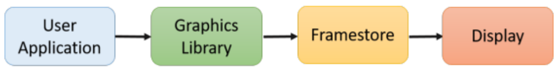


Figure 3: The Graphics Pipeline

- **Control system**: Flight simulators consider control systems as mathematical transfer functions. Consequently, the designer of the simulator must include these transfer functions in algorithmic format to ensure accurate modeling of the input-output relationship of the system. The system engineers considered the time response, frequency response, and transfer function as being equivalent and interchangeable (Allerton, n.d.). In this project, researchers built robust system model by combing control system.
- **Flight route identification:** The aircraft's navigation system facilitates the determination of both present and future positions for its users. The navigation system

collaborates with several additional subsystems to facilitate various functionalities, including the identification of flight routes, avoidance of collisions, planning of optimal paths, and assignment of air corridors. However, in the flight simulator, there are no signals or sensors, and the simulator designers must face tough challenges.

- **Application-specific requirements:** UAV simulators may have extra criteria that are specific to their intended applications. An effective UAVNet (UAV networks) simulator may necessitate the incorporation of the communication component of drones. UAV is considered as a node within the network, and communication is established through the utilization of UAVNet specific networking protocols. In the same way, a simulator used for the purpose of cybersecurity research may necessitate further modules related to potential attack scenarios. Simulators may necessitate the installation of a module focused on comprehensive data or result analysis. The speed of the simulator should be adjustable to allow the users to extract essential details (Javaid *et al.*, 2015). In addition, it is important that the simulations are capable of functioning in several circumstances, each with distinct mobility models. In the context of simulations, particularly in the domains of flight or vehicle simulations, a mobility model refers to a mathematical or computational representation of the movement patterns displayed by objects or individuals inside a specified environment as time progresses. The concept defines the patterns, behaviors, and qualities related to the mobility of things, hence enabling simulation designers to replicate realistic instances of movement. Furthermore, to facilitate the modeling process, many assumptions are used. These assumptions include considering UAV as rigid, assuming the Earth as the inertial reference frame, and assuming a constant aircraft mass throughout the simulation (Vogeltanz, 2016).

2.4 Background and Related work

In 1910, pilots were trained on the first flying simulator that didn't depend on wind. Figure 4 shows that it was made up of two halves of a barrel that were mounted and moved by hand by a pilot sitting in the upper half of the barrel. The pilots had to line up the reference bar of the simulator that shows the sky (X-Plane.Org Forum,2011).

Figure 4: Early Manual Flight Training Devices

Also, the First World War pushed people to make better machines for teaching pilots to fly. Control and recording equipment for electricity were put in these models. Later, these flight simulators got better to the point where they were supposed to respond to the pilot's control inputs like a real plane would. This was made possible by installing electrical and mechanical equipment. Between 1927 and 1930, Edwin Link worked in Binghamton, New York, USA, to make one of the most famous and successful flight simulators. At that time, it was thought to be the best and most innovative way to train, and it became the most famous and preferred way to train pilots during World War II (Ray L, n.d.) .

During World War II, electronic models became popular because computers could now solve the equations for flight dynamics. During the 1950s, Curtiss Wright made the first full simulator for an airline called Pan America. This simulator, which has sounds and images of planes, was bought by US Airlines for $3 million. It is thought to be one of the first modern flight simulators. In the 1960s, movements along the pitch, roll, and yaw axes were added to simulators. In the 1970s, very simple computer graphics models with a single moving white dot on a black background came out. Over time, the screens changed into more detailed and completely dynamic ones. The Microsoft Flight Simulator was one of the biggest successes in this field (Flight Simulator X For Pilots_ Real World Training, 2008) . In its early years, it was a collection of 3D flight simulation graphics pieces written by Bruce Artwick in 1976.

There are two primary categories of 3D drone simulators:

- To help drone-enthusiasts practice First Person View (FPV)
- To refine the handling of camera drones to capture good quality videos and photos.

Unmanned Aerial Vehicle (UAV) simulators can be categorized based on their adaptability with either outdoor or indoor environments, research, or entertainment, single or network of UAVs, and purely software/hardware or hybrids (Jahan *et al.*, 2015).

Simulators are essential tools for drone enthusiasts, design engineers, and researchers, since they serve a variety of purposes. One such use is to facilitate the training and improvement of videography abilities, particularly in the context of FPV. Furthermore, considering the specific field of application, the inclusion of graphic detail in a simulator is vital to achieve a truly authentic experience and ensure the technical integrity of the aerodynamic design. The precise aerodynamic configuration of an Unmanned Aerial Vehicle (UAV) is of utmost importance across all shapes, sizes, and designs, as any deviation from optimal design principles may result in a catastrophic failure. Hence, it is imperative to accurately construct a model of it within a simulation environment.

There are two primary methods for simulating drone dynamics: traditional techniques that depend on physics and mathematics, and modern AI-based technology. Traditional approaches are based on established physical and mathematical principles, offering a solid foundation but frequently requiring exact input parameters and assumption. On the other hand, AI-based simulations make use of modern techniques like deep learning and reinforcement learning, which allow the simulations to pick up complex patterns and adjust to a variety of situations.

2.4.1 Math model of UAV with fixed wing aerodynamic

Exact methods can be used to solve some equations, but numerical estimation approaches are required for more complex equations. A UAV simulation method using numerical techniques was introduced by David Orbea, Jessica Moposita, and Jessica Moposita. They include numerical approximations in their mathematical model, which includes several Single Input Single Output (SISO) systems. These systems have relationships between altitude, pitch, roll, yaw angles as input parameters and yaw speed x, y, and z axis speeds as output parameters (Orbea *et al.*, 2017) . Identifying the complicated relationships between the aircraft's speed on the X-axis and the pitch angle, its speed on the Y-axis and the roll angle, and its speed on the Z-axis and the yaw angle are the main objectives of this hard mathematical analysis. Figure 5 shows the aircraft's angles and orientations. These connections are carefully defined with the help of precise mathematical transfer functions. Data collection involved executing a pre-planned flight, where the aircraft followed a specific low-speed rectangular path. Additionally, controlled fixed conditions were used to determine the initial values that are important to these simulations. This methodological decision was made with the purpose of reducing the computing calculation that comes with figuring out these complex transfer functions.

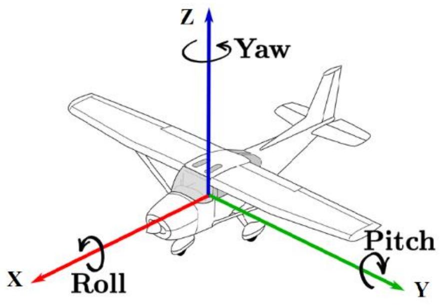


Figure 5: Pitch, Yaw and Roll Angles of an Aircraft

2.4.2 Modelling Quadrotor Dynamics

Using the Newton-Euler method, Fernando H.C.T.E., De S Silva A.T.A., and De Zoysa M.D.C. have provided a different approach for simulating quadrotor dynamics (Fernando *et al.*, 2013). The model predicts the effects of the forces and torques generated by the four propellers on the quadrotor motion. Four control variables were chosen for this method to match the four fundamental movements of roll, pitch, yaw, and vertical movement and considered the three reference frames for modeling. A global position perspective is provided by the Earth's reference frame, which is fixed to the earth. The quadrotor is connected to the body's reference frame, which describes the body's internal motion. A hybrid reference frame, which integrates the frames of the Earth and the body, is essential for understanding and controlling quadrotor movements. Kinematics of rigid body is given by,

$$i = J_\theta v \tag{1}$$

where i and v are the generalized velocity vectors regarding the Earth's frame and the body frame, respectively. J_θ is the generalized matrix that transfers velocities of the body frame to Earth's frame. The dynamics of the quadrotor is given by the following Newton Euler equation.

$$\begin{bmatrix} \boldsymbol{m I_{3\times 3}} & \boldsymbol{0_{3\times 3}} \\ \boldsymbol{0_{3\times 3}} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{V^B}} \\ \dot{\boldsymbol{\omega^B}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\omega^B} \times (\boldsymbol{m V^B}) \\ \boldsymbol{\omega^B} \times (\boldsymbol{I \omega^B}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{F^B} \\ \boldsymbol{\tau^B} \end{bmatrix} \tag{2}$$

Where m is the mass of the quadrotor, I is the inertia tensor, $I_{3\times 3}$ is the 3x3 unity matrix, w^B and v^B are the angular and linear velocities of the quadrotor with respect to the body frame. F^B and τ^B are the force and torque vectors. Using MATLAB Simulink, a quadrotor simulator was created based on the model, allowing for the development, and testing of several control algorithms. However, the gyroscopic effects, wind resistance, propeller dynamics, and crosscoupling effects of the spinning motors (one type of movement might affect its other movements) were not considered in this work.

2.4.3 Realistic Virtual Simulator for UAV

Javier Maldonado and Manuel Eduardo suggested another simulator to do dynamic simulations, this method first obtains a comprehensive dynamic model of the UAV using a quadcopter. Next, using Unity 3D, a virtual three-dimensional representation of the quadrotor and its operational environment is created. Then a virtual quadrotor simulation is created by connecting the model, its dynamic simulation, and the virtual elements (Mora-Soto *et al.*, 2021). Six generalized coordinates can be used to describe the position and orientation of the quadrotor at any given time, as shown in figure 6. The quadrotor navigation schematic diagram shown in this figure 6 is divided into two coordinate systems: the mobile quadrotor coordinate system {B} and the fixed or universal coordinate system {I}, which is used as a foundation for these generalized coordinates. The system {B} attached to the frame of this UAV from the mass center, which is used to analyze the quadrotor body. In this way, the coordinates in $\varepsilon = [x, y, z]^T$ measure the position of the mass center concerning the fixed coordinate system {I}. Regarding the variables in $\eta = [\phi, \sigma, \psi]^T$,they describe the orientation of the quadrotor in terms of a consecutive sequence of Euler's angle rotations.

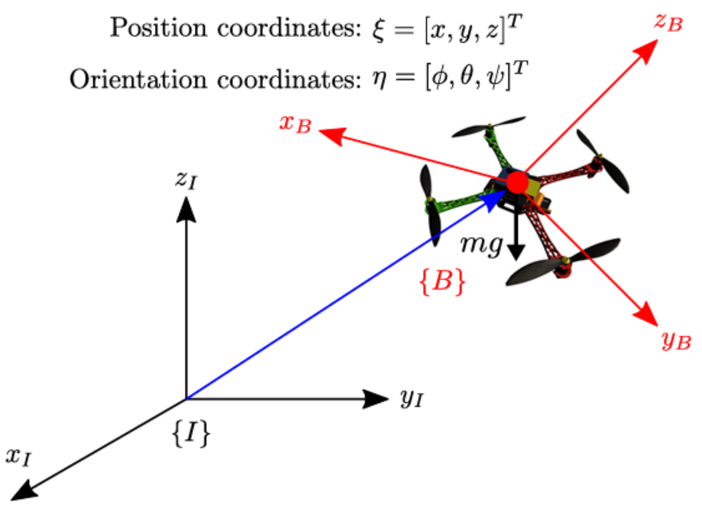

Position coordinates: $\xi = [x, y, z]^T$
Orientation coordinates: $\eta = [\phi, \theta, \psi]^T$
z_B
x_B
z_I
mg
$\{B\}$
y_B
y_I
$\{I\}$
x_I

Figure 6: The Quadrotor Navigation Scheme

This coordinator system is used to generate equations for quadrotor dynamics modeling. The framework is complicated because it has a lot of model parameters. Below are some examples.

- UAV model: To carry out a dynamic simulation within this framework, the UAV is meticulously designed.

- Virtual environment: A virtual 3D operating environment created in Unity 3D is controlled by using the data from the simulation mentioned above.
- Teleoperation algorithm: An input device and a suggested teleoperation algorithm are used to introduce the interaction with a human operator.
- Meta-heuristic optimization technique: This technique gives the simulation a realistic feel. In this process, the teleoperation algorithm's parameters are optimized using flight data gathered from a real UAV.

The developed simulator can be accessed using grabcad.com web site. (Quadcopter SM450 | 3D CAD Model Library | GrabCAD)

2.4.4 Hexagon Flight Simulator

Interactions take place among the three primary constituents of this simulator, namely the mathematical model software, the GUI based on Lab view, and the rendering engine. The first two components operate on distinct machines.

To effectively present the flight information, it is necessary to utilize a configuration consisting of three LCD displays. These monitors will be allocated for the display of the flight data, the virtual cockpit, and the real-time flight parameters updating. The major components of the system are referred to as the model and the engine. The models are created using Fortran (Programming Language) and consist of three sub-modules:

- Airplane dynamics: The field of airplane dynamics includes the study of the movement and characteristics of an aircraft as it reacts to outside factors, including aerodynamic forces, thrust, and gravitational forces.
- Propulsion system: The choice of propulsion system for a given trip is contingent upon several criteria, including the dimensions of the aircraft, the distance it must travel, the required velocity, and considerations of efficiency.
- Atmospheric system

The graphics engine utilized in this system is developed using the C/C++ programming language. It offers a wide range of camera options that facilitate the rotation of the virtual car within the simulation. Given the intricate nature of maneuverability in Micro Aerial Vehicles (MAVs), it is imperative for operators to possess a comprehensive understanding of the platform. To facilitate this, hexagon incorporates both a joystick and a radio controller (RC) for enhanced control. The User Interface of the RC simulator has been enhanced to incorporate this functionality, hence enhancing its realism and practicality. The pilot has the capability to conduct an initial evaluation of several autopilot parameters, such as the proportional-integral-derivative (PID) gains, to enhance the performance of the platform in real-world scenarios. From an initial uncomplicated scenario, it is feasible to modify the mission parameters while conducting the simulation (Capello *et al.*, 2009b). Figure 7 indicates the hexagon Flight Simulator hardware.

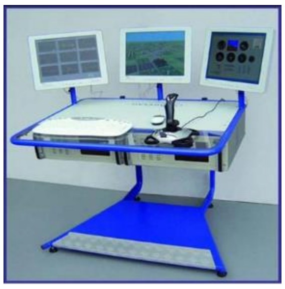


Figure 7: Hexagon Flight Simulator

2.4.5 JSBSim Flight Dynamics Model

A flight dynamics model that is open source and compatible with multiple systems, including Windows and Linux. The aircraft possesses several notable attributes, such as a highly customizable flight control system, aerodynamic models, propulsion mechanisms, landing gear configuration, planetary rotational influences, and engine characteristics. JSBSim has been employed for research purposes in motion-based simulators at the University of Naples in Italy, as well as at the Institute of Flight System Dynamics and the Institute of Aeronautics and Astronautics at RWTH Aachen University in Germany (Berndt, 2004). Here are some potential issues associated with JSBSim.

- Learning Curve: JSBSim can be hard to get the hang of for people who are new to flight dynamics models or simulation software in general. This is because it is so complicated, and you need to understand some basic aerospace engineering concepts.
- Problems with Integration: Adding JSBSim to some simulation environments or frameworks might be hard, based on how well they work together and what the project needs.

2.4.6 Flight Gear Model

The primary motivation behind the conception of the desktop-based Flight Gear project was to develop a sophisticated flight simulator with potential applications in research and education, operator training, engineering design, enthusiast engagement (by facilitating the exploration of preferred flight simulation concepts), and entertainment purposes. (Flight Gear Flight Simulator, 2023).While Flight Gear is primarily designed for aircraft simulation, it also has the capability to simulate mini-UAVs (Vogeltanz and Jašek, 2015). Additionally, the software boasts a vast collection of over 20,000 real-world airports, as well as accurately rendered terrains, rivers, and oceans. The software incorporates a sophisticated sky model that accurately simulates the celestial bodies such as the sun, moon, and stars. This model is capable of dynamically adjusting their positions in accordance with the current time. The networking capabilities of Flight Gear allow for seamless communication and interaction between many instances of Flight Gear, as well as with external devices such as GPS receivers and control modules. The software is compatible with various operating systems, including Windows, Linux, Mac OS-X, FreeBSD, Solaris, and IRIX. The Flight Dynamic (FDM) model is a physics and mathematical model that characterizes the motion of an aircraft in response to external forces and moments, including those exerted by control devices and natural forces. The process of FDM encompasses the creation of a tangible model that accurately represents the physical, inertial, and aerodynamic characteristics of the UAV.

2.4.7 AirSim Model

AirSim is an operating system (O/S) drone simulator that has been developed by Microsoft's Aerial Informatics and Robotics (AIR) team. Its primary purpose is to serve as a valuable tool for doing AI research, with a specific focus on deep learning, computer vision, and reinforcement learning algorithms. The simulator is designed to cater to a wide range of autonomous drones. Initially, the scope of this concept was restricted to the simulation of quadcopters. However, the Autonomous Intelligence Robotics (AIR) initiative aims to incorporate other aerial robotic models that are frequently utilized. The utilization of this simulator has the potential to provide training data that can be employed in the construction of machine learning (ML) models. One notable characteristic of this simulator is its incorporation of protocols, such as Micro Air Vehicle Link (MAVLink), which enhances the authenticity of the simulations (Air Sim Microsoft Research, 2023). Figure 8 indicates the architecture of Air Sim simulator.

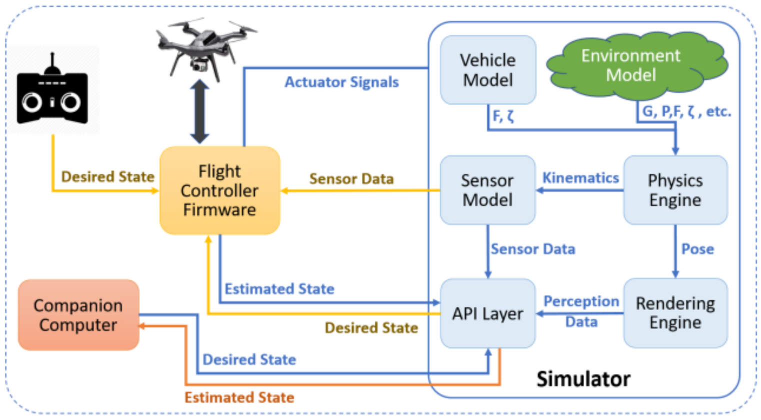


Figure 8: AirSim Architecture

The fundamental components of Air Sim consist of an environment model, a vehicle model, a physics engine, sensor models, an API layer, a rendering interface, and an interface layer. The flight controller receives the intended state and sensor data as inputs, computes the overall estimate of the present state, and generates the actuator control signals necessary to achieve the desired state. In the case of quadrotors, the intended state is determined by the user-specified roll, pitch, and yaw angles. The flight controller utilizes data from the accelerometer and gyroscope to evaluate the present angles and calculates the motor impulses necessary to get the required angles. The simulation facilitates the transmission of sensor data to the flight controller, which subsequently transmits actuator commands to the vehicle model within the simulator. The purpose of the vehicle model is to calculate the forces and torques generated by the simulated actuators. In the context of quadrotors, the propellers are responsible for generating thrust and torques, which are determined based on the computed motor voltages. Similarly, it is possible that additional forces, such as drag, friction, and gravity, may also be at work. The forces are utilized as an input to the physics engine, facilitating the computation of the subsequent kinematic state of bodies within the simulation. The inclusion of many environmental factors such as gravity, air density, air pressure, magnetic field, and GPS location is crucial in determining the kinematic state of bodies within the simulator sensor model. The user or a computer can set the desired state input for the flight controller. The computer is responsible for executing intricate computations, such as the identification of the subsequent desired waypoint and the execution of Simultaneous Localization and Mapping (SLAM) algorithms. The system can efficiently handle a substantial amount of data produced by the sensors, while also being capable of identifying instances of collisions. The purpose of the API layer is to insulate the companion computer from discerning whether it is engaged in a simulation or a real-world experiment. This procedure facilitates the development of algorithms within the simulator, which may subsequently be applied to an actual vehicle without requiring significant alterations (Shah *et al.*, 2017). In this method, aerodynamics simplifications and other considerations can make accuracy problematic. Discrepancies may arise from the sensitivity of input factors. Validation is a challenging process that needs a lot of processing power. Effective usage of AirSim is difficult to learn, and not all cases may be covered by its scope. AirSim is nevertheless useful for the design and analysis of aeroplanes.

2.4.8 DIMAV Flight Simulator

The Dragonfly Inspired Micro Air Vehicle (DIMAV) Flight Simulator is designed with the purpose of evaluating the operational capabilities of a DIMAV under three distinct scenarios: flight (either gliding or forward flight), hovering, and maneuvering. Figure 9 presents a comprehensive depiction of the architectural framework of the DIMAV flight simulator. The mission profile is determined by the user, whilst the autopilot module emulates the autopilot functionality of an aircraft. The purpose of the autopilot system is to execute the mission profile by detecting discrepancies between the anticipated and current conditions of the DIMAV. It achieves this by adjusting the wing articulation to minimize the mistake. The autopilot system employs many sensors, including GPS and accelerometers, to determine the current condition of the drone. This information is then represented in the sensor model block. The "Physics" model encompasses the study of kinetics and behaviour of wing articulation, as well as other relevant physical features. The equations of motion are assessed by including force inputs derived from the aerodynamics and environment modules. The modules encompassed by the delineated area, namely wing dynamics, aerodynamics, and physics, bear relevance to the maneuverability of the DIMAV. The fundamental environmental parameters encompass wind and gravity, while the aerodynamic module addresses the aerodynamic forces associated with specific wing movements and body dynamics (Kok and Chahl, 2015).

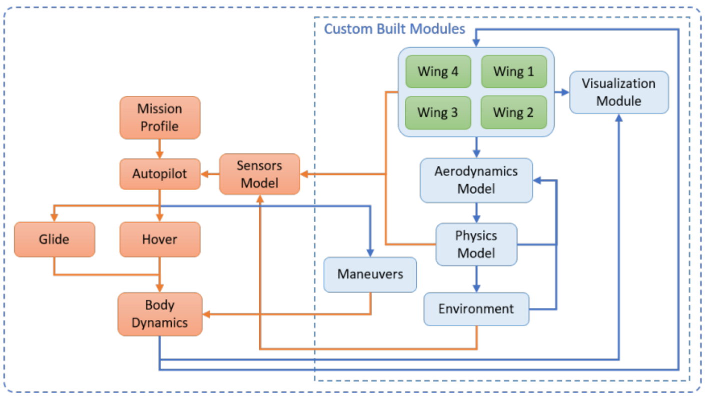


Figure 9 : DIMAV Flight Simulator

2.4.9 Real Drone Simulator

The Real Drone Simulator (Real drone simulator, 2023.) is still in its initial stages of development, showing significant potential for future advancements. There are two practical scenarios in which this can be applied: virtual reality, which is generated by computers, and the real-world using Google Earth. The levels of difficulty and other profile variables, such as the size of the region and the strength of the wind, can be adjusted. Furthermore, the realism of the aerodynamics, handling, and weather situations is evident. The purpose of this simulator is mostly for enjoyment, with the developers intending to provide it without charge. The upcoming upgrades will augment the user experience by introducing a career option that enables players to accumulate virtual currency. This currency may be utilized for the purpose of constructing and upkeeping aircraft, as well as engaging in competitive races against other online participants. The multi-player option facilitates the participation of 4 to 8 individuals simultaneously, enabling them to operate their own drones using various input devices such as transmitters, gamepads, keyboards, or mice linked via USB. Potential future iterations could perhaps prioritize the incorporation of photography and aerial filmmaking endeavors.

