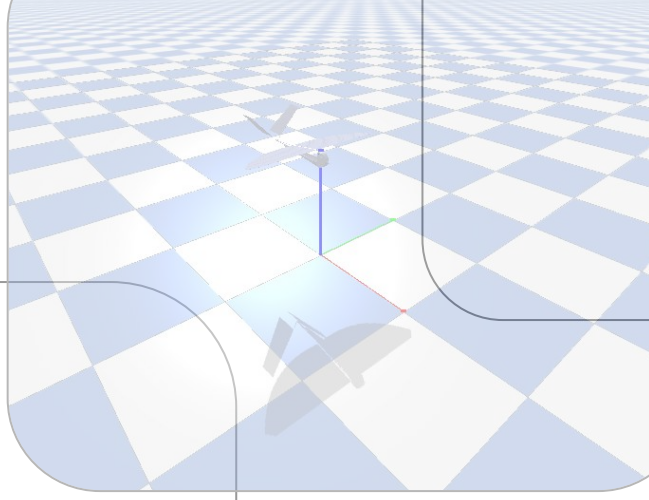
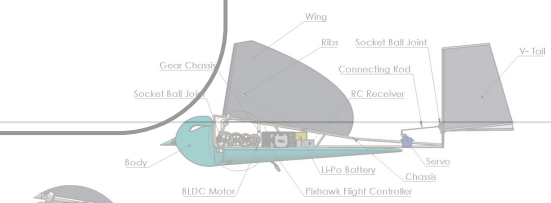
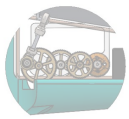
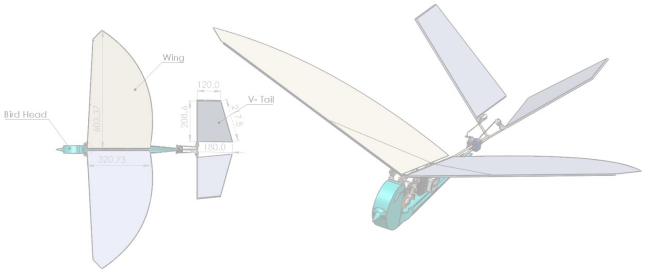


Deep Learning Based Control for Autonomous Ornithopter





DETAIL B
SCALE 2 : 5



-Dharambir Poddar

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

Deep Learning Based Control for Autonomous Ornithopter

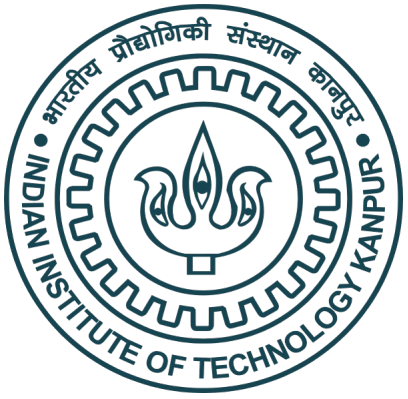
A thesis submitted in fulfilment of the requirements

for the degree of Master of Technology

by

Dharambir Poddar

19201262



DEPARTMENT OF AEROSPACE ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

May 2024

Certificate

It is certified that the work contained in this thesis entitled **"Deep Learning Based Control for Autonomous Ornithopter"** by **Dharambir Poddar** has been carried out under my supervision and that it has not been submitted elsewhere for a degree.

Prof. Debopam Das
Professor
Department of Aerospace Engineering
Indian Institute of Technology Kanpur

Prof. Indranil Saha
Associate Professor
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur

May, 2024

Declaration

This is to certify that the thesis titled **"Deep Learning Based Control for Autonomous Ornithopter"** has been authored by me. It presents the research conducted by me under the supervision of **Prof. Debopam Das** and **Prof. Indranil Saha**.

To the best of my knowledge, it is an original work, both in terms of research content and narrative, and has not been submitted elsewhere, in part or in full, for a degree. Further, due credit has been attributed to the relevant state-of-the-art and collaborations with appropriate citations and acknowledgments, in line with established norms and practices.