2.4.10 DJI Flight Simulator

The DJI(Da-Jiang Innovations) Flight Simulator is another option for simulating commercial drones. This software simulator lets users mimic the actions of several DJI-branded UAVs and controllers. The weather and sunshine settings can be customized, and the simulator offers a variety of 3D scenarios (Dji flight simulator, 2021). Using DJI's advanced flight control technology, DJI Flight Simulator is a professional pilot training program that simulates the feeling of really being in the air. It supports a large selection of DJI drones and is designed for business users. But this simulator is not flexible for addressing specific research and development. The primary problem with DJI Flight Simulators is their proprietary software, which means that the remote operation devices and simulated UAVs have limitations and cannot be expanded (Mora-Soto *et al.*, 2021). Furthermore, it is a very difficult challenge to simulate custom-built drones with these generalized drone dynamics.

UAV simulators encompass a diverse range of genres, exhibiting distinct purposes and features that are contingent upon the specific target audience they want to engage. Universities and research groups employ these technologies to enhance the security of UAVNet, get insights into drone technology, and advance the development of novel UAVNet protocols, among other applications. Novice pilots and gamers utilize flight simulators for the purposes of training and enjoyment, respectively. Moreover, certain simulators are specifically designed for the purpose of engineering design and optimizing the aerodynamics and efficiency of drones. Table 2 provide a potential use of the UAV simulators that were evaluated(Mairaj et al., 2019).

Table 2: Drone simulators and application areas

Simulator	Example Application Domains
Hexagon	Filmmaking, Fire department
JSBSim	Future Drone traffic control Study, Gaming
UAVSim	UAVNet Security, Academic
RealFlight Simulator	Film making, Drone Racing, Precision Agriculture
DIMAV	Testing MAV manoeuvrability
AirSim	Research and Development
Flight Simulator	Game, Survey, Mapping
RotorS	Path Planning Military, Drone Framing

2.4.11 Quadrotor with Reinforcement Learning

The primary objective of this study is to develop a methodology for controlling the movements of a quadrotor, which is a specific form of UAV that includes four rotors, by using a neural network that has been trained using reinforcement learning methodologies(Hwangbo *et al.*, 2017) . The objective of the researchers is to develop a control system in which a neural network is trained to directly establish a correlation between the quadrotor's state and the corresponding actuator commands. This approach aims to eliminate the requirement of a pre-established control structure during the training process. The develop algorithm consists of 2 hidden layers with 64 tanh nodes. The structures are illustrated in Figure 10.

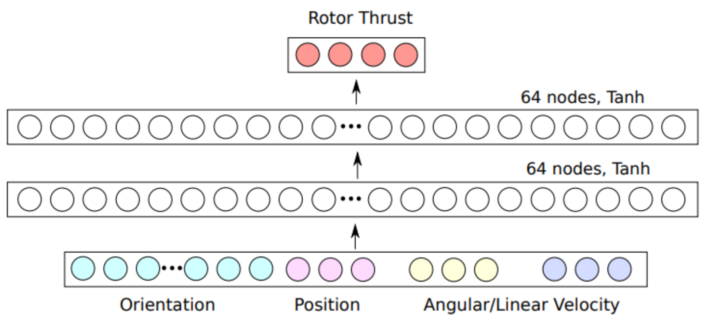


Figure 10 : Neural Networks of Quadrotor

The author’s technique is distinguished by the introduction of a novel learning algorithm, characterized as conservative yet stable, and specifically designed for complex tasks. The authors contend that their approach demonstrates greater applicability in the realm of quadrotor control when compared to previous algorithms. The efficacy of the trained neural network is showcased by the researchers through its application in both simulated environments and practical trials with a physical quadrotor.

The research emphasizes several notable achievements. These include the competent performance of the trained policy in accurately reacting to step responses, effectively stabilizing the quadrotor in mid-air despite difficult initial conditions (such as being thrown upside-down with an initial velocity of 5 m/s) and achieving these outcomes with a minimal computation time of 7 µs per time step. The primary objective of this research is to enhance the domain of quadrotor control through the introduction of an innovative neural network-based methodology that demonstrates efficacy and efficiency in both simulated and real-world contexts. The researchers demonstrated the efficacy of the instructed strategy through both simulated experiments and practical implementation using an actual quadrotor. The discovered policy shows exceptional performance while maintaining computational efficiency. Additionally, neural network policies offer other advantages beyond their versatility. The scope of this experiment is constrained to a confined space with dimensions of approximately 2m x 2m x2m for creating a controlled environment. Due to this small space this method is unable to simulate complex drone moments.

2.4.12 Simulate Drone Dynamics Related to Eight Maneuvering Patten

The data collection technique involved the utilization of the figure of eight maneuvering pattern and the construction of a machine learning based drone dynamic model. The data gathered in this study relates to the eight-maneuvering pattern because of its wide use in different activities all over the domains of aviation, maritime, and ground vehicle operations (Sandaruwan *et al.*, 2019). In this method, an experienced pilot was chosen to do several figure-eight maneuvers using a DJI Phantom drone. The maneuvering testing was carried out in a claim environment with minimal wind conditions. Figure 11 illustrates the actual aerial view of sample drone pilot drill, which has a shape like a figure of eight. The track's actual dimensions are a width of 24 m and a length of 60 m. The recorded data can be classified into two primary categories as follows:

- The drone operator uses the drone Radio Controller (RC) to input commands, such as throttle and rudder values, which determine the drone's location and orientation adjustments.

- The drone's position and orientation are dynamic and vary over time, as do other pertinent sensor readings such as battery level, accelerations, and velocities.

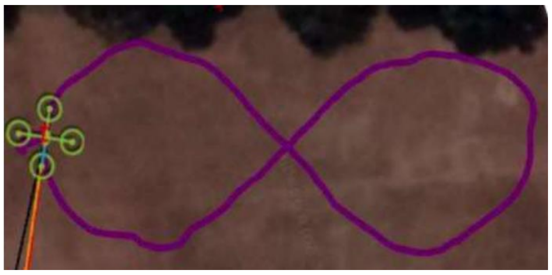


Figure 11: Actual Aerial View of a Sample Drone Pilot Drill

A machine-learning model was constructed utilizing the time stamp, radio controller orders, battery voltage, position, and orientation data of the drone. The machine-learning model that has been created is built upon the principles based on regression and multiple target variables concept. The machine-learning model was provided with a data set of recorded radio controller inputs from a figure of eight trials. The model then predicted the position and orientation of the drone based on this data. The form of the expected path of the drone is depicted in Figure 12. The observed pattern and the expected pattern are considered to have sufficient overall shape similarity. However, this approach was inefficient in addressing accuracy issues caused by data preparation procedures, the little impact of wind, and the choice of machine learning technique.

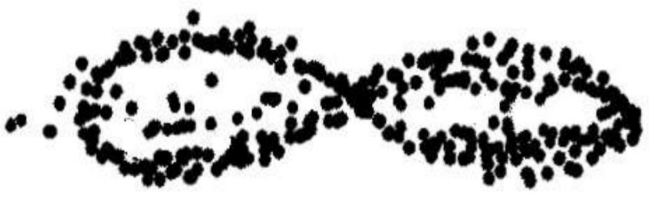


Figure 12 : Shape of the Predicted Path of the Drone

2.4.13 User Perceive Realism of Machine Learning-based Drone Dynamic Simulator

The study focused on creating a drone dynamic model using machine learning techniques in combination with a virtual reality environment. The research also involved validating the physical and behavioural validity of the overall solution as perceived by the user (Sandaruwan et al., 2023). Data regarding drone behavior and the actions performed by the drone pilot were collected by employing a figure-eight manoeuvring pattern with the use of a DJI Phantom drone. The study utilized a dynamic model of a four-rotor drone to determine the relationship between the radio controller inputs from the drone operator and the precise position and orientation of the drone. The development of the drone dynamic model utilized a Neural Network-based approach with Leaky ReLU (Punjani and Abbeel, 2014) activation function in the field of machine learning. The model consists of three layers: the first hidden layer has 56 nodes, the second hidden layer has 112 nodes, and the output layer has one node. All three levels utilize a linear activation function. The loss is computed using the mean squared logarithm error and optimized with the Adam optimizer. Validating the research and development work involves comparing the simulated results with the actual real-world scenario, which is one of the employed validation procedures in this type of work. One way to achieve this is by analyzing the relative locations and orientations of objects at different timestamps. The user test is an essential method to verify the user's impression of the simulated environment. This technique is used to validate the virtual world through the observations of highly experienced drone pilots. Figure 13 depicts the actual locations of the drone and the predicted locations of the drone.

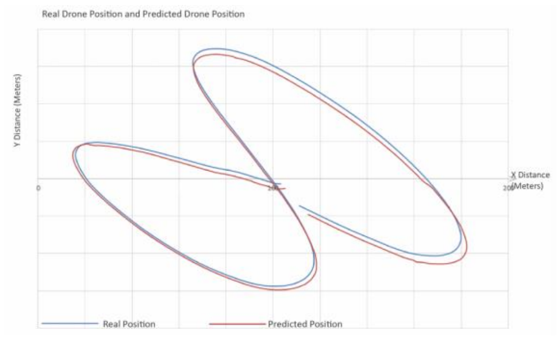


Figure 13 : Actual Position and Predicted Positions of the Drone

Figure 14 depicts the change in the prediction error of the machine learning model over time. The data indicates that the error fluctuates within a range of 0.5 meters. It is necessary to examine the elements that influence this inaccuracy. In cases where rudder deviations are difficult to forecast, then the model's dependence on rudder variation for accurate forecasts could be problematic. Especially for activities with long durations, the observed variation in long-term forecasts over time could be a major drawback of the simulation. The results indicate that the forecasts are diverging over time, and the beginning location predictions are in closer proximity to the real position of the drone. Figures 15 and 16 show the drone's actual and predicted trajectories, respectively. Additionally, the dataset that was obtained only includes eight different maneuver patterns.

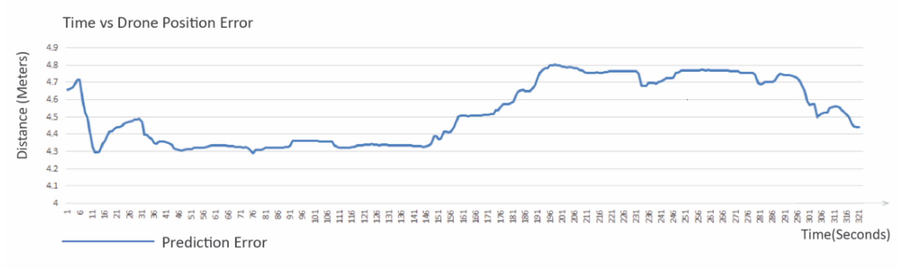


Figure 14: Variation of the Prediction Error of the Machine Learning Model

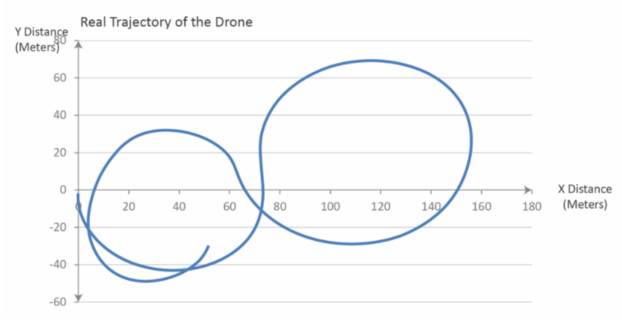


Figure 15: Actual Trajectory of the Drone

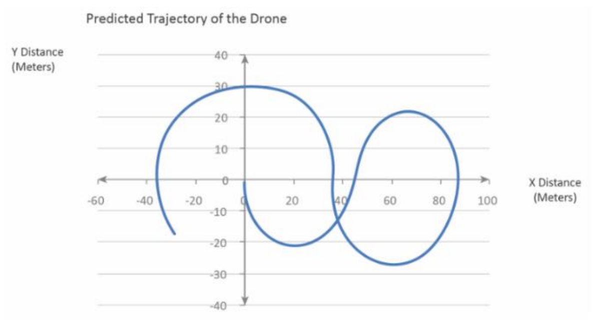


Figure 16 : Predicted Trajectory of the Drone

Table 3 displays the model type and framework utilized for development and the limitations of simulation.

Table 3 :Comparison of model

Simulator	Model Type	Framework	Limitations
Multi-Rotor Flight Simulator	Mathematical transfer function	MavLink Telemetry	• Data collection was conducted using a rectangular flight path that did not imitate complex drone movements. • The wind effect is assumed to be constant.
Quadrotor Flight Simulator	Newton-Euler method	MATLAB Simulink	• Employed multiple model parameters to forecast the dynamics of the drone, such as its form and weight.
Hexagon Flight Simulator	proportional-integral-derivative	-	• Simulators cannot completely substitute real-world experience. • The accuracy of a system relies on various elements such as the stability of the network and integration of third-party software.
AirSim Simulator	Machine Learning Model	• Unreal Engine • Computer Vision and AI Libraries	• Simulations were limited to quadcopters. • require robust hardware for optimal performance.
DJI Flight Simulator	Mathematical models	• Unity 3D • DJI SDK	• Simulate only DJI brand drones. • Commercialized simulator.

			• Running a realistic flight simulator requires a powerful computer.
Simulator with Reinforcement Learning	Reinforcemen t Learning	• TensorFlow	• Data Collection involves small environments. • Mismatch between simulated and real-world dynamics
Simulation with Eight Maneuvering Patten	Multi-output regression model	• MATLAB • AI Libraries	• Model has poor accuracy. • Unbale to predict complex dynamics

2.5 Chapter Summary

This chapter describes the strategies and techniques used in drone simulation and examines the classification of UAVs. Existing drone simulators are limited by factors including commercialization and lack of generality, sometimes neglecting several model parameters. Each part of the literature review aims to provide a thorough summary of current technologies and systems relevant to the research field. There is a lack of research on using AI-based methods to simulate drone dynamics, especially when using three-dimensional flight data. The author of this research aims to investigate the simulation of drone dynamics using neural networks due to the flaws found in present systems. The next chapter will explain in detail the technique behind this study project.

CHAPTER 3: METHODOLOGY

3.1 Research Methodology

This chapter provides an overview of the research and development methodologies selected to continue this research. This chapter discusses the principles, research design, research technique, and the process involved in the research study. The study method includes the solutions to the research aims and objectives outlined in the introduction chapter.

The selection and justification for each selection are listed in table 4 and figure 17 indicates the basic steps of research methodology.

Table 4 : Selection of Research Methodology

Philosophy	**Pragmatism** is a philosophical method that emphasizes practical outcomes and utility. It emphasizes the relevance of what works in practice rather than abstract theories or concepts. By incorporating machine learning technology, the project aims to build a model for simulating custom-made drones, prioritizing utility, and real-world applicability. This pragmatism perspective is obvious in the research's focus on building a useable solution for modeling drone behavior.
Approach	An inductive approach is a form of reasoning that involves progressing from specific observations or examples to bigger generalizations or predictions. The research aligned with an inductive approach. Because the approach involves gathering data on drone behavior and pilot inputs, using this data to train a machine learning model, and testing the accuracy of the simulation model using real-world drone maneuvers flight data.
Strategies	The application of machine learning techniques to construct a drone dynamic model includes **quantitative methods**. Machine learning algorithms rely on quantitative data and mathematical computations to learn patterns and make predictions. Validation against real-world drone maneuvers likely involves quantitative analysis, such as comparing numerical metrics or performance indicators between the simulated and real-world data.

Techniques and other procedures	Research combines a combination of machine learning algorithms, data gathering methods, validation against real-world data, and analysis of validation results to develop and analyze the performance of the machine learning-based drone simulation model.

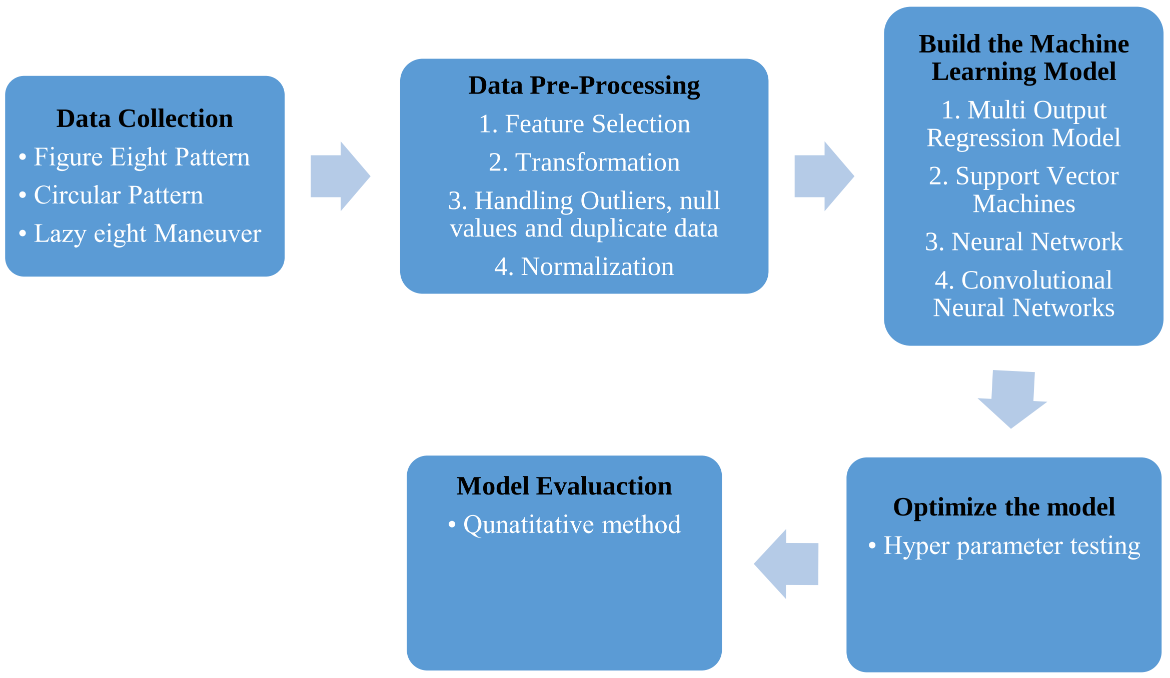


Figure 17: Research Methodology Framework

3.2 Data Collection

Four-rotor drones are a subclass of multirotor systems, and these drones employ four rotors to keep them hovering. A prominent example of these multirotor drones is the regularly used DJI Mavic mini drone developed by the SZ DJI Technology Co. Ltd ("Support for Mavic Mini - DJI", n.d.). The data collection for the present study was conducted via a DJI Mavic Mini drone, as depicted in Figure 18.

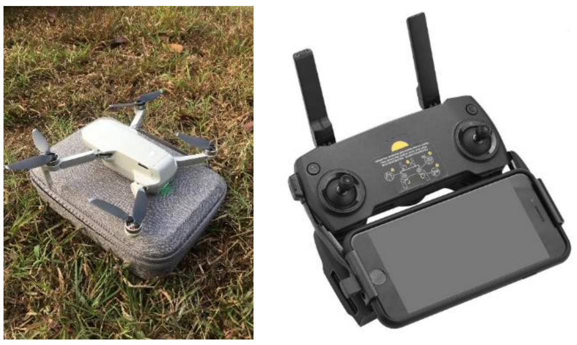


Figure 18 : DJI Mavic Mini Drone and Remote Controller

The drone's position and orientation are determined by the inputs it gets from the radio controller. This study's main goal is to use machine learning techniques to create a dynamic model for a drone with four rotors. Its primary goal is to establish a relationship between the drone's location and orientation changes with the radio controller inputs given by the drone operator. Six degrees of freedom are available to the drone for movement, which includes three rotational and three translational motions. The drone's movement in three dimensions is controlled by the remote controller using four main adjustable variables. The terms throttle, pitch, roll, and yaw refer to these parameters. A radio controller with these important adjustable variables is shown in Figure 19.

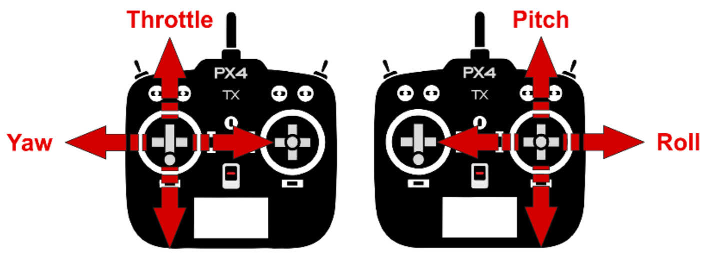


Figure 19 : A Remote Controller with Key Controllable Variables

- Throttle: The throttle governs the vertical movement of the drone, specifically its ascent and descent. Increasing the throttle increases the drone's altitude while decreasing it causes the drone to descend.
- Yaw: Yaw refers to the rotation of the drone around its vertical axis. Rotating the drone clockwise increases its yaw angle and causes it to turn right, while rotating it counterclockwise results in a left turn.
- Pitch: Pitch refers to the forward and backward tilt or rotation of the drone around its lateral axis. Tilting the drone forward increases its pitch angle and causes it to move forward while tilting it backwards results in backward movement.
- Roll: Roll refers to the side-to-side tilt or rotation of the drone around its longitudinal axis. Rolling the drone to the left increases its roll angle and causes it to move leftward while rolling it to the right results in rightward movement.

Horizontal movement is controlled by adjusting the platform's tilt with different motor thrusts, while vertical movement is achieved by changing the total thrust of the motors. Figure 20 shows the basic movements of drones according to the motor thrusts.

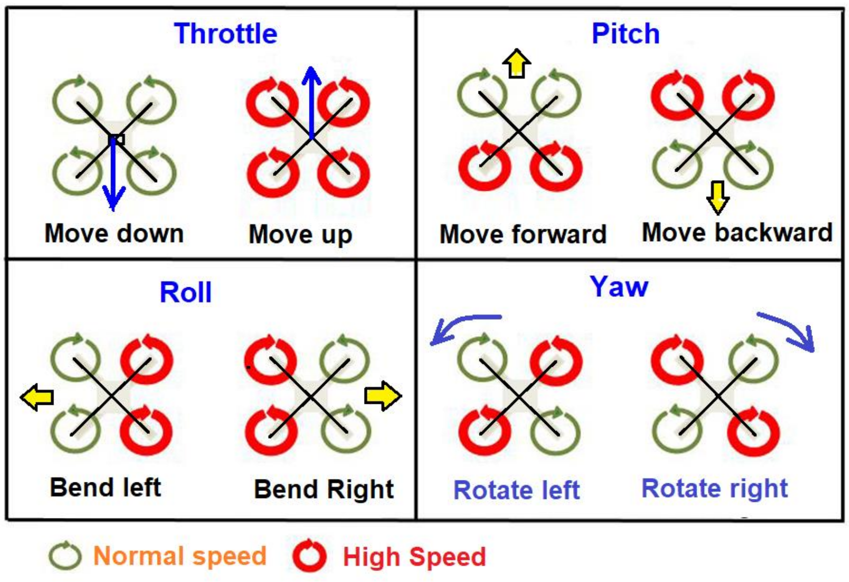


Figure 20 : Four Main Drone Dynamics

Among these primary dynamics, a drone can maneuver through complex motions in three-dimensional space based on the parameters received from the radio controller. The DJI Mavic Mini drone is equipped with a variety of integrated sensors that are intended to guarantee accurate flying and consistent performance. The sensors, such as accelerometers, GPS, and GLONASS, work together to ensure precise location, navigation, and flight stability. Accelerometers help stabilize flying by detecting acceleration forces and sending data to the drone's flight control system. This feature assists the Mavic Mini in maintaining its orientation and performing movements with precision. GPS and GLONASS satellite navigation systems offer accurate location data, allowing for functions like Return-to-Home (RTH) and precise hovering. The Mavic Mini utilizes signals from both GPS and GLONASS satellites to ensure precise navigation in areas with poor or obstructed GPS signals. The Mavic Mini offers a flight log that documents flight position, orientation data, and radio controller input data. The log records 10 data points per second (10Hz), providing detailed insights into the drone's flying patterns and pilot commands. While the Mavic Mini maintains a baseline location accuracy of roughly 0.5 meters using its built-in GPS sensors, customers can boost this accuracy up to 1 centimeter by installing extra sensors. This feature enables enhanced accuracy in tasks like surveying or mapping.

3.2.1 Types of Maneuvering Patterns for Data Collection

The history of driver training for ground vehicles, pilot training for aircraft, and naval training spans several decades, with each sector inventing specialized exercises and exams to enhance abilities and safety. These training techniques have grown over time with technological improvements and have established knowledge bases with verified track records. (Russell Peter ,2002, “the institute of driver education research”) In pilot training, fundamental maneuvers including steep power turns, steep spirals about a point, chandelle turns, lazy eights, and eights-on-pylons are often utilized to build pilots' skills and abilities. These maneuvers need precise control and coordination of the aircraft. Naval training and shipbuilding employ standardized maneuvering tests such as the turning circle maneuver, Z-maneuver, straight spiral maneuver, halting test, and parallel course maneuver. These tests measure ships' mobility and handling characteristics in various settings (ITTC-Recommended Procedures Full Scale Measurements Maneuverability,2002). Most of these training techniques incorporate six degrees of freedom (6DoF) motions, accounting for the dynamic interaction between the vehicle and the operator. The figure-eight maneuvering pattern, or its variations is extensively employed across several areas to imitate real-world circumstances and test essential maneuverability features. Below main maneuvering patterns were selected for collecting drone flight data.

- Figure-Eight Pattern: This pattern involves flying the drone in a figure-eight configuration, which can assist pilots learn coordinated turns, maintaining altitude, and controlling speed. Figure 21 shows the eight-maneuvering pattern.

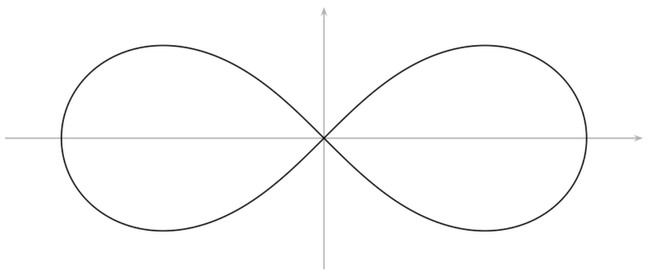


Figure 21 : Figure of Eight Maneuvering Pattern

- Circular Pattern: Flying the drone in a circular path around a fixed point. Figure 22 shows the circular pattern.

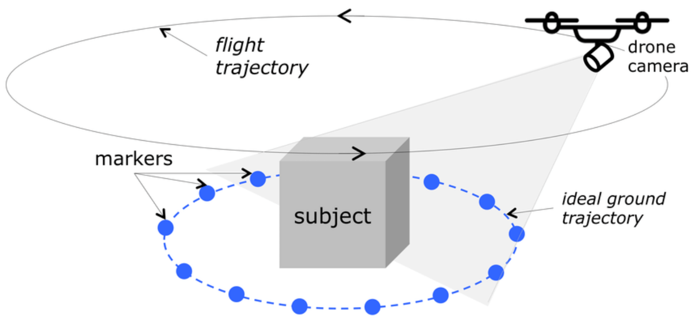
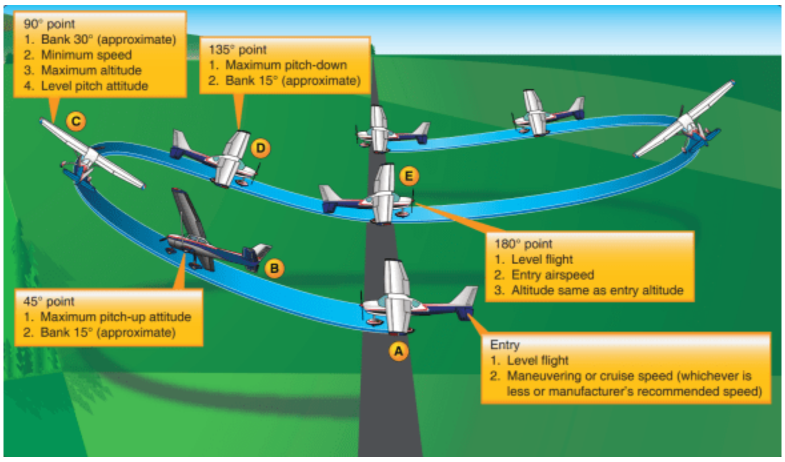


Figure 22 : Figure of Circular Pattern

- Lazy eight Maneuver: The Lazy Eight maneuver is an advanced aviation technique comprising two 180-degree rotations in opposite directions while climbing and descending in a symmetrical manner. Pilots utilize it to enhance coordination, control, and altitude competency. Figure 23 shows the lazy eight pattern.

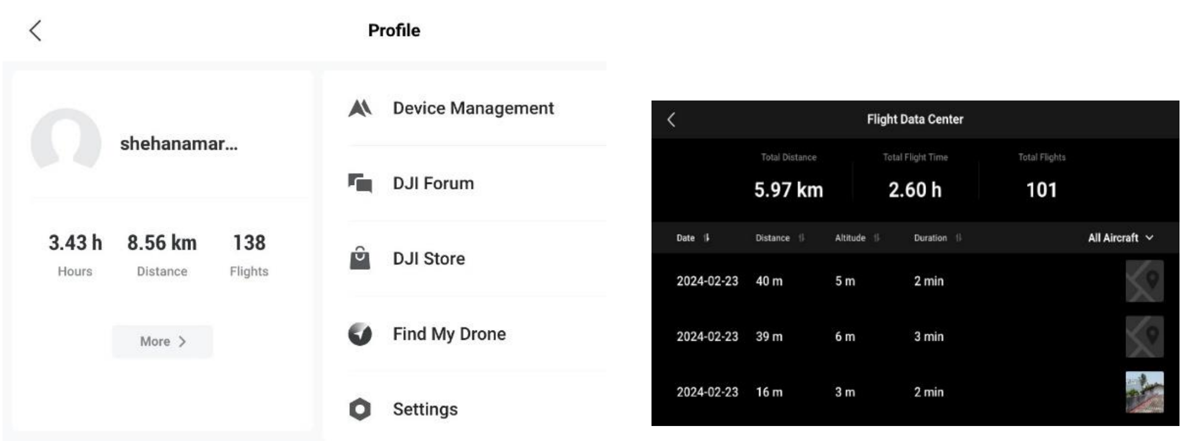

90° point
1. Bank 30° (approximate)
2. Minimum speed
3. Maximum altitude
4. Level pitch attitude
135° point
1. Maximum pitch-down
2. Bank 15° (approximate)
C
D
E
B
A
180° point
1. Level flight
2. Entry airspeed
3. Altitude same as entry altitude
45° point
1. Maximum pitch-up attitude
2. Bank 15° (approximate)
Entry
1. Level flight
2. Maneuvering or cruise speed (whichever is less or manufacturer's recommended speed)
Figure 23 : Figure of Lazy eight Maneuver

These maneuvering patterns give a dataset covering a wide spectrum of drone motions, from basic maneuvers to more complex flight tactics. Some wide spectrum drone dynamics that illustrate these maneuvering patterns are shown below.

- Forward Flight: Propelling the drone in a linear trajectory.
- Reverse Flight: Propelling the drone on a linear path in the opposite direction.
- Sideways Flight (Left/Right): Moving the drone horizontally to the left or right.
- Yaw , Pitch and roll rotations moments.
- Ascend: Increasing the drone's altitude.
- Descend: Decreasing the drone's altitude.
- Banked Turns: Tilting the drone to commence a turn, providing a smooth arc rather than a flat turn.
- Orbit: The drone flies in a circular path around a subject, maintaining a steady distance and orientation.

This extensive dataset is vital for analyzing drone performance, boosting pilot abilities, and furthering research in diverse domains such as aerial photography, mapping, and robotics. By employing the three Pattern Eight Maneuver, Circular Pattern, and Lazy Eight Maneuver, a comprehensive array of dynamic motions can be addressed without requiring data collection for each dynamic separately. These specific patterns are chosen over random patterns because they inherently include varied dynamics within their structured movements, providing a comprehensive overview of the drone's capabilities and pilot ability.

Nearly 100 drone flights were done over a period of nearly three months to obtain the data, totaling 8 hours of flying time covering 10 kilometers. This distributed strategy was adopted instead of gathering data in a single session to ensure a more comprehensive understanding of diverse flight situations over time. The flight details were captured using the DJI Fly app, with figure 24 showing sample of these recorded flights. Each experiment completed by a single drone pilot involves nearly thousands of data points, while the overall dataset contains over 100,000 data points. These flight logs are stored either on the mobile phone linked to the drone remote controller or on the SD card installed into the drone, saved as text files. The duration of each flight varied from 2 minutes to 10 minutes.



3.43 h	8.56 km	138
Hours	Distance	Flights

Total Distance	Total Flight Time	Total Flights
5.97 km	2.60 h	101

Date	Distance	Altitude	Duration
2024-02-23	40 m	5 m	2 min
2024-02-23	39 m	6 m	3 min
2024-02-23	16 m	3 m	2 min

Figure 24 : Sample Flight Details

Drone operations sometimes involve using a camera attached to the drone to send a live video feed to a viewing device, known as FPV (First-person view). Figure 25 displays the pilot using FPV by obtaining real-time footage from a camera attached to the drone. Figure 26 depicts a real aerial view captured during a data collection period held on university premises.

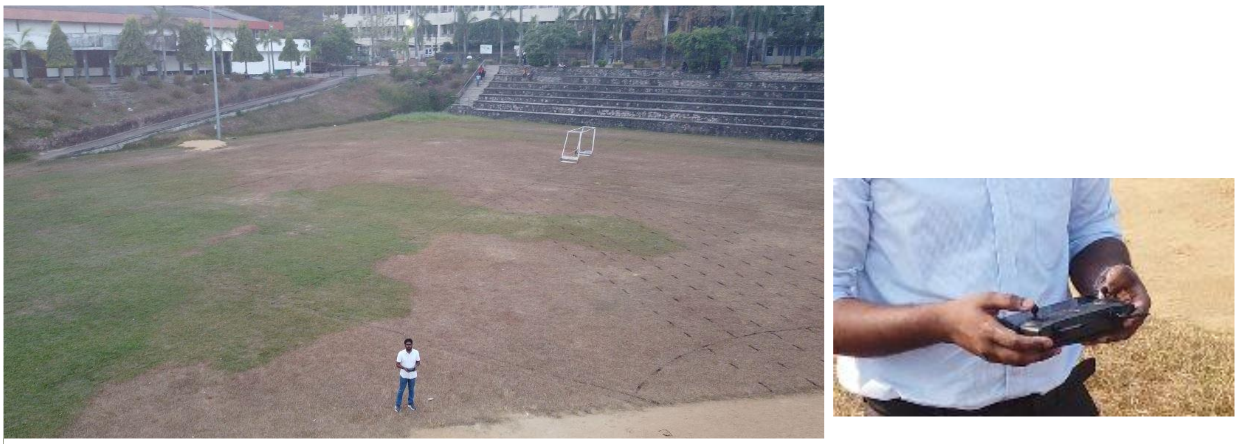


Figure 25 : First person view of the drone via a drone camera

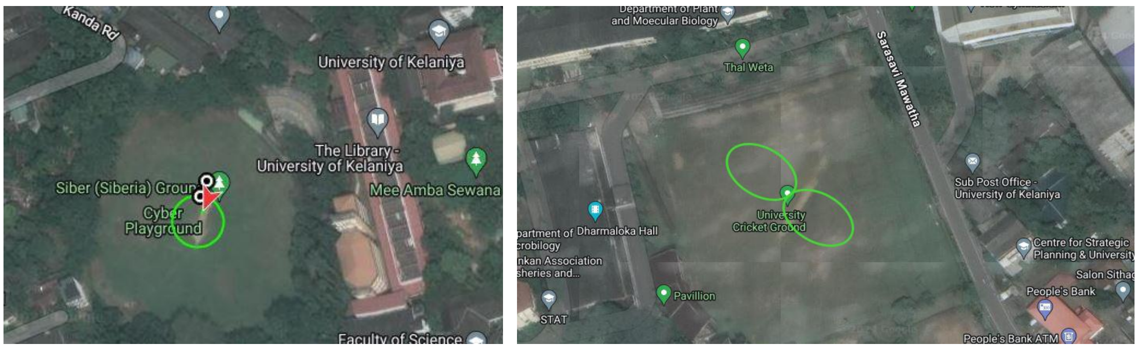


Figure 26 : Actual aerial view of a sample drone pilot drill

Accurately maneuvering the drone in three-dimensional space without any visible indicators according to these three pattern shapes is a difficult challenge. To attain accurate shapes, researchers employ physical markers placed on the ground to direct the trajectory of the drone. By operating the drone and utilizing these markings as visual references, it's able to maintain its original form. To ensure that the acquired data of the drone matches with the real flight path, the researcher examined the size and shape of the recorded pattern with real markers on the ground. It verified there was no GPS error in the recorded data. Additionally, the experienced drone pilots visually observed the drone's position relative to these landmarks on the ground.

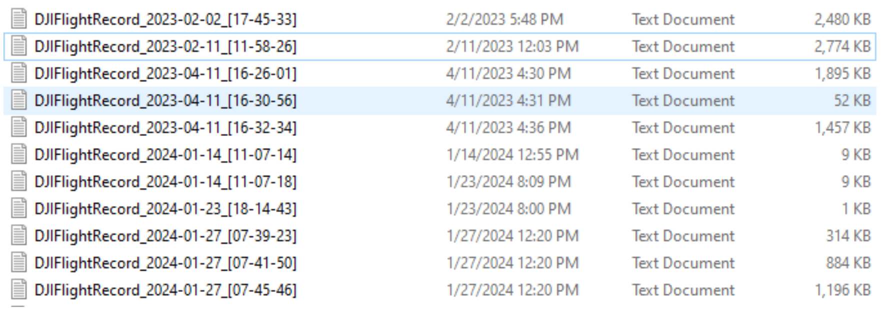
The Waypoint Planning functionality in DJI drones improves accurate navigation by guiding the drone to fly constantly between defined positions of shapes. However, this approach does not possess the ability to compile detailed flight records. Because it does not record input parameters from the remote controller. The research focuses on analyzing the connection between drone remote controller inputs and drone position/orientation. Therefore, employing waypoints isn't suitable.

3.3 Data Pre-Processing

There are serval methods including pre-processing steps. Each of these steps is described below.

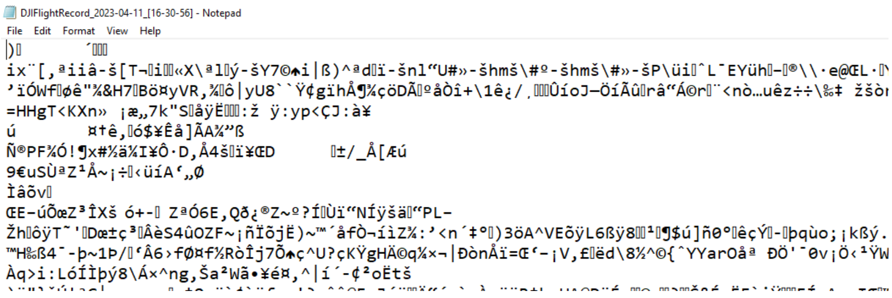
3.3.1 Numerical Measurements of Data

The data saved on either the mobile phone or SD card is kept as a text file. An example of such text files are presented in figure 27 and the content of the text file indicates figure 28. The initial stage involved translating the string values of the text file data into numerical numbers. DJI provides a flight log viewer on their website, which makes the transformation of all text files into CSV (Comma Separated Values) format.("DJI Flight Log Viewer | Phantom Help", n.d.)
Link for web site: https://www.phantomhelp.com/LogViewer/Upload/



DJIFlightRecord_2023-02-02_[17-45-33]	2/2/2023 5:48 PM	Text Document	2,480 KB
DJIFlightRecord_2023-02-11_[11-58-26]	2/11/2023 12:03 PM	Text Document	2,774 KB
DJIFlightRecord_2023-04-11_[16-26-01]	4/11/2023 4:30 PM	Text Document	1,895 KB
DJIFlightRecord_2023-04-11_[16-30-56]	4/11/2023 4:31 PM	Text Document	52 KB
DJIFlightRecord_2023-04-11_[16-32-34]	4/11/2023 4:36 PM	Text Document	1,457 KB
DJIFlightRecord_2024-01-14_[11-07-14]	1/14/2024 12:55 PM	Text Document	9 KB
DJIFlightRecord_2024-01-14_[11-07-18]	1/23/2024 8:09 PM	Text Document	9 KB
DJIFlightRecord_2024-01-23_[18-14-43]	1/23/2024 8:00 PM	Text Document	1 KB
DJIFlightRecord_2024-01-27_[07-39-23]	1/27/2024 12:20 PM	Text Document	314 KB
DJIFlightRecord_2024-01-27_[07-41-50]	1/27/2024 12:20 PM	Text Document	884 KB
DJIFlightRecord_2024-01-27_[07-45-46]	1/27/2024 12:20 PM	Text Document	1,196 KB

Figure 27 : Samples of single drone flights

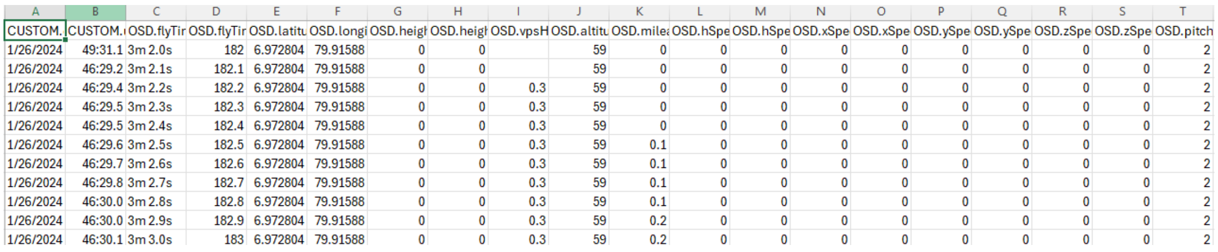


DJIFlightRecord_2023-04-11_[16-30-56] - Notepad
File Edit Format View Help

```
)□         ´□□□
ix¨[,ªiiâ-š[T¬□i□□«X\ª1□ý-šY7©♠i|ß)^ªd□ï-šnl“U#»-šhmš\#º-šhmš\#»-šP\üi□ˆLˉEYüh□–□®\\·e@ŒL·□
’ïÓWf□øê"%&H7□Bö¤yVR,¾□ô|yU8``Ÿ¢gïhÅ¶¾çöDÃ□ºåÒî+\1ê¿/‚□□□ÛíoJ—ÖíÃû□râ“Á©r□¨<nò…uêz÷÷\‰‡ žšòı
=HHgT<KXn» ¡æ„7k"S□åÿË□□□:ž ÿ:yp<ÇJ:à¥
ú        ¤†ê,□ó$¥Êå]ÃA¾”ß
Ñ®PF¾Ó!¶x#½ä¼I¥Ô·D,Å4š□ï¥ŒD      □±/_Å[Æú
9€uSÙªZ¹Å~¡÷□‹üíA‘„Ø
Ìâõv□
ŒE–úÕœZ³ÎXš ó+-□ ZªÓ6E,Qð¿®Z~º?Í□Ùï“NÍÿšä□“PL-
Žh□ôÿT˜'□Dœ±ç³□ÂèS4ûOZF~¡ñÏõjË)~™´åfÒ¬íìZ¾:’<n´‡°□)3öA^VEõÿL6ßÿ8□□¹□¶$ú]ñ0°□êçÝ□-□þqùo;¡kßý.
™H‰ß4¯-þ~1Þ/□‘Â6›fØ¤f½RòÎj7Õ♠ç^U?çKŸgHÄ©q¼×¬|ÐònÅï=Œ‘–¡V,£□ëd\8½^©{ˆYYarOåª ÐÖ'¯0v¡Ö‹¹ŸW
Àq>i:LóÍÌþý8\Á×^ng,Ša²Wã•¥é¤,^|í´-¢²oËtš
```

Figure 28 : Text File of a single Drone Flight

Each flight log covers 184 features and covers nearly 1000 data points, with an approximate flight length of 10 minutes. The details of each parameter are provided below, with a sample of CSV data points depicted in figure 29.