Dharambir Poddar

Roll No. 19201262

Department of Aerospace Engineering

Indian Institute of Technology Kanpur

Abstract

Name of the student: **Dharambir Poddar** Roll No: **19201262**

Degree for which submitted: **M.Tech.** Department: **Aerospace Engineering**

Thesis title: **Deep Learning Based Control for Autonomous Ornithopter**

Thesis supervisors: **Prof. Debopam Das** and **Prof. Indranil Saha**

Month and year of thesis submission: **May 2024**

Have you ever wondered about nature's precision and adaptability? Millions of years of evolution have sculpted creatures perfectly for survival, even without understanding the universe's physics. Similarly, in the realm of machine learning, particularly Reinforcement Learning, we witness a parallel process of exploration, learning, and evolution.

The research begins with the construction of an ornithopter(robotic bird) using techniques like 3D printing and pultrusion. The focus is on designing aerodynamic wings that generate both lift and thrust. By incorporating the perforation feature observed in real bird wings, a 12% improvement in efficiency is achieved. Furthermore, a novel bio-inspired mechanism is developed to generate diverse wing tip motion, enhancing the ornithopter's agility to replicate the agile flight of a bird.

The study then integrates intelligence into the ornithopter. A Recurrent Neural Network (RNN) based controller is designed using post-processed flight data, enabling the ornithopter to learn from past experiences and make better future decisions. Then we discuss the integration of a Reinforcement Learning (RL) strategy that allows the ornithopter to handle novel scenarios. Training is conducted using a digital model in a simulated environment like OpenAI Gymnasium, preparing the ornithopter for real-world challenges.

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Prof. Debopam Das and Prof. Indranil Saha. Their guidance wasn't simply about providing solutions but truly fostered the spirit of discovery. In the spirit of the proverb,

"Give a man a fish, and you feed him for a day. Teach a man to fish, and you feed him for a lifetime. Give a man a taste of fish, and he'll learn how to fish for himself"

they ignited my passion for bio-inspired research, machine learning and reinforcement learning. Their unwavering support, extending even to my start-up endeavors, and their belief in my abilities have been invaluable.

I am sincerely grateful to my senior, Joydeep Bhowmik, for sharing his knowledge in manufacturing techniques and flight data modeling. My thanks also goes to Javed Mohd for his continuous support and guidance on the complexities of research life. Special appreciation goes to Anil Pal for his tireless commitment to building physical models and setting up experiments. To my fellow labmates – Pawan Kumar Karn, Praveen Nuvvula, Amar Yadav, Rijin Rajan, Keesanth, Ashwani, Gosu Satish, Deepak Rawat, Shivam, Nitin, Sidhharth and Mann– thank you for your unwavering motivation and for making the lab a truly enjoyable place to work. I'd also like to thank collaborators and interns Ansh Gangwar, Bidisha Mukherjee, Suman Banerjee and Vanu Pratap Akheriya – working with you was a pleasure. I extend my appreciation to Vikram Saini, Vivek , Akash Kumar Sing, Nikhil Sing, and Harshit Verma for their insightful discussions and equipment support.

My heartfelt thanks goes to my family for their unwavering love, encouragement, and unwavering belief in me.

Finally, once again, to my supervisors – thank you for your exceptional mentorship and opportunities. IIT Kanpur, thank you for providing such a rich and intellectually stimulating environment.

Contents

List of Figures

List of Tables

Abbreviations

UAV	Unmanned Aerial Vehicle
FWUAV	Flapping Wing Unmanned Aerial Vehicle
RL	Reinforcement Learning
DNN	Deep Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
BPTT	Backpropagation Through Time
GYM	Gymnasium
MAV	Micro Aerial Vehicle
MDP	Markov Decision Process
BLDC Motor	Brushless DC Motor
PWM	Pulse Width Modulation
IMU	Inertial Measurement Unit
GPS	Global Positioning System