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
CUSTOM.	CUSTOM.ı	OSD.flyTir	OSD.flyTir	OSD.latitu	OSD.longi	OSD.heigł	OSD.heigł	OSD.vpsH	OSD.altitu	OSD.milea	OSD.hSpe	OSD.hSpe	OSD.xSpe	OSD.xSpe	OSD.ySpe	OSD.ySpe	OSD.zSpe	OSD.zSpe	OSD.pitch
1/26/2024	49:31.1	3m 2.0s	182	6.972804	79.91588	0	0		59	0	0	0	0	0	0	0	0	0	2
1/26/2024	46:29.2	3m 2.1s	182.1	6.972804	79.91588	0	0		59	0	0	0	0	0	0	0	0	0	2
1/26/2024	46:29.4	3m 2.2s	182.2	6.972804	79.91588	0	0	0.3	59	0	0	0	0	0	0	0	0	0	2
1/26/2024	46:29.5	3m 2.3s	182.3	6.972804	79.91588	0	0	0.3	59	0	0	0	0	0	0	0	0	0	2
1/26/2024	46:29.5	3m 2.4s	182.4	6.972804	79.91588	0	0	0.3	59	0	0	0	0	0	0	0	0	0	2
1/26/2024	46:29.6	3m 2.5s	182.5	6.972804	79.91588	0	0	0.3	59	0.1	0	0	0	0	0	0	0	0	2
1/26/2024	46:29.7	3m 2.6s	182.6	6.972804	79.91588	0	0	0.3	59	0.1	0	0	0	0	0	0	0	0	2
1/26/2024	46:29.8	3m 2.7s	182.7	6.972804	79.91588	0	0	0.3	59	0.1	0	0	0	0	0	0	0	0	2
1/26/2024	46:30.0	3m 2.8s	182.8	6.972804	79.91588	0	0	0.3	59	0.1	0	0	0	0	0	0	0	0	2
1/26/2024	46:30.0	3m 2.9s	182.9	6.972804	79.91588	0	0	0.3	59	0.2	0	0	0	0	0	0	0	0	2
1/26/2024	46:30.1	3m 3.0s	183	6.972804	79.91588	0	0	0.3	59	0.2	0	0	0	0	0	0	0	0	2

Figure 29 : Sample Data Set as CSV file

The data comprises several characteristics relating to drone flight activities. Each parameter gives information about many aspects of the flight, such as position, altitude, speed, battery state, remote control signals, camera functionality, and more. Here's a explanation of some of the major parameters:

- CUSTOM.date [local]: This parameter indicates the flight date in the local time zone of the drone's operation.
- CUSTOM.updateTime [local]: Represents the most recent update time of the flight data in the local time zone.
- OSD (On-Screen Display) flyTime: Cumulative duration of the drone's flight from the beginning.
- The OSD.latitude represents the latitude coordinates of the drone's current position.

- OSD.longitude: The longitudinal coordinates of the drone's present location.
- OSD.height [ft]: Current altitude above ground level in feet.
- OSD.hSpeed [MPH]: The drone's horizontal speed, usually measured in miles per hour.
- OSD.pitch: The pitch angle of the drone indicates the degree of upward or downward tilt of the aircraft.
- OSD.roll: The roll angle of the drone shows the degree of tilt to the left or right.
- OSD.yaw: The yaw angle of the drone indicates the direction the drone is looking in relation to its original orientation.
- RC (Remote Controller).aileron: Aileron control often refers to the input that controls the roll of an aircraft, especially drones. Ailerons are control surfaces on the wings that move in opposite directions to each other to tilt the aircraft, thus generating a roll.
- RC.elevator: Elevator control is responsible for managing the pitch of the aircraft. It alters the angle of the horizontal stabilizer (or elevators) which in turn controls the pitch attitude of the aircraft.
- RC.throttle: Throttle control regulates the power output of the drone's propulsion system, commonly managing the speed or altitude of the aircraft. Increasing throttle normally makes the drone soar, while decreasing throttle causes it to sink.
- RC.rudder: Rudder control governs the yaw of the aircraft. Yaw is the rotation of the airplane about its vertical axis, controlled by the rudder. This control helps the drone to shift direction.
- RC.mode: This parameter likely corresponds to the flying mode of the drone. Drones often have numerous flight modes like as manual, stabilized, altitude hold, GPS hold, etc., each offering different levels of control and aid to the pilot.
- GIMBAL.pitch: Pitch angle of the gimbal, which regulates the vertical alignment of the camera.
- GIMBAL.roll: Roll angle of the gimbal, controlling the horizontal alignment of the camera.
- GIMBAL.yaw: Yaw angle of the gimbal, determining the direction the camera is pointed relative to the drone's orientation.
- CAMERA.isPhoto: Indicates whether the camera is set to take a photo.
- CAMERA.isVideo: Indicates whether the camera is set to record video.
- RC.downlinkSignal: Represents the signal strength of the downlink connection between the remote controller and the drone.
- BATTERY.chargeLevel: Current charge level of the drone's battery.

- BATTERY.voltage [V]: Voltage of the drone's battery.
- BATTERY.temperature [F]: Temperature of the drone's battery, commonly measured in degrees Fahrenheit.
- HOME.latitude: Latitude coordinates of the home point, which is the original launch location of the drone.
- HOME.longitude: Longitude coordinates of the home point.
- DETAILS.totalTime [s]: Total time passed during the flight in seconds.
- DETAILS.totalDistance [ft]: Total distance covered during the flight, measured in feet.
- DETAILS.maxHeight [ft]: Maximum altitude attained during the flight.
- DETAILS.maxHorizontalSpeed [MPH]: Maximum horizontal speed reached throughout the flight.
- DETAILS.maxVerticalSpeed [MPH]: Maximum vertical speed attained throughout the flight.
- APPGPS.latitude: Latitude coordinates retrieved from the GPS of the mobile app.
- APPGPS.longitude: Longitude coordinates retrieved from the GPS of the mobile app.
- APPGPS.accuracy: Accuracy of the GPS coordinates provided by the mobile app.
- APP.tip: Tips linked to drone flight data, likely delivered by the flight control software or mobile app.
- APP.warning: Warnings tied to drone flight data, providing alarms for potential faults or hazards during flight.

3.3.2 Feature Selection for Build Simulation Drone Dynamics Model

The next stage is to determine the key features that are used to construct the model. Starting from a battery level between 97% and 100% carrying out all operations with fully charged batteries that helps to prevent the risk of performance variations in the drone due to variances in battery power. Several conclusions about the parameters and implications were drawn from the analysis of the raw data set:

- **Positional Parameters**: The drone's position in three dimensions is indicated by parameters like latitude, longitude, and altitude. The drone's geographic location can be described by the combination of these connected factors.
- **Time Stamp Consistency**: Every data point was recorded at consistent intervals, with a fixed amount of time elapsed between each capture. This guarantees the temporal component of the data's uniformity and consistency, enabling precise analysis throughout time.

- **Velocity Parameters**: In three dimensions, the drone's velocity components are represented by VelocityX, VelocityY, and VelocityZ. These values aid in understanding the drone's movement and trajectory over time, much like location parameters do.
- **Constant Parameters**: Some sets of parameters don't change much or don't change at all during the experiment. For instance, unless there are notable alterations or problems with the drone's power source, parameters like GpsCount, GpsLevel, Battery Power (%), Battery Voltage, Battery Voltage Deviation, and Battery Cell Voltages usually stay constant.
- **Static Parameters**: Some parameters don't change based on the drone's specified settings, or they offer advice and warnings in place of dynamic data. Examples of such parameters are App Tip, App Warning, App Message, and Flight Mode, which provide information or notifications on how the drone is operating but don't change much while in flight.

In considering the above-described variables, the following crucial parameters were chosen for selection from the dataset:

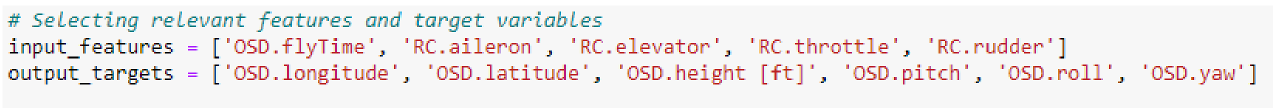
- **Time** (seconds): This option shows how long it has been since the drone was turned on. It functions as a foundational measure for monitoring the flight data's time component.
- **RcAileron**: The radio controller signals used to control the drone's roll are represented by this parameter. It offers information on how the drone's horizontal tilting is changed by the pilot.
- **RcElevator**: The drone's horizontal pitch attitude is controlled by these signals from the radio controller. They show that the drone's forward or backward tilt has been adjusted.
- **RcRudder**: The drone's rotation around its vertical axis is controlled by the yaw, which is determined by these signals from the radio controller.
- **RcThrottle**: The engine's speed is controlled by these signals from the radio controller, which affect how quickly the drone climbs or falls. They provide details regarding the drone's total speed of movement.
- **Cartesian coordinates**: These three parameters (x, y, z) stand for the Cartesian coordinates that are obtained from latitude and longitude measurements. They enable analysis in three dimensions by giving a spatial representation of the drone's position.
- **Orientation**: The drone's head's degree of bearing is indicated by this parameter. It means pitch , yaw and raw angles of drone. It helps to comprehend the orientation of

the drone in relation to its surroundings by providing information about which way it is facing.

These parameters capture the location and orientation of the drone when changing the input parameter of the radio controller. After combining these parameters, it provides a complete dataset for the creation of a machine-learning drone dynamic model. Together, these parameters capture important facets of drone operation, allowing the model to efficiently learn and simulate the drone's behavior.

In response to radio controller input, the machine learning-based drone dynamic model must predict the position and orientation of the drone. As a result, the data can be divided into the following two major categories. Figure 30 illustrates certain target variables and input parameters.

- The position and orientation of the drone changed over time.
- Drone pilot inputs are entered via the drone RC such as throttle and rudder values that are responsible for the drone position and orientation changes.

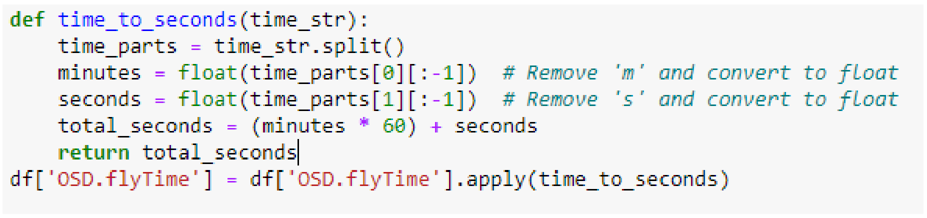


```python
# Selecting relevant features and target variables
input_features = ['OSD.flyTime', 'RC.aileron', 'RC.elevator', 'RC.throttle', 'RC.rudder']
output_targets = ['OSD.longitude', 'OSD.latitude', 'OSD.height [ft]', 'OSD.pitch', 'OSD.roll', 'OSD.yaw']
```

Figure 30 : Input features and target variables

3.3.3 Transform String Values of Time into Seconds

Prior to starting model training, it is essential to choose the features. The time in the CSV file is formatted as a string, for example, "12m 15.1s". Before proceeding, the time values in string format need to be transformed to numerical format in total seconds. Figure 31 displays a python code for converting a text representing time into numerical seconds.



```python
def time_to_seconds(time_str):
    time_parts = time_str.split()
    minutes = float(time_parts[0][:-1])  # Remove 'm' and convert to float
    seconds = float(time_parts[1][:-1])  # Remove 's' and convert to float
    total_seconds = (minutes * 60) + seconds
    return total_seconds
df['OSD.flyTime'] = df['OSD.flyTime'].apply(time_to_seconds)
```

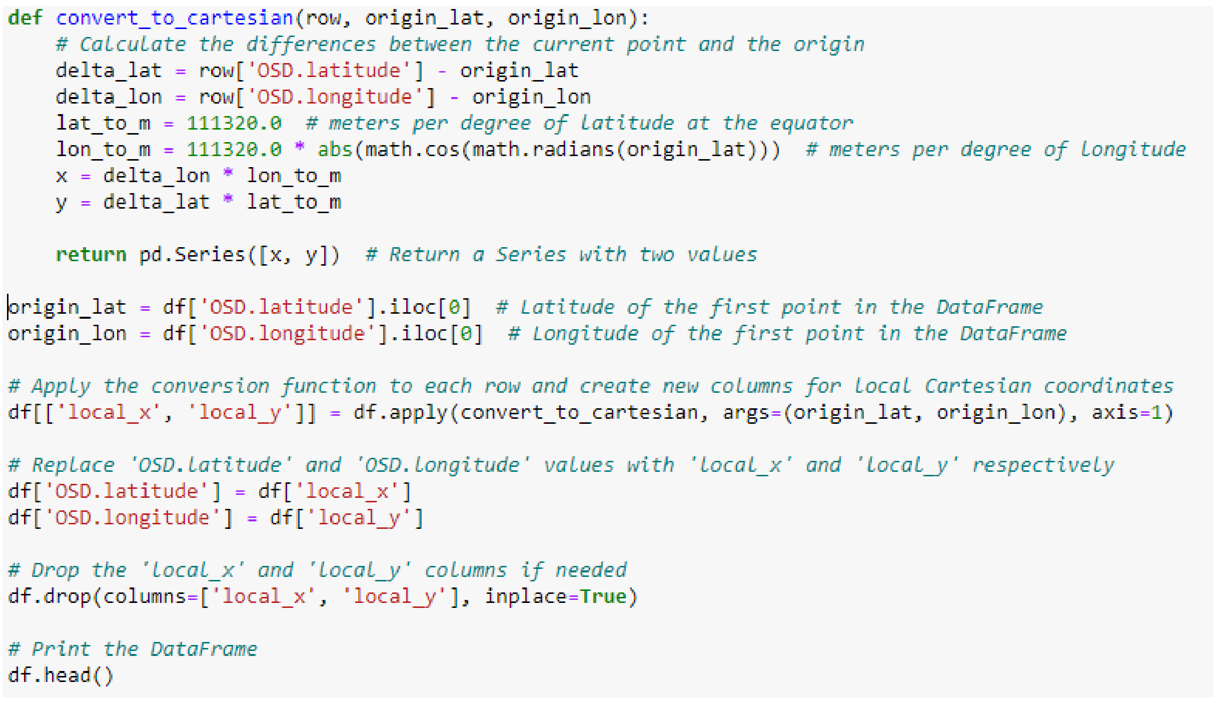
Figure 31 : Convert Time into Second

3.3.4 Convert Drone Position into Local Cartesian Coordinates

To identify the drone position latitude and longitude values are used in CSV files. For simple calculation of machine learning model, it is necessary to convert latitude and longitude values into local cartesian system. For example calculation of velocities, accelerations, or angles are more easily in cartesian coordinates, which can serve as valuable features for predicting drone dynamics. There are numerous global ways for translating latitude and longitude into Cartesian coordinate systems:

- Equirectangular Projection: This basic approach immediately maps longitude and latitude to x and y coordinates, assuming equal distances for one degree of latitude and longitude. However, it does not accurately store distances.
- Mercator Projection: This cylindrical map projection retains angles and is commonly used for navigation. Latitude is directly mapped to the y-coordinate, whereas longitude is scaled and mapped to the x-coordinate. It does, however, distort areas at high latitudes.
- UTM (Universal Transverse Mercator): This approach divides the Earth into zones and projects each zone separately using a transverse Mercator projection. It delivers exact measurements in meters inside each zone.
- Albers Equal Area Conic Projection: This projection balances area distortion with shape distortion, making it suited for maps of continents or countries. It's typically used for thematic maps.

But the study focuses on a specific local area rather than a bigger region. hence, Local coordinate systems may be better appropriate for small-scale mapping projects (Patrik *et al.*, 2019). In this coordinate system, choosing the drone's home location as the initial point (0,0) simplifies calculations and permits accurate distance measurements over smaller regions. Figure 32 displays python implementation of this transformation.



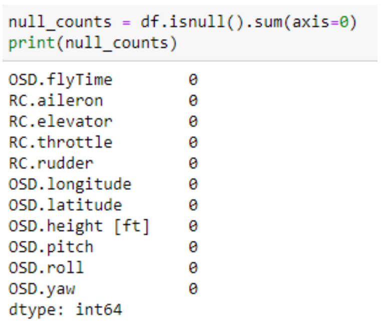
```python
def convert_to_cartesian(row, origin_lat, origin_lon):
    # Calculate the differences between the current point and the origin
    delta_lat = row['OSD.latitude'] - origin_lat
    delta_lon = row['OSD.longitude'] - origin_lon
    lat_to_m = 111320.0  # meters per degree of latitude at the equator
    lon_to_m = 111320.0 * abs(math.cos(math.radians(origin_lat)))  # meters per degree of longitude
    x = delta_lon * lon_to_m
    y = delta_lat * lat_to_m

    return pd.Series([x, y])  # Return a Series with two values

origin_lat = df['OSD.latitude'].iloc[0]  # Latitude of the first point in the DataFrame
origin_lon = df['OSD.longitude'].iloc[0]  # Longitude of the first point in the DataFrame

# Apply the conversion function to each row and create new columns for local Cartesian coordinates
df[['local_x', 'local_y']] = df.apply(convert_to_cartesian, args=(origin_lat, origin_lon), axis=1)

# Replace 'OSD.latitude' and 'OSD.longitude' values with 'local_x' and 'local_y' respectively
df['OSD.latitude'] = df['local_x']
df['OSD.longitude'] = df['local_y']

# Drop the 'local_x' and 'local_y' columns if needed
df.drop(columns=['local_x', 'local_y'], inplace=True)

# Print the DataFrame
df.head()
```

Figure 32 : Convert drone position into cartesian coordinate system

The code defines a function convert_to_cartesian that turns latitude and longitude coordinates into local Cartesian coordinates relative to a provided origin point. It determines the differences between each coordinate and the origin, then applies factor conversions to convert degrees to meters for latitude and longitude. The generated Cartesian coordinates are added as new columns to the Data Frame. Furthermore, the original latitude and longitude values are replaced with the Cartesian coordinates. This procedure effectively turns geographic coordinates into a local Cartesian coordinate system, enabling calculations and measurements inside a specific area.

3.3.5 Remove Null Values

All data points about drone position and remote controller input settings were accurately created in the DJI flight logs without any missing values. Figure 33 shows the distribution of null value counts.



```python
null_counts = df.isnull().sum(axis=0)
print(null_counts)
```

```
OSD.flyTime        0
RC.aileron         0
RC.elevator        0
RC.throttle        0
RC.rudder          0
OSD.longitude      0
OSD.latitude       0
OSD.height [ft]    0
OSD.pitch          0
OSD.roll           0
OSD.yaw            0
dtype: int64
```

Figure 33 : Null Values Count

3.3.6 Handling Outliers

Out of the total 100,998 data points in the dataset, 42,023 are recognized as outliers. While deleting these outliers may highly hinder model performance. Because of these large number of outliers, deleting outlier data points can lower the dataset's size and diversity, potentially resulting in weaker generalization and skewed estimates. Additionally, the elimination of outliers may alter the dataset's distribution and properties, impacting measures like mean, variance, skewness and correlation. Furthermore, it may cause errors or inaccuracies, such as inconsistencies or duplications within the dataset. Figure 34 demonstrates the outliers of certain features using boxplots.

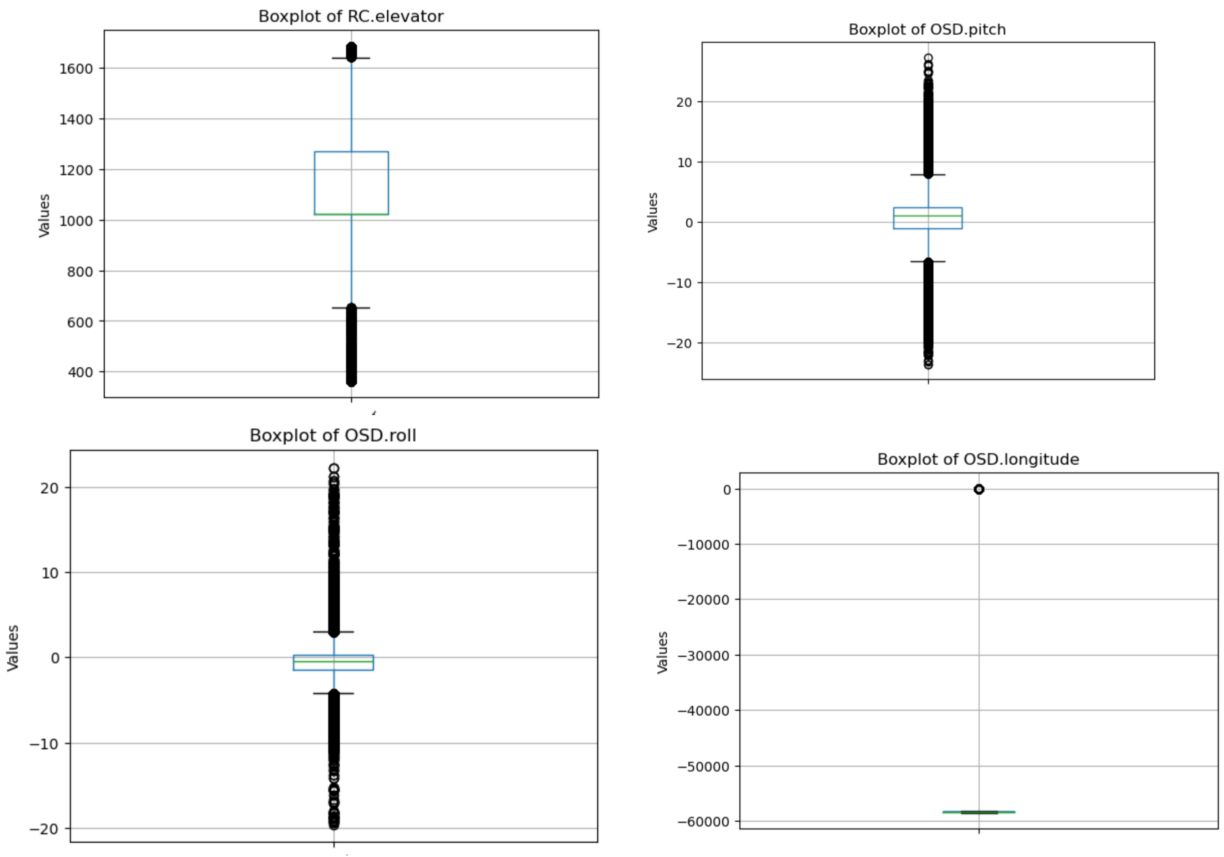
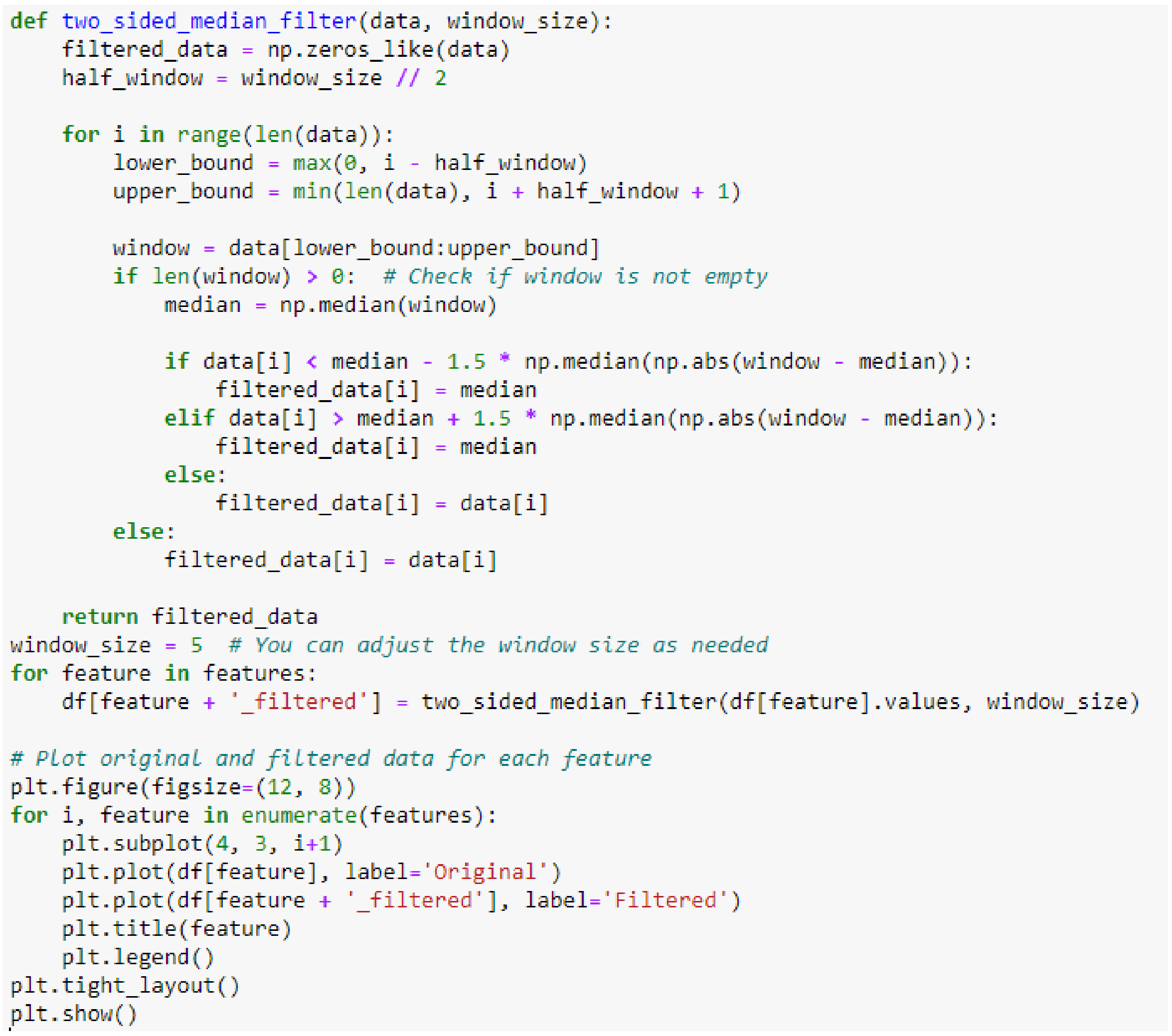