Symbols

V_1	Upstream velocity (m/s)
V_2	Downstream velocity (m/s)
N	Number of holes
A_{total}	Total projected area (m^2)
A_1	Hole inlet area (m^2)
A_2	Hole outlet area (m^2)
θ	Angle between the axis of hole and the vertical axis of the plate (°)
$\hat{n}$	Direction of area vector
F_{xFF}	Force on plate perpendicular to the velocity when inlet flow is in a converging section (g)
F_{yFF}	Force on the plate parallel to the velocity when the flow is in a converging section(g)
F_{xBF}	Force on plate perpendicular to the velocity when inlet flow is in the diverging section(g)
F_{yBF}	Force on the plate parallel to the velocity when the flow is in the diverging section (g)
F_{xFP}	Force on plate perpendicular to the velocity for flat plate (g)
F_{yFP}	Force on the plate parallel to the velocity for flat plate (g)
C_{lg}	Lift gain coefficient
C_{tg}	Thrust gain coefficient
σ	Activation function
df	Data Frame
$\hat{y}$	Predicted Output

List of Publications

Publications

1. **D. Poddar**, A. Gangwar, and D. Das. Bio-inspired perforated flapping wings. **International Conference on Intelligent Unmanned Systems**, 2023.

2. **D. Poddar**, N. Kumar, J. Mohd., and D. Das. Aerodynamics of flapping fin inspired from manta ray. In Fluid Mechanics and Fluid Power, Volume 2, pages 513–523, Singapore, 2024. **Springer Nature** Singapore. ISBN 978-981-99-5752-1.

3. G. Wadhwa, A. Tambe, J. Mohd, **D. Poddar**, R. Rajan, G. Mandal, P. Kadam, S. Saha, and D. Das. Regime transitions in a laminar film flowing over a cylinder. Bulletin of **the American Physical Society**, 2023

4. R. Rajan, B. P. Akherya, J. Mohd., **D. Poddar**, G. Wadhwa, S. Saha, and D. Das. Flow falling from slit and circular hole over a horizontal cylinder. In Fluid Mechanics and Fluid Power, Volume 6, pages 261–271, Singapore, 2024. **Springer Nature** Singapore. ISBN 978-981-99-5755-2

Patents and Copyright

1. Design Software Architecture for Virtual Ecosystem for Enabling Revenue, Open Platform.
   Copyright Reg.No-**L145397/2024**

2. A Biomimicry Scout Camera System for Monitoring Activities of Objects.
   Patent No. **IN 480305**

   Featured in **Dainik Jagran** newspaper for invention.

3. Modular Design of a Mechanism for Neutralizing Flying Objects. (Submitted)

4. Modular Design of a Mechanism for Ejection of a Neutralizer. (Submitted)

Achievements & Awards

- Selected for grant from **STPI**(Software Technology Parks of India, Government Ministry) in the FinBlue Startup Challenge by TiE Chennai. (*Y'24*)
- Selected by **IIM B-NSRCEL** for Startup LP-25. (*Y'24*)
- **Head Maintenance Secretary Award**, for outstanding dedication & service, Hall-7, IIT-K (*Y'24*)
- Selected for "Future Green Aviation Technologies Seminar for Young Scientists and Specialists" at **TSAGI, Moscow**, Russia. (*Y'23*)
- **Mentorship award** for HAL 44^{th} batch Training Program. (*Y'23*)
- **Mentorship award** for HAL 45^{th} batch Training Program. (*Y'22*)

Dedicated to the most innovative person in the world, my Father

"Think Unthinkable"

Chapter 1

Introduction

1.1 Motivation

The development of autonomous ornithopters using reinforcement learning is a captivating area of research with the potential to revolutionize various fields. Ornithopters, inspired by the flight of birds, offer unique advantages in maneuverability, efficiency, and silent operation compared to traditional fixed-wing or rotary-wing aircraft. These characteristics make them ideal for applications in search and rescue, environmental monitoring, and surveillance in sensitive areas.

However, controlling the complex flapping-wing dynamics of ornithopters presents significant challenges. Traditional control approaches often struggle to model the intricate interactions between wing movements and aerodynamics, leading to instability and unpredictable behavior.