Figure 34 : Outliers of Features

Xiuzhen Jiao and Hui Lu Rongling discovered a technique to handle the high number of outliers contained in drone flight data (Jiao, X., Lu, H. and Lang, R, (2009)). They proposed the application of the Two-sided median filtering strategy, which greatly minimizes the amount of outlier data points. This approach is used to smooth the data by replacing outlier values with the median value of surrounding items within a specified interval. The approach is particularly beneficial when working with time-series data or data where outliers might reflect noise. After following the application of this approach, the dataset had just 6088 outlier points. Figure 35 displays the Python implementation of the two-sided median filtering technique.



```python
def two_sided_median_filter(data, window_size):
    filtered_data = np.zeros_like(data)
    half_window = window_size // 2

    for i in range(len(data)):
        lower_bound = max(0, i - half_window)
        upper_bound = min(len(data), i + half_window + 1)

        window = data[lower_bound:upper_bound]
        if len(window) > 0:  # Check if window is not empty
            median = np.median(window)

            if data[i] < median - 1.5 * np.median(np.abs(window - median)):
                filtered_data[i] = median
            elif data[i] > median + 1.5 * np.median(np.abs(window - median)):
                filtered_data[i] = median
            else:
                filtered_data[i] = data[i]
        else:
            filtered_data[i] = data[i]

    return filtered_data
window_size = 5  # You can adjust the window size as needed
for feature in features:
    df[feature + '_filtered'] = two_sided_median_filter(df[feature].values, window_size)

# Plot original and filtered data for each feature
plt.figure(figsize=(12, 8))
for i, feature in enumerate(features):
    plt.subplot(4, 3, i+1)
    plt.plot(df[feature], label='Original')
    plt.plot(df[feature + '_filtered'], label='Filtered')
    plt.title(feature)
    plt.legend()
plt.tight_layout()
plt.show()
```

Figure 35 : Two-sided Median Filtering Method

3.3.7 Remove Duplicate Data

Duplicate data points can occur when drones hover or stay still because their sensors keep recording readings even when there is no movement. This phenomenon happens because sensors, such as GPS receivers and IMUs, continuously send data to flight log including position, orientation, and velocity every 0.1 seconds, regardless of whether the drone is moving. Thus, during periods of hovering or stationary flight, repeated sensor readings may produce identical or almost identical values, resulting in duplicate entries in the dataset. Removing duplicate entries from drone flight datasets is critical to ensure the accuracy and reliability of the data. Also, duplicate entries might develop due to different factors such as sensor noise, signal interference, or limitations in sensor precision. By recognizing and deleting these duplicates, researchers can receive a more precise representation of the drone's behavior. This method helps prevent skewed analysis, false insights, and errors in model training. In this dataset, 6014 duplicate data points have been eliminated to improve the quality of the data.

3.3.8 Normalization

Normalization is a fundamental step in the data preprocessing phase of machine learning. It is applied to translate features inside a dataset that may reside on various ranges of values into a standardized scale, often ranging from 0 to 1 or, in certain situations, -1 to 1. This standardization is crucial as considerable discrepancies in feature ranges might adversely affect the learning process. Various normalizing methods are available, including the standard scaler and the min-max scaler.

In the typical scaler approach, scaling is conducted individually on each feature by computing relevant statistics from the dataset. On the other hand, the min-max scaling strategy entails scaling each feature independently using the minimum and maximum values provided in the dataset. Normalization of specified columns of a data frame is a preprocessing step in machine learning workflows to ensure that features are on the same size. Figure 36 and figure 37 demonstrate the data points before and after scaling, respectively.

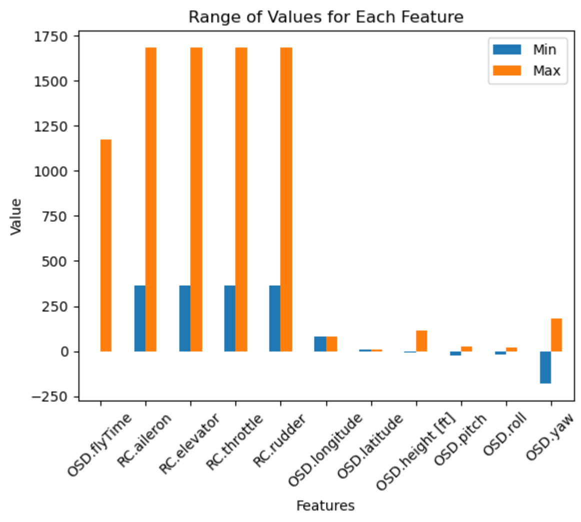


Figure 36 : Range of values before scaling

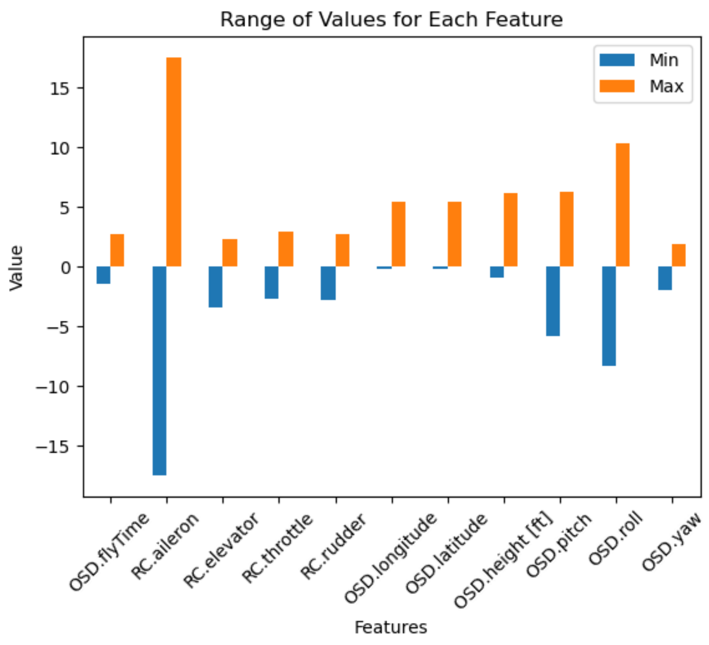


Figure 37 : Range of values after scaling

Before normalization, the values of each feature ranged from -250 to 1750. After normalization, these values were scaled down to a range between -15 and 15.

3.4 Build the Machine Learning Model

Machine learning with simulating drone dynamics involves utilizing computational techniques and statistical models to enable drones to learn and adjust based on data, rather than depending on explicit programming. Machine learning methods enable drones to assess extensive data on their dynamics, forecast outcomes, and enhance their performance without the need for explicit programming for each circumstance. Machine learning is beneficial for modeling drone dynamics. Below are some advantages explained.

- Drones utilize sensors and actuators to collect extensive data on flight dynamics, control inputs, environmental variables, and other factors. Machine learning algorithms may evaluate data to detect patterns, correlations, and intricate dynamics that may not be visible to human programmers.
- Drone dynamics can be affected by various conditions like wind, geography, payload, and battery life, showcasing adaptability. Machine learning algorithms can adjust to varying circumstances by iteratively revising their models using fresh data. Drones can adjust to changes in their surroundings to stay stable, function at their most effectively, and react efficiently to varying conditions.
- Complexity Handling: Drone dynamics require complicated interactions with multiple elements, including aerodynamics, propulsion systems, control algorithms, and external

forces. Traditional analytical tools may struggle to represent this complexity accurately. Machine learning approaches, such as neural networks and deep Learning are capable of modeling complicated, non-linear interactions and can better reflect the intricate dynamics of drones.

- Machine learning models trained on past drone flight data may accurately predict the future states of the drone based on current inputs. These prediction capabilities are essential for autonomous drone navigation, obstacle avoidance, trajectory planning, and other control tasks.
- Machine learning algorithms can optimize drone performance by discovering optimal control tactics, energy-efficient flying pathways, and cargo distributions. By exploiting data-driven insights, machine learning may help drones operate more efficiently, increase battery life, and complete mission objectives with less resources.

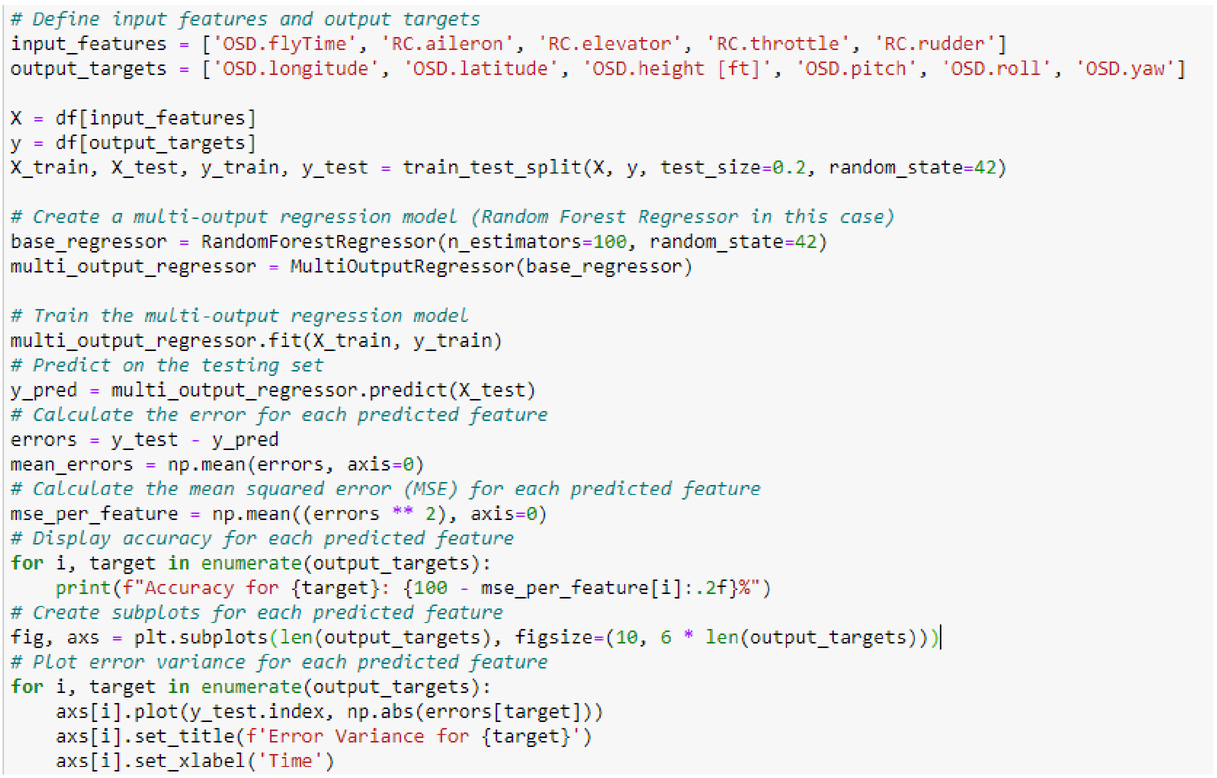
Due to the multiple advantages described above this research focuses on a machine learning-based drone dynamic model based on a subset of artificial intelligence.

Given the dynamic and nonlinear character of both the input and output datasets, which involve the drone pilot's radio controller inputs and the drone's position and orientation, there are various acceptable models to explore. These models should be capable of handling the quick changes in the data and capturing the complex relationships between the inputs and outputs. Below are some potential models that can be utilized to simulate drone movements simultaneously (Mahesh, 2018).

3.4.1 Multi-Output Regression Model

A multi-output regression model is a potent analytical tool utilized to predict many continuous target variables simultaneously. This model allows for the simultaneous prediction of key factors in drone flight data processing, including geographical coordinates (latitude and longitude), altitude, and orientation (pitch, roll, yaw). The model can comprehensively understand the drone's flight dynamics by analyzing these variables collectively to capture the interactions and dependencies among them. This method simplifies the model, maintains uniformity in predictions, and improves resilience to missing data, making it well-suited for assessing drone flight data with numerous interrelated variables (Schmid *et al.*, 2022) .

A multi-output regression model is a strong tool for assessing drone flight data by predicting numerous variables at once, but it may face difficulties in situations with complex and nonlinear drone motions. Because linear regression models often struggle to handle complex relationships, leading to poor performance. Considering the constraints in precision seen in simulated situations, this method may not be appropriate. Hence, different approaches must be investigated to accurately depict the complex dynamics of drone flights. The Python code of the developed model is shown in figure 38.



```
# Define input features and output targets
input_features = ['OSD.flyTime', 'RC.aileron', 'RC.elevator', 'RC.throttle', 'RC.rudder']
output_targets = ['OSD.longitude', 'OSD.latitude', 'OSD.height [ft]', 'OSD.pitch', 'OSD.roll', 'OSD.yaw']

X = df[input_features]
y = df[output_targets]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a multi-output regression model (Random Forest Regressor in this case)
base_regressor = RandomForestRegressor(n_estimators=100, random_state=42)
multi_output_regressor = MultiOutputRegressor(base_regressor)

# Train the multi-output regression model
multi_output_regressor.fit(X_train, y_train)
# Predict on the testing set
y_pred = multi_output_regressor.predict(X_test)
# Calculate the error for each predicted feature
errors = y_test - y_pred
mean_errors = np.mean(errors, axis=0)
# Calculate the mean squared error (MSE) for each predicted feature
mse_per_feature = np.mean((errors ** 2), axis=0)
# Display accuracy for each predicted feature
for i, target in enumerate(output_targets):
    print(f"Accuracy for {target}: {100 - mse_per_feature[i]:.2f}%")
# Create subplots for each predicted feature
fig, axs = plt.subplots(len(output_targets), figsize=(10, 6 * len(output_targets)))
# Plot error variance for each predicted feature
for i, target in enumerate(output_targets):
    axs[i].plot(y_test.index, np.abs(errors[target]))
    axs[i].set_title(f'Error Variance for {target}')
    axs[i].set_xlabel('Time')
```

Figure 38 : Implementation of Multi-Output Regression Model

The dataset is divided into input features (X) which include flight characteristics such as fly time and control inputs, and output targets (y) which include parameters like longitude, latitude, altitude, pitch, roll, and yaw. The dataset is then separated into training and testing sets to assist model validation. A Random Forest Regressor is instantiated as the foundation model for multi-output regression and then wrapped within a multi-output regressor. This ensemble technique allows for the prediction of numerous target variables simultaneously. The model is trained on the training data, and subsequently, predictions are generated on the testing set.

3.4.2 Support Vector Machines

Support Vector Machines (SVMs), offer versatility in handling both classification and regression tasks effectively. Particularly in regression, SVM is more suitable, especially when dealing with high-dimensional datasets and nonlinear connections between variables. The kernel trick, an essential aspect of SVMs, allows for the transformation of input data into higher-dimensional spaces. This enables the more accurate representation of nonlinear relationships by creating decision boundaries.

Furthermore, SVMs are resilient against overfitting, especially in high-dimensional spaces, making them excellent for evaluating drone data that may entail several factors. Additionally, SVMs are known for their generalization capability, which is critical in cases when datasets are restricted or noisy.

However, despite these advantages, it's crucial to realize that SVMs, like multi output regression model, may provide unsatisfactory results in certain cases. For instance, if the relationship between inputs and outputs in drone data is particularly complicated or non-linear, SVMs may struggle to capture it adequately. In such circumstances, deep learning algorithms might offer superior performance.

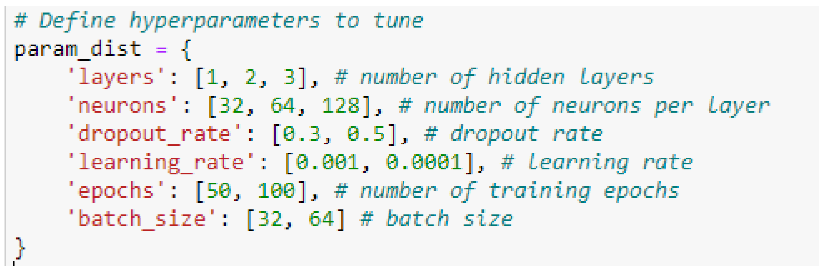
3.4.3 Neural Network

Traditional machine learning algorithms have difficulty in finding complicated connections within drone input and output data due to the so sophisticated nature of the data. However, neural network models offer a possible answer to this challenge. TensorFlow, an open-source deep learning framework built by Google, has emerged as a significant player in this arena since its creation in 2015 ("TensorFlow", n.d.). TensorFlow is known for its clear documentation, robust training tools, and scalability choices. It is beneficial for constructing and utilizing machine learning applications. In addition, it boasts a vibrant community that offers a variety of valuable resources and tools, thus to its popularity. Among these resources, Keras, a high-level neural network library developed on top of TensorFlow, stands out for its user-friendly interface and accelerated model construction process.

Keras simplifies the building of artificial neural networks, offering developers an easy Python-based API for constructing and training models. With its seamless connection with TensorFlow, Keras enables developers to use the potential of deep learning without the complexity commonly associated with older neural network frameworks. Together, Keras and TensorFlow provide a formidable toolkit for addressing the difficulties of drone data analysis. By utilizing the power of neural networks, developers may reveal insights from drone data that may have been challenging to extract using traditional machine learning methodologies.
The neural network architecture adopted in this model has various configurable parameters, each playing a significant role in influencing the model's behavior and performance. Below are the required parameters used in this study (Kadhim *et al.*, 2022).

- Number of Hidden Layers (layers): This parameter influences the depth of the neural network architecture. More layers might potentially capture complicated correlations in the data but may also lead to overfitting if not correctly regularized.
- Number of Neurons per Layer (neurons): Each hidden layer consists of a variable number of neurons, which are computational units responsible for processing the input data and creating output. The number of neurons per layer influences the model's potential to learn and represent patterns in the data.
- Dropout Rate (dropout_rate): Dropout is a regularization technique used to prevent overfitting in neural networks. It randomly deactivates a fraction of neurons during training, forcing the network to acquire more robust properties and lowering reliance on specific neurons.
- Learning Rate (learning_rate): The learning rate specifies the step size at which the model's parameters are updated during training. It plays a significant function in determining the convergence speed and stability of the training process. A greater learning rate may lead to faster convergence but risks overshooting the ideal solution, while a lower learning rate may require more training cycles but ensures smoother convergence.
- Number of Epochs (epochs): An epoch refers to a single run over the complete training dataset throughout the training phase. The number of epochs indicates how many times the model will loop over the dataset. Training for additional epochs allows the model to learn from the data several times, potentially boosting performance, but may also increase the danger of overfitting.
- Batch Size (batch_size): During training, the dataset is divided into smaller batches, and the model updates its parameters based on the average loss estimated over each batch. The batch size determines the number of samples processed in each training iteration. Larger batch sizes can expedite training by using parallel processing and optimizing memory use, but lower batch sizes may give superior generalization and convergence features.

Hyperparameter tuning is critical for optimizing machine learning models, and Randomized Search Cross-Validation (RandomizedSearchCV) is a valuable technique for this purpose. It randomly samples from a predetermined distribution of hyperparameters to discover the ideal combination. These hyperparameters include factors relating to the neural network's architecture (e.g., number of hidden layers, neurons per layer) and training parameters (e.g., dropout rate, learning rate, epochs, batch size). By carefully analyzing each combination's performance using a defined scoring metric, such as negative mean squared error (neg_mean_squared_error), RandomizedSearchCV determines the configuration that produces the best results. This approach efficiently explores hyperparameter space, leading to a well-optimized model capable of obtaining improved accuracy and better generalization on unseen data. Figure 39 shows the defined parameters for tuning the model.



```
# Define hyperparameters to tune
param_dist = {
    'layers': [1, 2, 3], # number of hidden layers
    'neurons': [32, 64, 128], # number of neurons per layer
    'dropout_rate': [0.3, 0.5], # dropout rate
    'learning_rate': [0.001, 0.0001], # learning rate
    'epochs': [50, 100], # number of training epochs
    'batch_size': [32, 64] # batch size
}
```

Figure 39 : Hyper Parameters

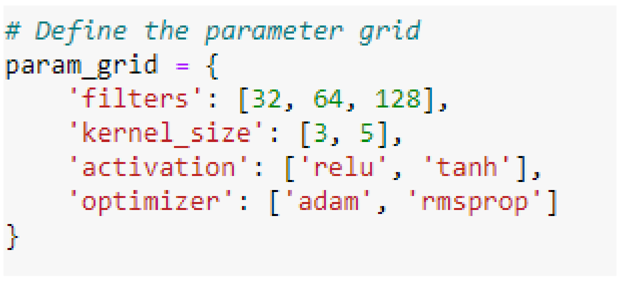
The KerasRegressorWrapper is a custom wrapper class meant to assist in the building and optimization of neural network models using Keras within the Scikit-learn framework. By inheriting the BaseEstimator and RegressorMixin classes in Scikit-learn, this wrapper class offers smooth integration with Scikit-learn's API for model evaluation and hyperparameter adjustment. The neural network model utilizing Keras Sequential API. It starts with an input layer of 64 neurons with ReLU activation, followed by dropout regularization to prevent overfitting. Then, two hidden layers, each with 64 neurons with ReLU activation, are added along with dropout layers. Finally, the output layer is designed to meet the number of output targets for making predictions. The model gave a satisfactory outcome comparison with typical machine learning methods. But further implementation is required.

3.4.4 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a special sort of neural network that is particularly excellent for processing structured grid-like data, such as images and sequential data. CNN and Neural Networks (NNs) differ fundamentally in their architecture and application domains. While NNs are made of fully connected layers where each neuron is linked to every neuron in the subsequent layer, CNNs integrate convolutional layers that apply learnable filters to extract features from grid-like data, such as images or time series. Unlike NNs, which are appropriate for a wide array of tasks, including classification and regression, CNNs are specifically developed for tasks involving spatially structured data (Purwono *et al.*, 2022).

CNN analyze data by utilizing filters that move over the input, identifying local patterns. Pooling layers decrease the size of feature maps, which helps to lower computational complexity. ReLU and other non-linear activation functions introduce non-linearity. Fully connected layers facilitate advanced reasoning processes. CNNs change their weights throughout training to minimize a loss function. They acquire features in a hierarchical manner, starting from basic and progressing to more intricate, which enhances their ability to recognize patterns effectively. Grid search, random search, or Bayesian optimization approaches can be utilized to systematically search across the hyperparameter space. Below are explained available hyperparameters. Figure 40 shows the list of parameters that were trained in the CNN model. After using the appropriate settings, the model displayed improved performance compared to other models. Additionally, the model was tested across diverse maneuvering patterns, further verifying its accuracy.

- filters: Number of filters in the convolutional layers.
- kernel_size: Size of the convolutional kernels.
- activation: Activation function employed in the convolutional layers.
- optimizer: Optimization algorithm utilized during training.



```
# Define the parameter grid
param_grid = {
    'filters': [32, 64, 128],
    'kernel_size': [3, 5],
    'activation': ['relu', 'tanh'],
    'optimizer': ['adam', 'rmsprop']
}
```

Figure 40 : Hyperparameters for CNN Model

3.5 Chapter Summary

In this section, an overview of the methodology is offered. To simulate drone dynamics, four distinct models were applied. Among these models, the CNN model provided excellent findings, notably excelling in scenarios requiring intricate dynamics and large datasets.

CHAPTER 4: EVALUATION AND RESULTS

This chapter provides an overview of the evaluation technique and results of the study. In the context of an AI-based strategy to simulate drone dynamics in three-dimensional space, evaluation methodologies play a significant role in measuring the correctness and effectiveness of the simulation model. One often adopted validation method is comparing simulated outcomes with real-world scenarios. This comparison allows researchers and engineers to evaluate the performance of the simulation model by examining aspects such as the positions and orientations of the drone at specific timestamps.

Quantitative validation methods are particularly valuable in this context since they provide objective measures to assess the accuracy of motion predictions. These methods require extensive statistical analysis and numerical comparisons between simulated and actual data. For instance, metrics such as mean squared error (MSE), root mean squared error (RMSE), or mean absolute error (MAE) can be generated to measure the disagreement between simulated and observed drone movements.

4.1 Summary of Dataset

The collection comprises over 100,000 data points indicating diverse moving patterns observed during drone drills. Each practice was conducted under diverse conditions, including varying velocities and experienced drone pilots. Figure 41 shows the sample maneuvering patterns.

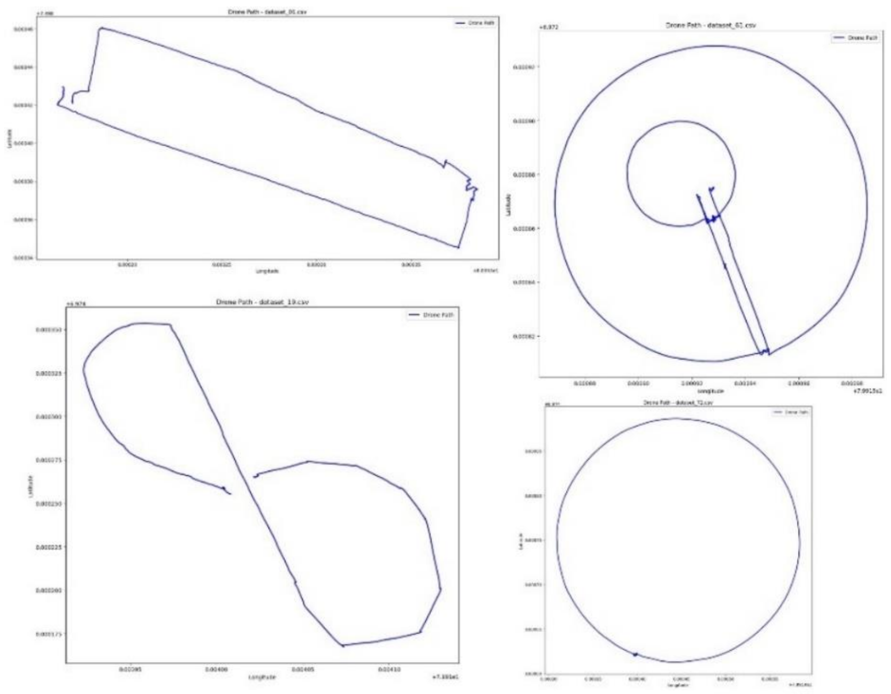
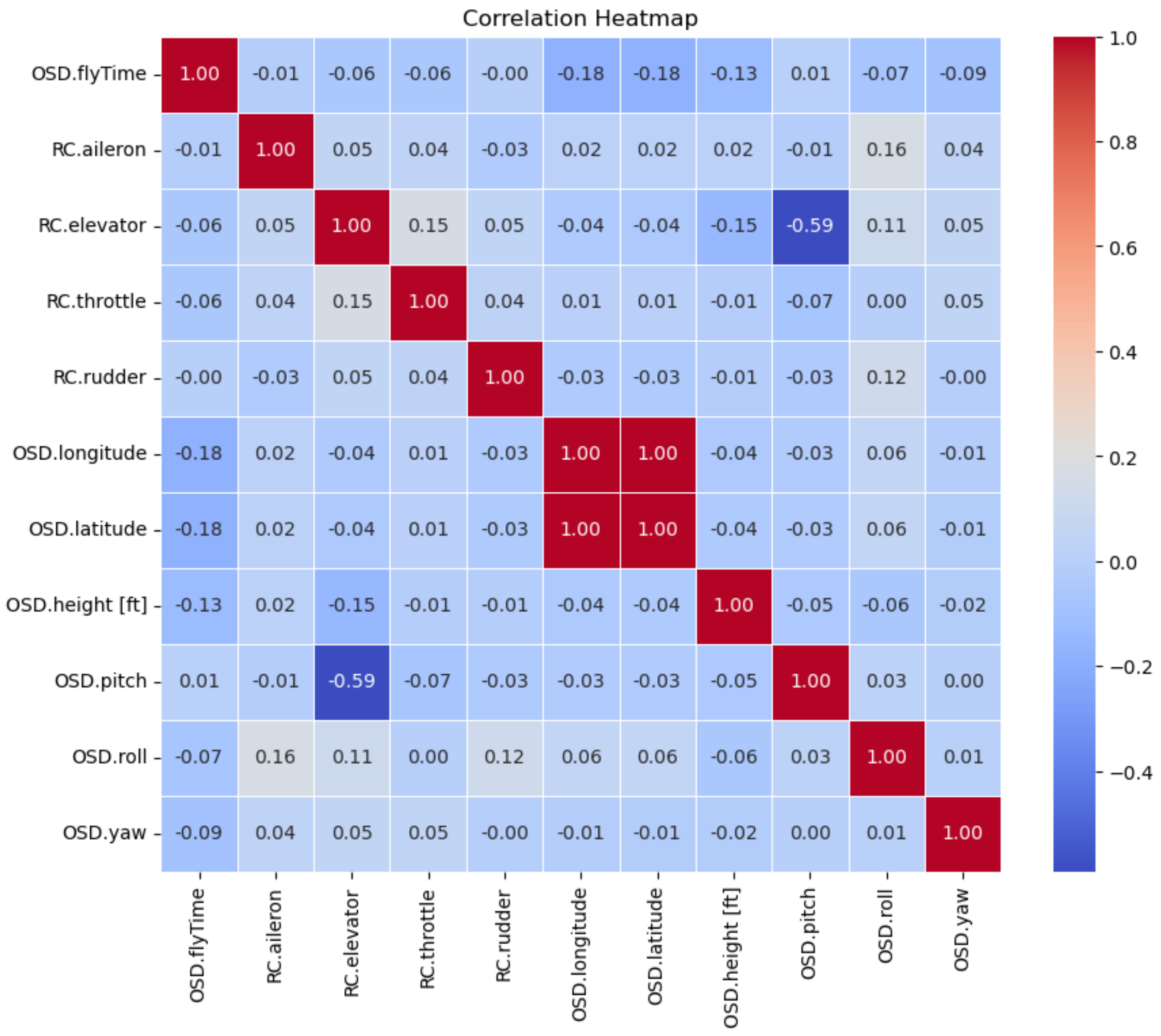


Figure 41 : Sample Drone Drills

Figure 42 depicts the link between input and output attributes. A correlation matrix is a tabular form exhibiting correlation coefficients between variables in a dataset. Each cell in the matrix denotes the correlation coefficient between two variables, indicating the strength and direction of their linear link. The correlation coefficient varies from -1 to 1: 1 shows a perfect positive linear link, where both variables increase together; -1 indicates a perfect negative linear relationship, where one variable increases as the other drops; and 0 indicates no linear relationship.

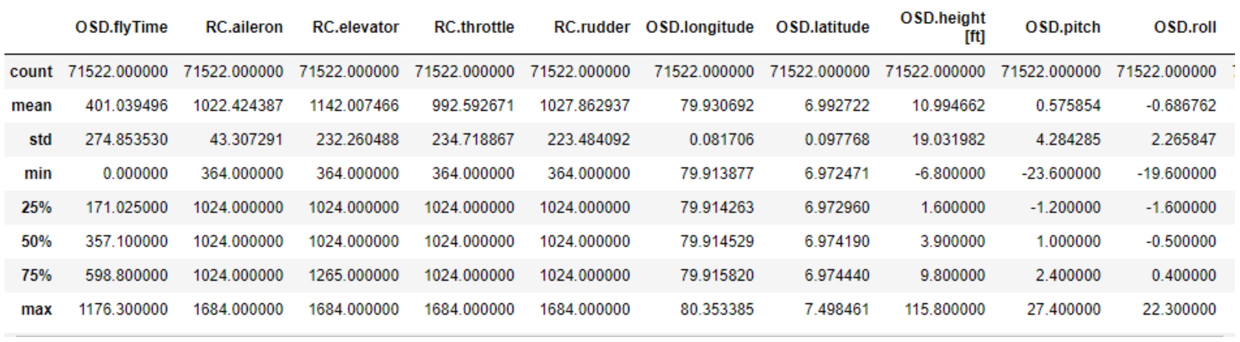


	OSD.flyTime	RC.aileron	RC.elevator	RC.throttle	RC.rudder	OSD.longitude	OSD.latitude	OSD.height [ft]	OSD.pitch	OSD.roll	OSD.yaw
OSD.flyTime	1.00	-0.01	-0.06	-0.06	-0.00	-0.18	-0.18	-0.13	0.01	-0.07	-0.09
RC.aileron	-0.01	1.00	0.05	0.04	-0.03	0.02	0.02	0.02	-0.01	0.16	0.04
RC.elevator	-0.06	0.05	1.00	0.15	0.05	-0.04	-0.04	-0.15	-0.59	0.11	0.05
RC.throttle	-0.06	0.04	0.15	1.00	0.04	0.01	0.01	-0.01	-0.07	0.00	0.05
RC.rudder	-0.00	-0.03	0.05	0.04	1.00	-0.03	-0.03	-0.01	-0.03	0.12	-0.00
OSD.longitude	-0.18	0.02	-0.04	0.01	-0.03	1.00	1.00	-0.04	-0.03	0.06	-0.01
OSD.latitude	-0.18	0.02	-0.04	0.01	-0.03	1.00	1.00	-0.04	-0.03	0.06	-0.01
OSD.height [ft]	-0.13	0.02	-0.15	-0.01	-0.01	-0.04	-0.04	1.00	-0.05	-0.06	-0.02
OSD.pitch	0.01	-0.01	-0.59	-0.07	-0.03	-0.03	-0.03	-0.05	1.00	0.03	0.00
OSD.roll	-0.07	0.16	0.11	0.00	0.12	0.06	0.06	-0.06	0.03	1.00	0.01
OSD.yaw	-0.09	0.04	0.05	0.05	-0.00	-0.01	-0.01	-0.02	0.00	0.01	1.00

Figure 42: Correlation Matrix of Features

In the dataset, a correlation coefficient of 1.0 between latitude and longitude denotes a perfect positive linear relationship. This indicates that while latitude grows, longitude also increases correspondingly, and vice versa. Additionally, there's a noteworthy relationship between the elevator control input (RC.elevator) and the pitch angle of the drone (OSD.pitch), albeit in different directions. Increasing the elevator control input, such as pulling back on the control stick to elevate the nose of the drone, decreases the pitch angle (nose points downward). Conversely, decreasing the elevator control input, such as pushing forward on the control stick to lower the nose, raises the pitch angle (nose points upward). This negative association is expected since the elevator control alters the drone's pitch, and changes in elevator control input naturally led to opposing adjustments in pitch angle to maintain stability during flight.

Figure 43 shows the basic statics of the dataset and figure 44 shows the distribution of each feature.



	OSD.flyTime	RC.aileron	RC.elevator	RC.throttle	RC.rudder	OSD.longitude	OSD.latitude	OSD.height [ft]	OSD.pitch	OSD.roll
count	71522.000000	71522.000000	71522.000000	71522.000000	71522.000000	71522.000000	71522.000000	71522.000000	71522.000000	71522.000000
mean	401.039496	1022.424387	1142.007466	992.592671	1027.862937	79.930692	6.992722	10.994662	0.575854	-0.686762
std	274.853530	43.307291	232.260488	234.718867	223.484092	0.081706	0.097768	19.031982	4.284285	2.265847
min	0.000000	364.000000	364.000000	364.000000	364.000000	79.913877	6.972471	-6.800000	-23.600000	-19.600000
25%	171.025000	1024.000000	1024.000000	1024.000000	1024.000000	79.914263	6.972960	1.600000	-1.200000	-1.600000
50%	357.100000	1024.000000	1024.000000	1024.000000	1024.000000	79.914529	6.974190	3.900000	1.000000	-0.500000
75%	598.800000	1024.000000	1265.000000	1024.000000	1024.000000	79.915820	6.974440	9.800000	2.400000	0.400000
max	1176.300000	1684.000000	1684.000000	1684.000000	1684.000000	80.353385	7.498461	115.800000	27.400000	22.300000

Figure 43 : Statistics of Dataset

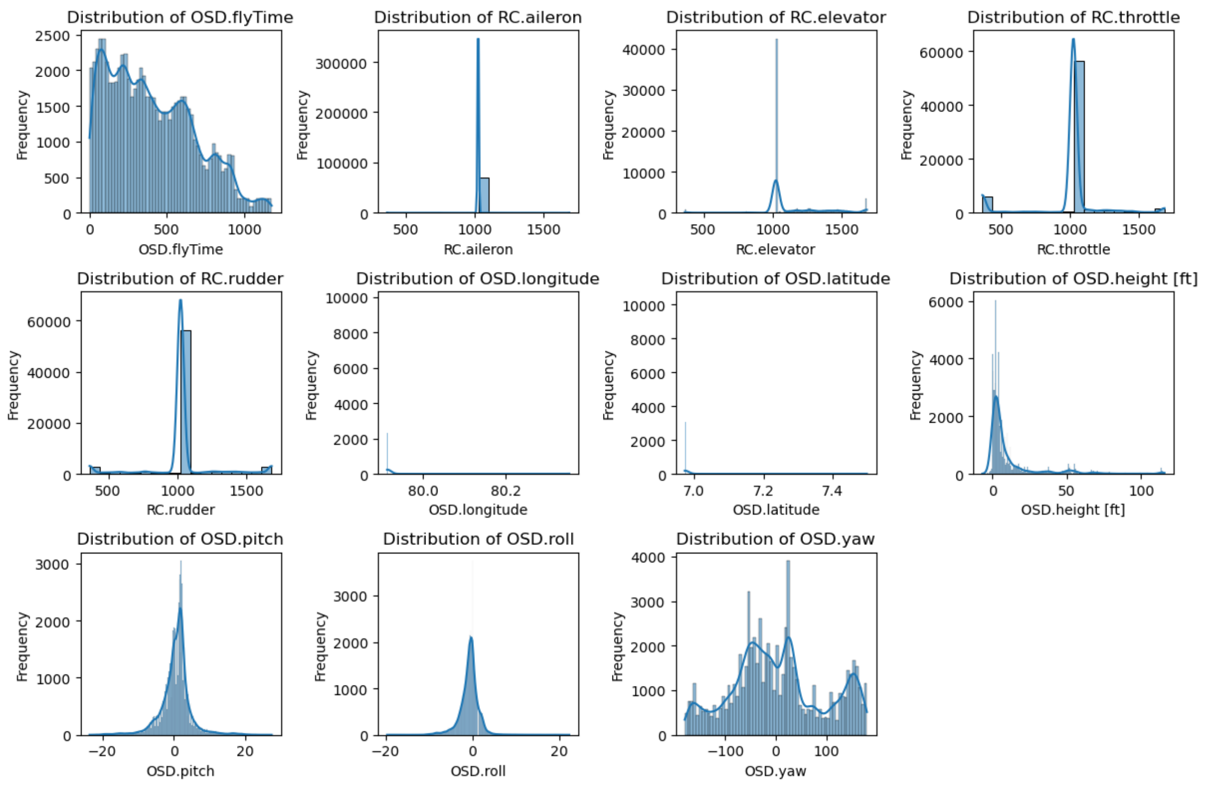


Figure 44: Distribution of Dataset

Height, Pitch, and Roll Distribution: The height, pitch, and roll values have essentially normal distributions, indicating that they are reasonably evenly distributed within their respective ranges. This shows that these variables may have a wider range of values and are likely influenced by multiple factors leading to their distribution.

Aileron, Elevator, Throttle, and Rudder: These variables illustrate that most of the data points are centered around specified value ranges, approximately 1000 in this case. This concentration suggests that these qualities may have restricted fluctuation or tend to take on specific values more frequently. Certain qualities may be restrained by operational constraints or are controlled within specified ranges during drone operations.

Based on these observations, it appears that the dataset comprises a varied range of data, with certain variables demonstrating higher fluctuation than others. The concentration of values around certain ranges for aileron, elevator, throttle, and rudder suggests that these elements may have set or limited operational ranges, whereas the regularly distributed height, pitch, and roll values signal more variability in these aspects of drone flying. Further study and domain knowledge would be needed to adequately assess the implications of these distributions for the dataset.

4.2 Result of Models

The training of the models utilized an Intel Core i9 CPU running at a clock speed of 3.7 GHz, along with an RTX 3080 graphics card and 32 GB of DDR4 RAM. The training duration ranged from 15 minutes to 3 hours. The dataset is partitioned, with 80% allocated for training and 20% allocated for testing. Furthermore, a specific drone movement pattern was selected for the purpose of evaluating the model. Figure 45 depicts the actual trajectory of the drone's maneuvering.

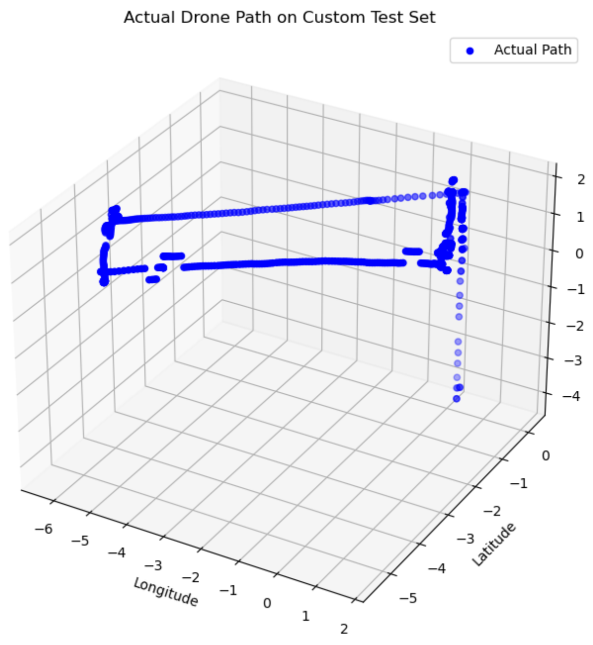


Figure 45 : Actual Trajectory of the drone

4.2.1 Multi-Output Regression Model

To assess the performance of the model, researchers rely on the Mean Squared Error (MSE), which quantifies the average squared difference between the model's predictions and the actual values. A lower MSE indicates higher performance, suggesting predictions that closely match actual results. Conversely, a greater MSE suggests bigger gaps between anticipated and actual values, indicating poorer model performance. In this situation, the MSE is 1541.7, suggesting the need for further improvement.

However, the claimed accuracy of -1441.79% appears to be erroneous, as accuracy levels normally fall between the range of 0% to 100%. Visual representations in figures 46 demonstrate the model's anticipated trajectory based on testing data. These graphics demonstrate the difference between the model's predictions and the real-world movement of the drone. Overall, the model's performance is rated poor, indicating the need for modification and additional optimization.

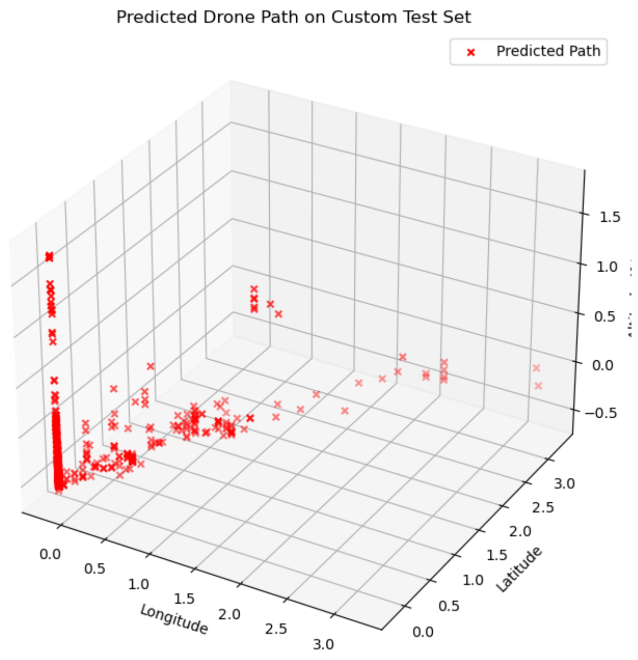


Figure 46 :Predicted path of multi-output regression model

4.2.2 Support Vector Machine Model

The SVM model generated similarly bad outputs, replicating the performance of the multi-output regression model. Reported accuracy on the test dataset stands at -1440.99%, with an MSE of 1540.99. Figures 47 display the model's anticipated trajectory based on the testing data. Additionally, Figure 48 displays the error fluctuation of each output feature over time. Notably, as compared to other characteristics, yaw exhibits a larger error variance over time. Yaw represents the rotation of the drone around its vertical axis. This movement can be impacted by many external factors such as wind speed and direction, which may not be effectively recorded by the model, resulting to higher error variance.

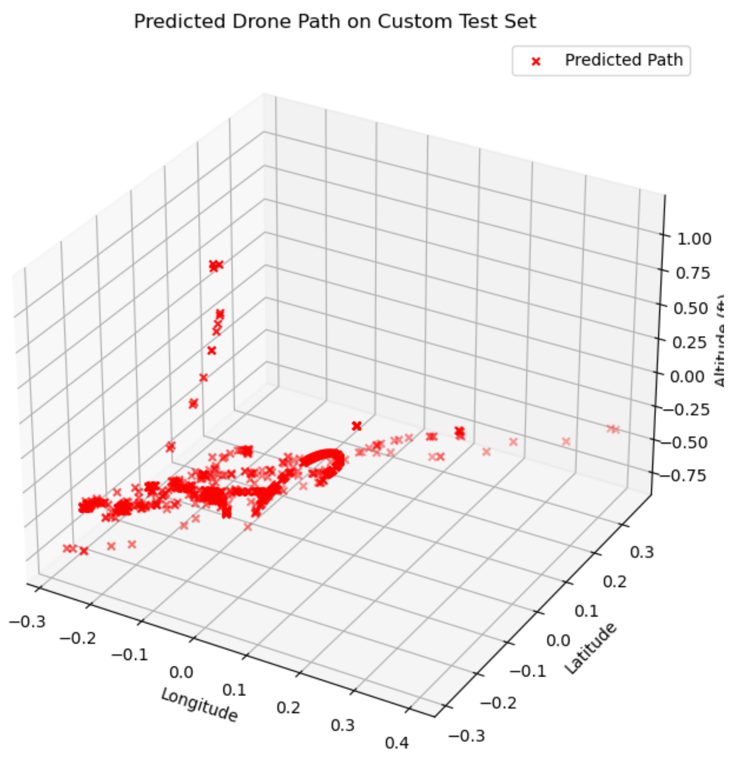


Figure 47 : Predicted path of SVM model

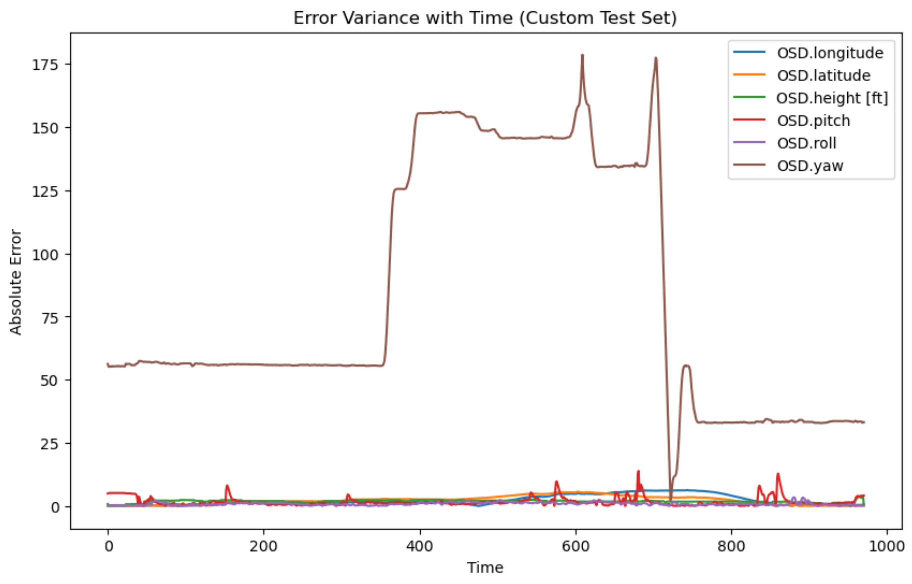


Figure 48 : Error variation of SVM model

4.2.3 Neural Network Model

Hyperparameter adjusting is a vital step in optimizing neural network models. To do this, RandomizedSearchCV is applied to explore a range of hyperparameters and determine the optimum combination for the model. The results of this search, including the optimal parameters, are presented in Table 5.

Table 5 : Hyper Parameters for Neural Network Model

Parameter	*Value*
neurons	64
learning_rate	0.001
layers	2
epochs	100
dropout_rate	0.3
batch_size	64

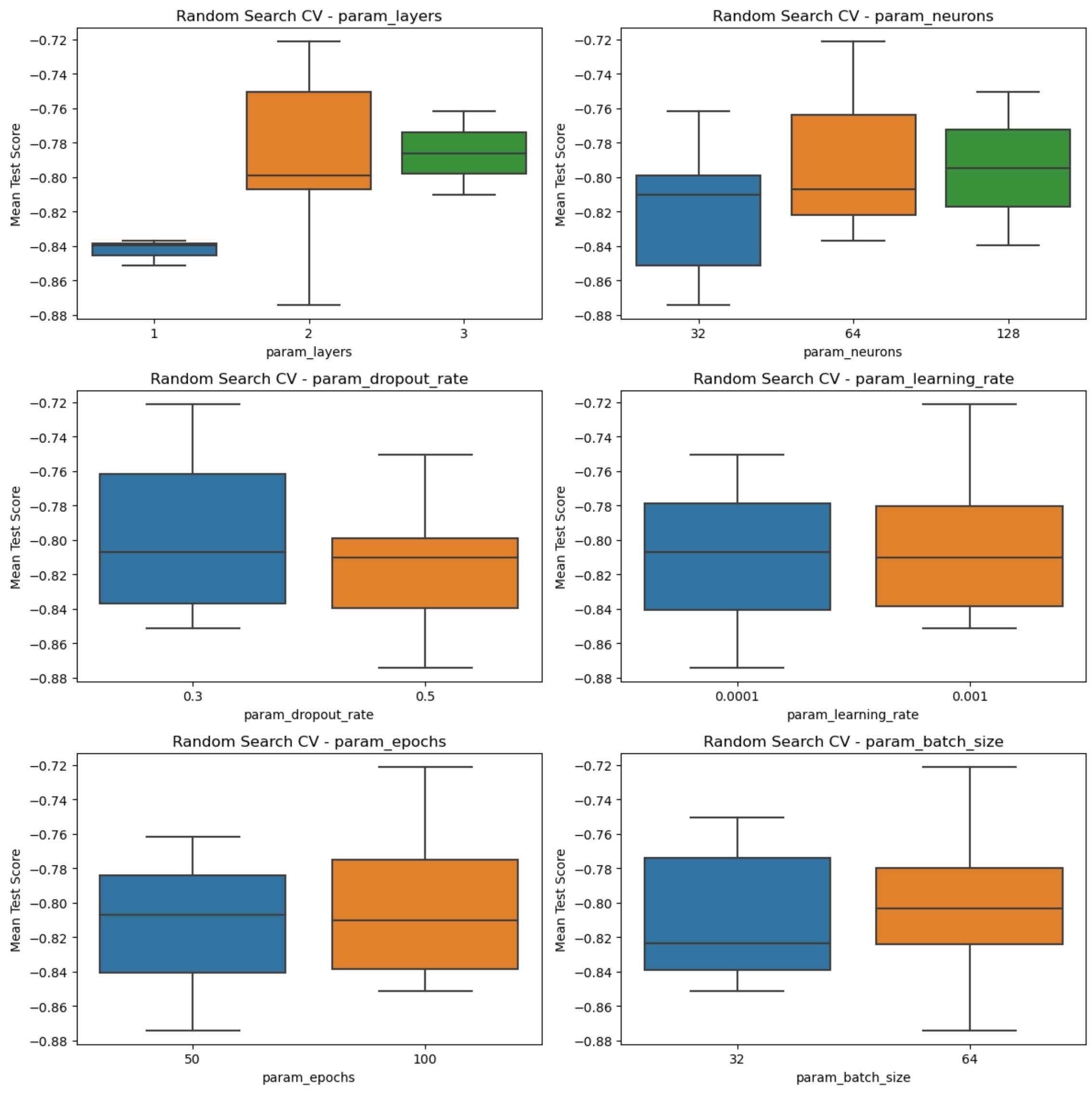


Figure 49 :Box plot of the mean test scores

Figure 49 displays boxplots of the mean test scores for a single hyperparameter obtained during the random search cross-validation. The mean test scores refer to the average performance metrics obtained by evaluating a test dataset. The x-axis indicates the values of the hyperparameter being displayed, while the y-axis represents the mean test result.

When compared to both the multi-output regression and SVM models, this model displayed considerably improved accuracy. With an MSE value of 0.72 and a model accuracy of 17%, it surpassed its peers. The figure 50 displays the error variations of each output feature, while figure 51 presents all output variables in a single diagram. It demonstrates that yaw has a greater rate of error variation compared with other output variables. Figure 52 shows the predicted path of the drone maneuver.

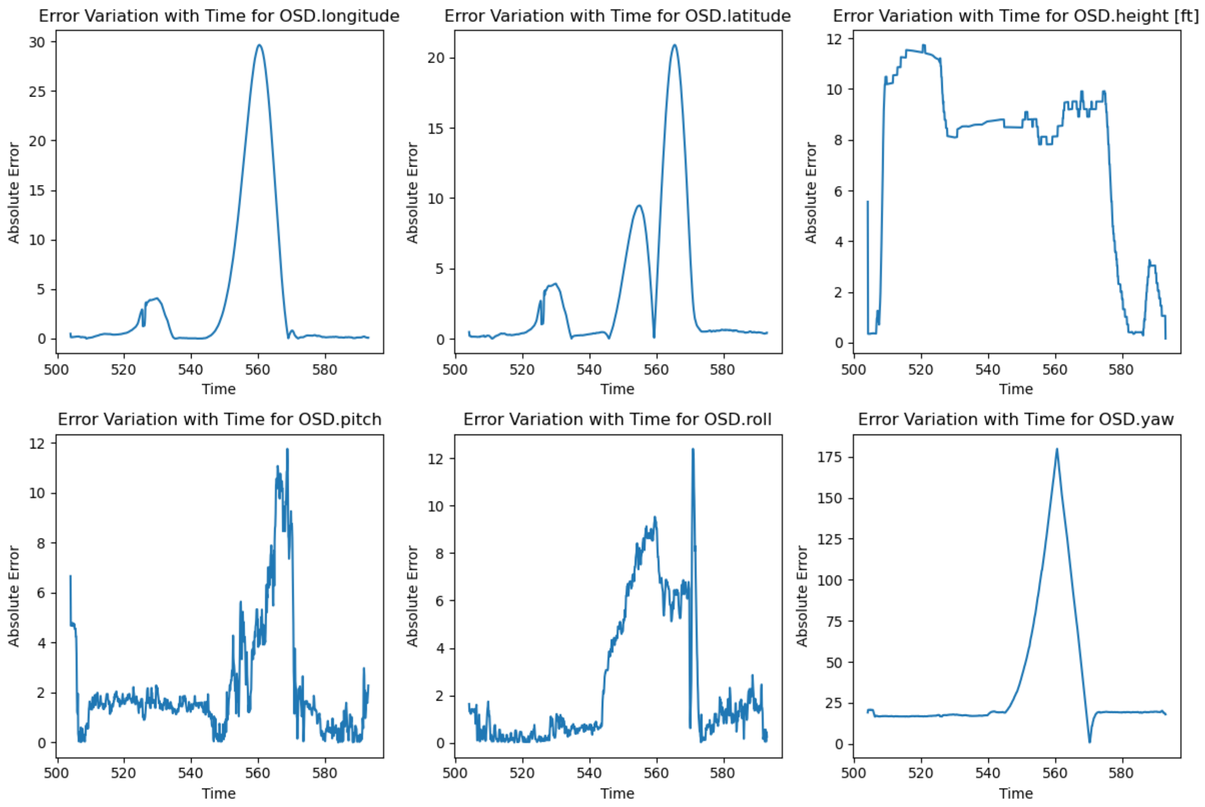


Figure 50 : Neural Network Model Error Variation

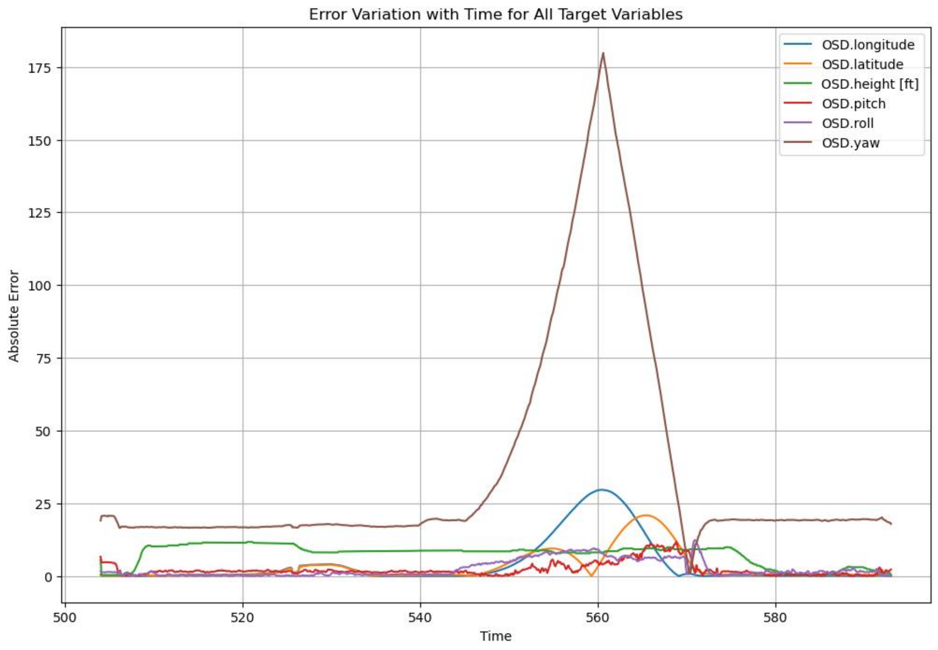


Figure 51 : Error variance of all target variables in the neural network model

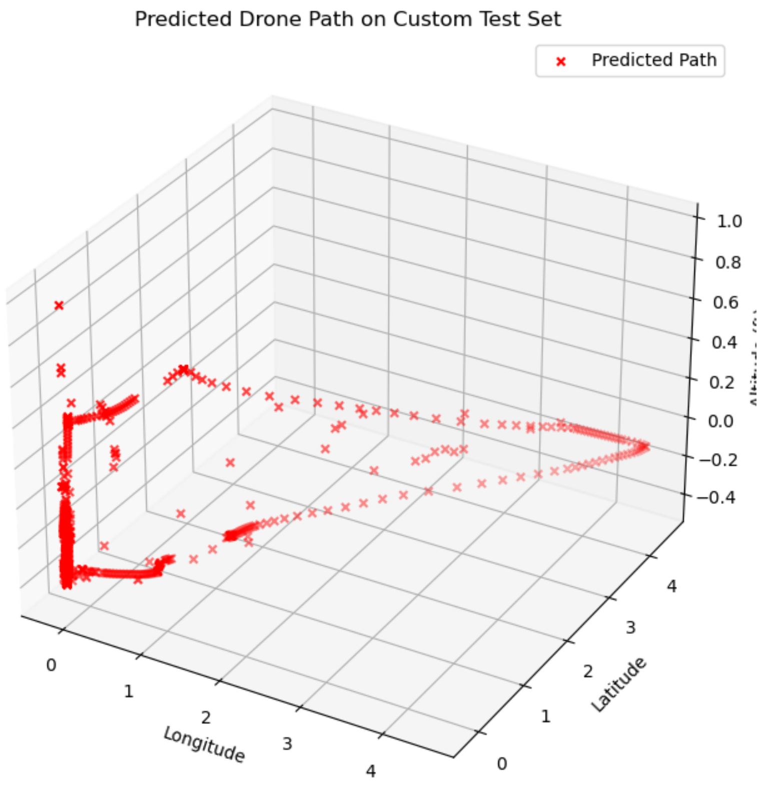


Figure 52 : Predicted Path of Neural Network Model

4.2.4 Conventional Neural Network Model (CNN)

Utilizing RandomizedSearchCV, appropriate hyperparameters were discovered for the CNN model. These optimal parameters are detailed in Table 06. The variance in error across output characteristics is displayed in Figure 53, suggesting successful prediction for all output aspects except yaw, which demonstrates significant error variation. This mismatch is related to the distribution of yaw values, as demonstrated in Figure 44, which is not normalized compared to other output features. The projected path of the test data reveals an MSE value of 0.65, with a model accuracy of 78% approximately. Significantly, the model displays greater accuracy compared to alternative models. Figure 54 shows the predicted path of the CNN model.

Table 6 : Hyper Parameter for CNN model

Parameter	*Value*
filters	128
kernel_size	5
activation	relu
optimizer	adam
learning_rate	0.001
epochs	50
batch_size	32

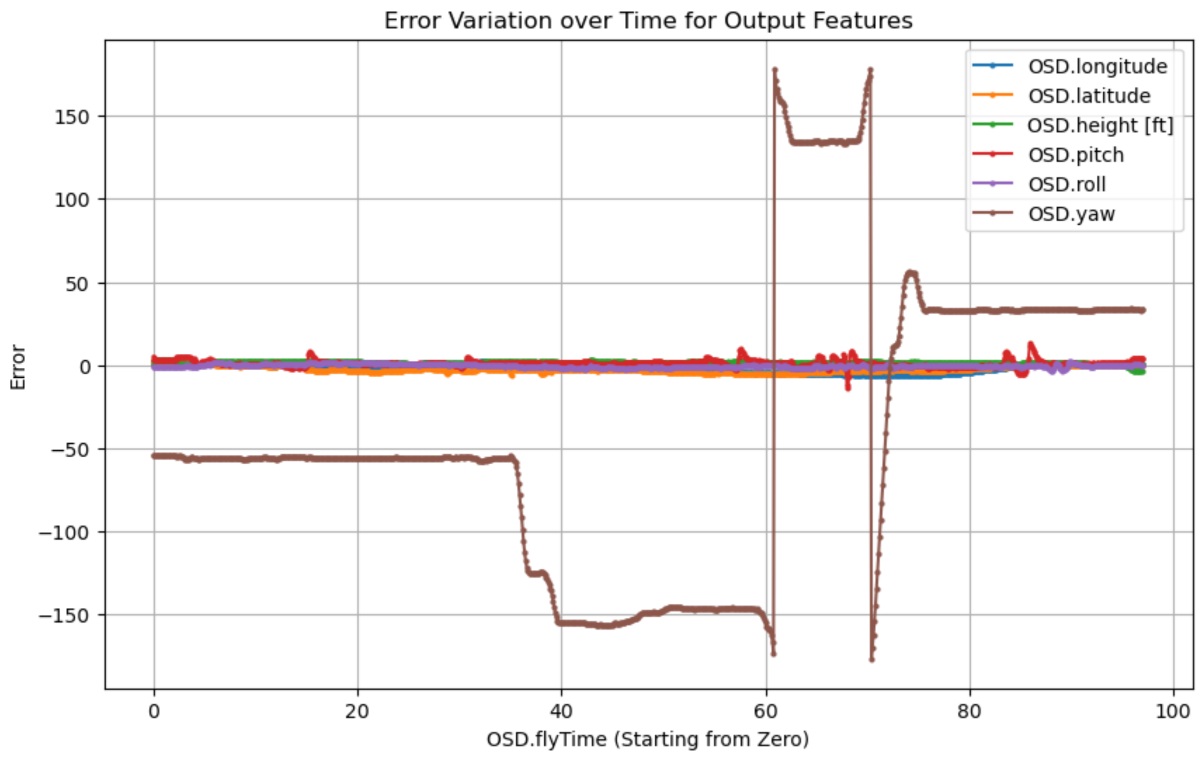


Figure 53 : Error variation of target features in CNN model

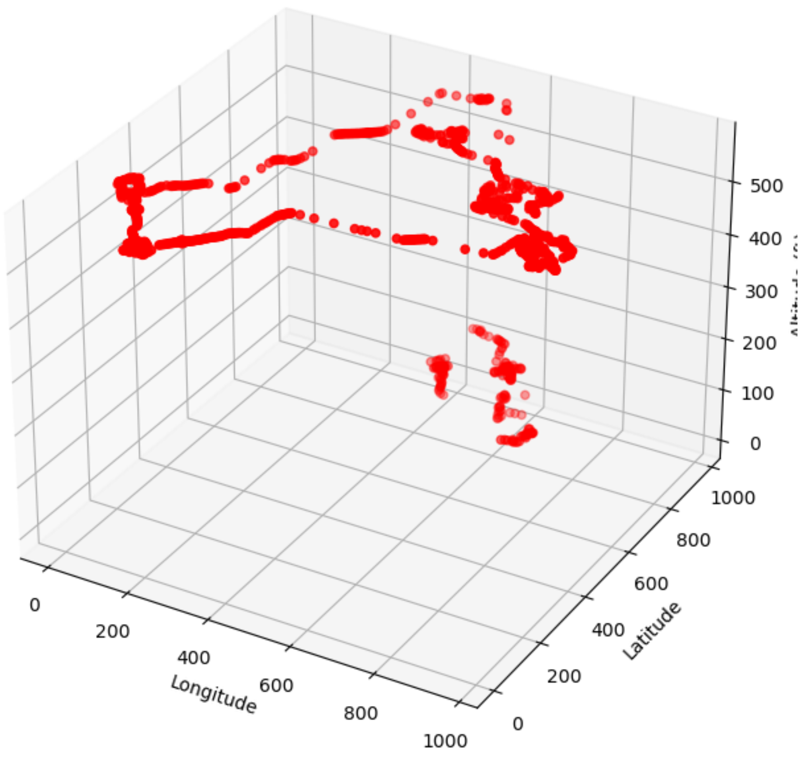


Figure 54 : Predicted path of CNN Model

The training and validation loss are shown in figure 55. CNN model, the loss curve records the performance improvement across training epochs by graphing the value of the loss function. This function quantifies the discrepancy between predicted and actual results. The curve tells if the model is learning effectively: a progressive reduction signifies convergence and learning, whereas variations or stagnation may suggest concerns like overfitting or underfitting. Monitoring this curve leads to adjustments to the model's architecture or hyperparameters and assures optimal performance. The loss curve for the CNN model training displays oscillations over the epochs, reflecting various levels of model performance. Initially, the loss reduces gradually, demonstrating that the model is learning and improving. However, during epochs 22 and 58, there is significant increase in loss, followed by swings in subsequent epochs. This behavior may suggest concerns like overfitting or instability in the training process.

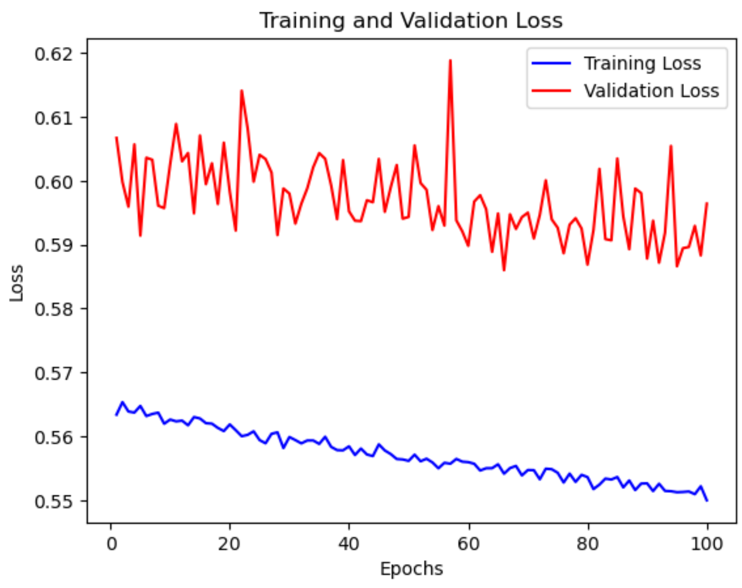


Figure 55 : Training and Validation loss of CNN model

Accuracy measures in a CNN model offer essential insights into its performance. R-squared (R^2) estimates the proportion of variance in the dependent variable explained by the model, ranging from 0 to 1, where larger values suggest a better fit. MAE computes the average absolute difference between expected and actual values, allowing a clear interpretation of prediction mistakes. Variance The term "explained," when expressed as a percentage, indicates the extent to which the model accounts for the variations in the target variable. These indicators together measure the accuracy and reliability of forecasts, driving future model development and decision-making processes. The box plot of the accuracy metric is displayed figure 56. The Numerical values are represented in table 6. These results demonstrate that the model works better for longitude, latitude, and height variables compared to pitch, roll, and yaw, as indicated by lower MSE, higher R-squared, and larger Variance Explained values. A comparison of each model is shown in table 8 and another sample test cases of the actual and predicted path are shown in figure 57.

Table 7 : Values of Accuracy metrics

Target Variable	*MSE*	*R-squared*	*MAE*	*Variance Explained*
OSD.longitude	0.68	0.33	0.26	63.26%
OSD.latitude	0.68	0.33	0.26	63.20%
OSD.height	0.69	0.33	0.44	64.83%
OSD.pitch	0.69	0.32	0.47	65.91%
OSD.roll	0.84	0.18	0.66	60.49%
OSD.yaw	0.81	0.18	0.72	52.89%

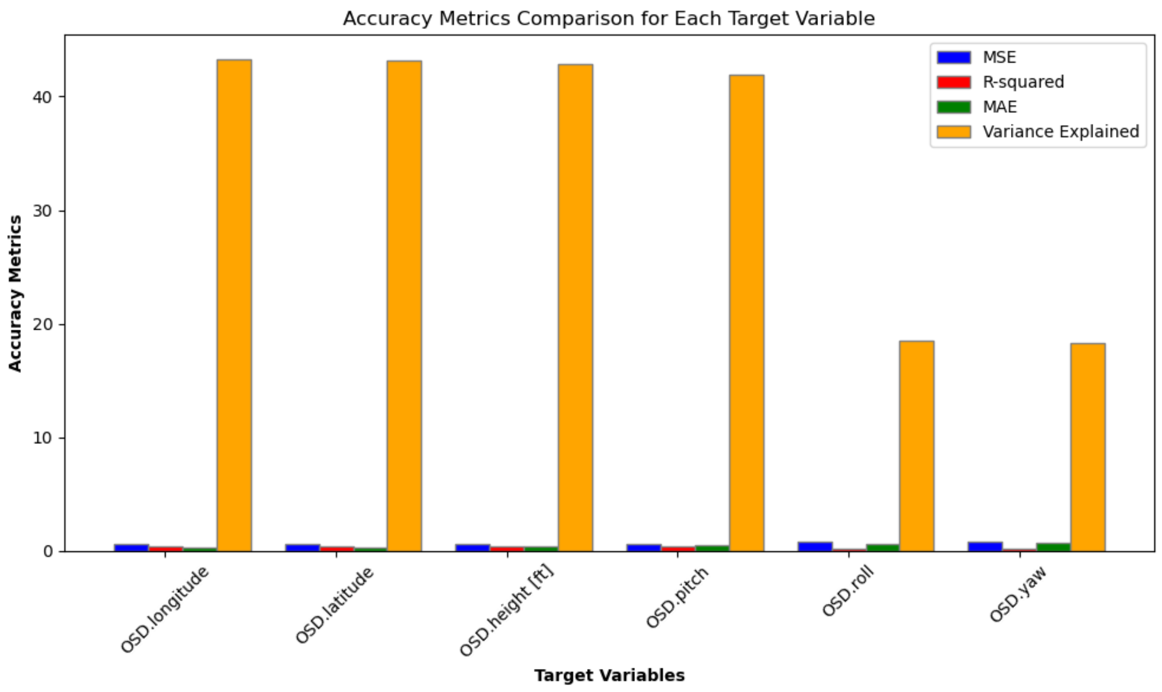
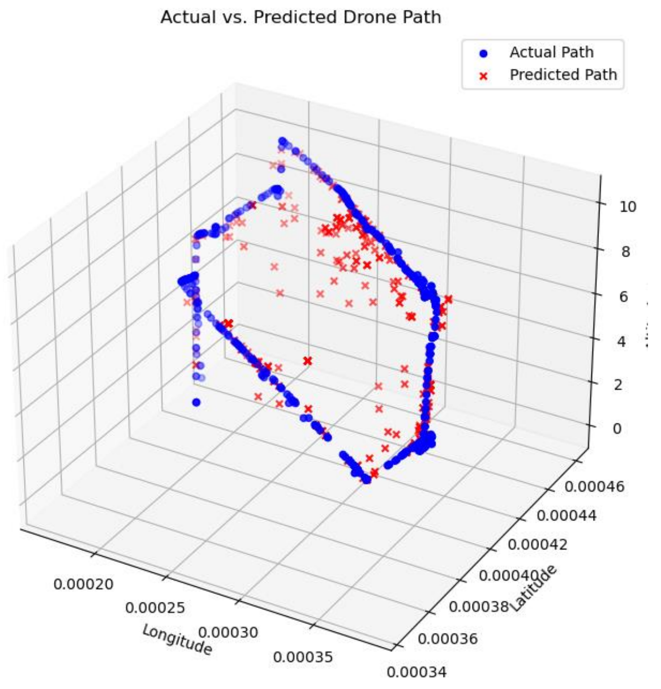


Figure 56 : Box plot of accuracy metrics

Table 8 : Comparison of models

Model	*Time takes for training*	*Accuracy*	*MSE*
Multi Output Regression Model	20 minutes	-1441.79%	1541.7
Support Vector Machine Model	38 minutes	-1440.99%	1540.99
Neural Network Model	97 minutes	17%	0.72
Conventional Neural Network Model	123 minutes	78%	0.65



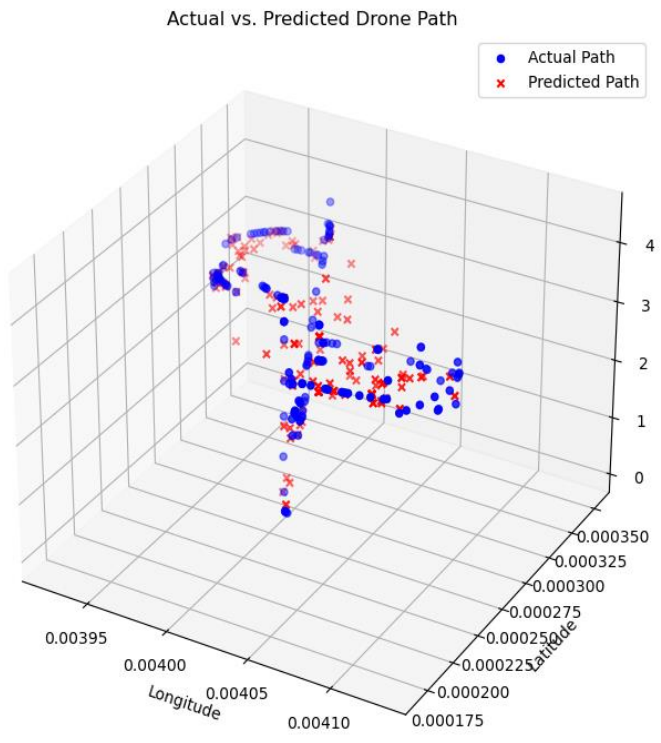


Figure 57 : Sample test cases

4.3 Chapter Summary

This section presents an overview of the dataset and evaluation for each model, along with a comparison between the models. It also outlines evaluation methodologies employed in the analysis.

CHAPTER 5: CONCLUSION AND FUTURE WORK

This chapter offers an overview of the research, a synthesis of the conclusions of the entire research work, the limitations of the research study, and an outline of further improvements.

5.1 Conclusion

The recent development of unmanned aerial vehicles has significantly altered various industries, such as surveillance, transportation, construction, and agriculture. However, evaluating drone behavior provides substantial challenges due to the complex interplay of elements like speed, altitude, orientation, and trajectory. Simulating drone dynamics has become significant as it enables researchers to test drones in complex circumstances that are unsafe or impractical. Furthermore, by utilizing simulation, drone pilots can receive thorough training in effectively managing various circumstances.

Traditionally, drone dynamics have been investigated with Newtonian and fluid dynamics concepts, involving various parameters such as force, gravity, propeller characteristics, and air density. Replicating certain drones becomes problematic due to the number of parameters involved. However, the provided model can efficiently simulate generic drones.

The research proposed a machine learning-powered drone dynamic model capable of simulating drone behaviors without the necessity of specialist subject expertise or elaborate laboratory setups. This strategy utilizes advanced AI models trained on huge datasets including real-world flight conditions. These datasets offer a wide range of movements, such as eight, circular, and lazy-eight patterns, indicating numerous sorts of drone motions. Multiple modeling approaches, including multi-output regression, support vector machine, neural network, and convolutional neural network (CNN), were applied in building the model. Among these, the CNN model had the highest accuracy, achieving 78%. Quantitative validation was accomplished by comparing predicted patterns with real-world drone maneuvers. One notable observation from the experiment is the high-level shape of the real drone trajectory and the anticipated path produced by the CNN algorithm are the same.

In contrast to non-AI-based methodologies, which typically require more attention to diverse variables, the model streamlines the simulation process. This simplification not only accelerates the modeling technique but also mitigates the complexity associated with earlier methodologies. The fundamental adaptability and learning capabilities of AI models, of CNNs, enable them to detect underlying patterns and correlations in drone dynamics from raw data.

Additionally, the employment of AI-driven methodologies has the potential to grow and adapt, enabling seamless interaction with multiple drone platforms and scenarios. The accessibility of drone simulation capabilities minimizes the necessity for specialist knowledge and sophisticated parameter calibration. This stimulates creativity and accelerates developments in the field of drone technology.

5.2 Limitation

One of the most demanding components of this research was the collecting of data. It required the recording of numerous drone maneuvers, a task that was handled by expert drone pilots. Special care had to be taken to ensure that data collecting happened under calm weather conditions to avoid mistakes caused by wind disturbances, which could ultimately hinder the accuracy of the model. Moreover, gathering data for specialized flying patterns, such as figure eights, circles, and lazy eights, proved to be extremely problematic. These moves need perfect forms, making it required for substantial effort and competence from the drone pilots.

Furthermore, the huge amount of data required for training the model presented another issue. Also, to manage such a vast dataset properly, high-performance computers are required.

5.3 Future work

As a future step for this study, it would be helpful to integrate qualitative validation alongside the existing quantitative validation approaches. While quantitative validation focuses on numerical measures to assess model performance, qualitative validation involves visual inspection and subjective evaluation of simulated results against real-world observations. To strengthen qualitative validation, expanding the range of drone maneuvers and gathering more extensive information including varied drone dynamics is required. Additionally, combining the established model with simulation tools such as MATLAB or Unity gives an interesting option for future research.

Furthermore, qualitative validation can be strengthened by comparing simulated trajectories, flying routes, and maneuvering patterns directly with real-world observations. This comparative analysis enables the discovery of any differences or inconsistencies between the simulation and actual drone behavior, offering significant insights into the model's accuracy and usefulness.

APPENDICES

Appendix A: URL for the dataset

https://drive.google.com/drive/folders/10xB9-
MF5I7moA2CzIiRXLz1Ckd61MPh8?usp=drive_link

Appendix B: URL for Code

https://drive.google.com/drive/folders/1MEipBUezFJ3HaiEdJ4NjBY7jz_E6SIJW?usp=sharin
g

REFERENCES

2009 9th International Conference on Electronic Measurement and Instruments. (2009), , I E E E.

“A Short History of Flight Simulation - Chuck’s Overview - X-Plane.Org Forum”. (n.d.). , available at: https://forums.x-plane.org/index.php?/forums/topic/54831-a-short-history-of-flight-simulation/ (accessed 14 November 2023).

Allerton, D. (n.d.). *Principles of Flight Simulation (Aerospace Series (PEP)).*

Berndt, J.S. (2004), *JSBSim: An Open Source Flight Dynamics Model in C++.*

Capello, E., Guglieri, G. and Quagliotti, F.B. (2009a), “UAVs and simulation: An experience on MAVs”, *Aircraft Engineering and Aerospace Technology*, Vol. 81 No. 1, pp. 38–50, doi: 10.1108/00022660910926890.

Capello, E., Guglieri, G. and Quagliotti, F.B. (2009b), “UAVs and simulation: An experience on MAVs”, *Aircraft Engineering and Aerospace Technology*, Vol. 81 No. 1, pp. 38–50, doi: 10.1108/00022660910926890.

“DJI Flight Log Viewer | Phantom Help”. (n.d.). , available at: https://www.phantomhelp.com/LogViewer/Upload/ (accessed 24 February 2024).

“DJI FLIGHT SIMULATOR”. (2021), , December, available at: https://www.dji.com/global/simulator (accessed 1 November 2023).

“Drones Transform Archaeology - Inside Unmanned Systems”. (n.d.). , available at: https://insideunmannedsystems.com/drones-transform-archaeology/ (accessed 12 November 2023).

Fernando, H.C.T.E., De Silva, A.T.A., De Zoysa, M.D.C., Dilshan, K.A.D.C. and Munasinghe, S.R. (2013), “Modelling, simulation and implementation of a quadrotor UAV”, *2013 IEEE 8th International Conference on Industrial and Information Systems, ICIIS 2013 - Conference Proceedings*, pp. 207–212, doi: 10.1109/ICIInfS.2013.6731982.

“Flight Simulator X For Pilots_ Real World Training”. (n.d.). .

Fritzson, P.A. (2004), *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*, IEEE Press.

“How Drones are being used in Disaster Management? - Geoawesomeness”. (n.d.). , available at: https://geoawesomeness.com/drones-fly-rescue/ (accessed 11 November 2023).

Hwangbo, J., Sa, I., Siegwart, R. and Hutter, M. (2017), “Control of a Quadrotor with Reinforcement Learning”, doi: 10.1109/LRA.2017.2720851.

“Introduction – FlightGear Flight Simulator”. (n.d.). , available at: https://home.flightgear.org/about/ (accessed 16 November 2023).

ITTC-Recommended Procedures Full Scale Measurements Manoeuvrability Full Scale Manoeuvring Trials Procedure. (n.d.). .

Jahan, F., Javaid, A.Y., Sun, W. and Alam, M. (2015), “GNSSim: An Open Source GNSS/GPS Framework for Unmanned Aerial Vehicular Network Simulation”, *ICST Transactions on Mobile Communications and Applications*, Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (ICST), Vol. 2 No. 6, p. 150091, doi: 10.4108/eai.11-8-2015.150091.

Javaid, A.Y., Sun, W. and Alam, M. (2015), “Single and Multiple UAV Cyber-Attack Simulation and Performance Evaluation”, *EAI Endorsed Transactions on Scalable Information Systems*, European Alliance for Innovation, Vol. 2 No. 4, pp. 1–11, doi: 10.4108/sis.2.4.e4.

Kadhim, Z.S., Abdullah, H.S. and Ghathwan, K.I. (2022), “Artificial Neural Network Hyperparameters Optimization: A Survey”, *International Journal of Online and Biomedical Engineering*, International Association of Online Engineering, Vol. 18 No. 15, pp. 59–87, doi: 10.3991/ijoe.v18i15.34399.

Kok, J.M. and Chahl, J.S. (2015), “A low-cost simulation platform for flapping wing MAVs”, *Bioinspiration, Biomimetics, and Bioreplication 2015*, Vol. 9429, SPIE, p. 94290L, doi: 10.1117/12.2084142.

Mahesh, B. (2018), “Machine Learning Algorithms-A Review”, *International Journal of Science and Research*, doi: 10.21275/ART20203995.

Mairaj, A., Baba, A.I. and Javaid, A.Y. (2019a), “Application specific drone simulators: Recent advances and challenges”, *Simulation Modelling Practice and Theory*, Elsevier B.V., 1 July, doi: 10.1016/j.simpat.2019.01.004.

Mairaj, A., Baba, A.I. and Javaid, A.Y. (2019b), “Application specific drone simulators: Recent advances and challenges”, *Simulation Modelling Practice and Theory*, Elsevier B.V., 1 July, doi: 10.1016/j.simpat.2019.01.004.

“Microsoft extends AirSim to include autonomous car research - Microsoft Research”. (n.d.). , available at: https://www.microsoft.com/en-us/research/blog/autonomous-car-research/ (accessed 16 November 2023).

Mora-Soto, M.E., Maldonado-Romo, J., Rodríguez-Molina, A. and Aldape-Pérez, M. (2021), “Building a realistic virtual simulator for unmanned aerial vehicle teleoperation”, *Applied Sciences (Switzerland)*, MDPI, Vol. 11 No. 24, doi: 10.3390/app112412018.

Orbea, D., Moposita, J., Aguilar, W.G., Paredes, M., León, G. and Jara-Olmedo, A. (2017), “Math model of UAV multi rotor prototype with fixed wing aerodynamic structure for a flight simulator”, *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 10324 LNCS, Springer Verlag, pp. 199–211, doi: 10.1007/978-3-319-60922-5_15.

Patrik, A., Utama, G., Gunawan, A.A.S., Chowanda, A., Suroso, J.S., Shofiyanti, R. and Budiharto, W. (2019), “GNSS-based navigation systems of autonomous drone for delivering items”, *Journal of Big Data*, SpringerOpen, Vol. 6 No. 1, doi: 10.1186/s40537-019-0214-3.

Punjani, A. and Abbeel, P. (2014), *Machine Learning for Helicopter Dynamics Models*.

Purwono, Ma'arif, A., Rahmaniar, W., Fathurrahman, H.I.K., Frisky, A.Z.K. and Haq, Q.M.U. (2022), "Understanding of Convolutional Neural Network (CNN): A Review", *International Journal of Robotics and Control Systems*, Association for Scientific Computing Electronics and Engineering (ASCEE), Vol. 2 No. 4, pp. 739–748, doi: 10.31763/ijrcs.v2i4.888.

Ray L. (n.d.). *Brief History of Flight Simulation*.

"REAL DRONE SIMULATOR". (n.d.). , available at: https://www.realdronesimulator.com/ (accessed 25 November 2023).

Ribeiro, R., Ramos, J., Safadinho, D., Reis, A., Rabadão, C., Barroso, J. and Pereira, A. (2021), "Web ar solution for uav pilot training and usability testing", *Sensors*, MDPI AG, Vol. 21 No. 4, pp. 1–32, doi: 10.3390/s21041456.

Sandaruwan, D., Jayasundara, M., Kodikara, N. and Pitigala, S. (2019), "MACHINE LEARNING BASED APPROACH TO SIMULATE DRONE DYNAMICS RELATED TO FIGURE OF EIGHT MANEUVERING PATTERN", *European Journal of Computer Science and Information Technology*, Vol. 7 No. 5, pp. 16–25.

Sandaruwan, D., Kodikara, N., Radeeshani, P., Mahima, K.T.Y., Suduwella, C., Pitigala, S. and Jayasundara, M. (n.d.). *User Perceive Realism of Machine Learning-Based Drone Dynamic Simulator*, *IJACSA) International Journal of Advanced Computer Science and Applications*, Vol. 14.

Schmid, L., Gerharz, A., Groll, A. and Pauly, M. (2022), "Machine Learning for Multi-Output Regression: When should a holistic multivariate approach be preferred over separate univariate ones?"

Shah, S., Dey, D., Lovett, C. and Kapoor, A. (2017), "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles".

"Support for Mavic Mini - DJI". (n.d.). , available at: https://www.dji.com/global/support/product/mavic-mini (accessed 23 February 2024).

"TensorFlow". (n.d.). , available at: https://www.tensorflow.org/ (accessed 2 March 2024).

"The INSTITUTE OF DRIVER EDUCATION RESEARCH". (n.d.). , available at: http://ider.org.uk/ (accessed 24 February 2024).

"Uses of Drones in Military: An Overview - Drones and UAV News". (n.d.). , available at: https://droneinternationalexpo.com/blog/uses-of-drones-in-military/ (accessed 12 November 2023).

Vogeltanz, T. (2016), "A Survey of Free Software for the Design, Analysis, Modelling, and Simulation of an Unmanned Aerial Vehicle", *Archives of Computational Methods in Engineering*, Springer Netherlands, Vol. 23 No. 3, pp. 449–514, doi: 10.1007/s11831-015-9147-y.

Vogeltanz, T. and Jašek, R. (2015), "FlightGear application for flight simulation of a mini-UAV", *AIP Conference Proceedings*, Vol. 1648, American Institute of Physics Inc., doi: 10.1063/1.4912769.

“Welcome to GEOG 892 - Geospatial Applications of Unmanned Aerial Systems | GEOG 892: Unmanned Aerial Systems”. (n.d.). , available at: https://www.e-education.psu.edu/geog892/node/508 (accessed 13 November 2023).

“Worldwide commercial drone market size by region 2025 | Statista”. (n.d.). , available at: https://www.statista.com/statistics/878022/global-commercial-drone-market-size-by-region/ (accessed 2 December 2023).