Reinforcement learning (RL) offers a promising approach to overcome these limitations. RL algorithms can learn effective control policies through trial and error, without requiring a precise mathematical model of the system. This makes them well-suited for complex and dynamic systems like ornithopters.

1.2 Benchmarks of Ornithopter Development

The earliest evidence of ornithopter design can be traced back to the 14th century through Leonardo da Vinci's codex pages, as depicted in figure 1.1. Several centuries later, in 1872,

the world experienced the first flight of a mechanical flapping-wing aircraft with Alphonse Penaud's rubber-powered model ornithopter, showcased in figure 1.1 in France.

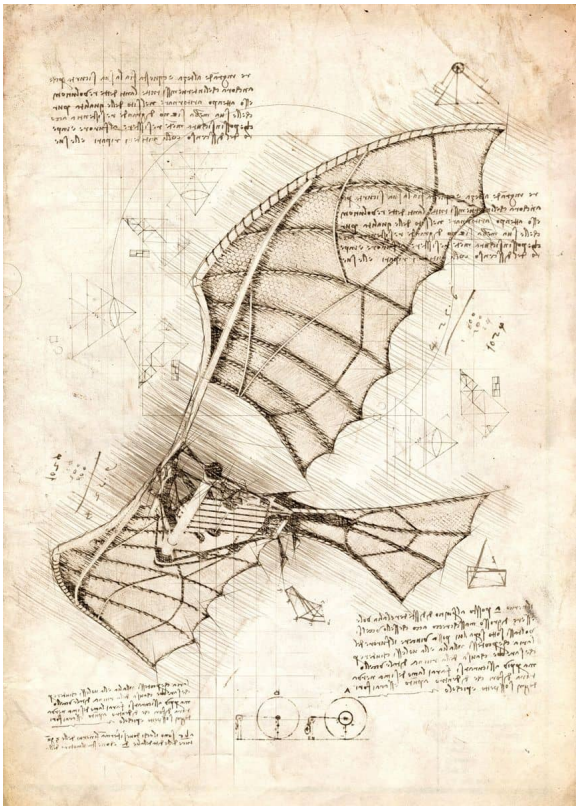


FIGURE 1.1: Ornithopter concept by Leonardo da Vinci

Several decades later, in 1894, Otto Lilienthal constructed a small wing-flapping glider powered by an engine, as illustrated in figure 1.2. Unfortunately, his attempts to fly the glider were unsuccessful. In a tragic turn of events, during a flight on August 9, 1896, Otto Lilienthal's glider collapsed, resulting in severe injuries. He succumbed to his injuries the following day in a Berlin hospital. This event is considered a significant setback in the history of aviation and a distinct blow to the progress of aerial flight. Prof. Dr. Erich von Holst (1908-1962) stands out as a pioneer who contributed significantly to the development of various rubber-powered flapping wing models. Employing slow-motion cameras, he meticulously observed the flights of these models. His extensive analysis extended to studying the flight patterns of seagulls, the brimstone butterfly, and dragonflies. Prof. Dr. Erich von Holst's dedication to this research is evident in the construction of numerous rubber-powered models, exemplified in figure 1.3

In the later years, specifically in 1930, Vincenz Chalupsky made notable contributions by constructing a series of birdlike ornithopters (figure 1.4). These innovative models

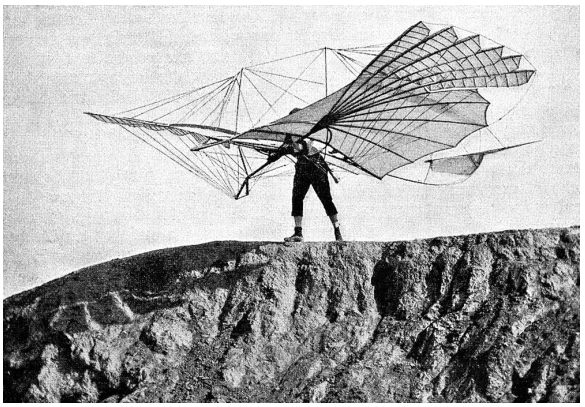


FIGURE 1.2: Otto Lilienthal Glider

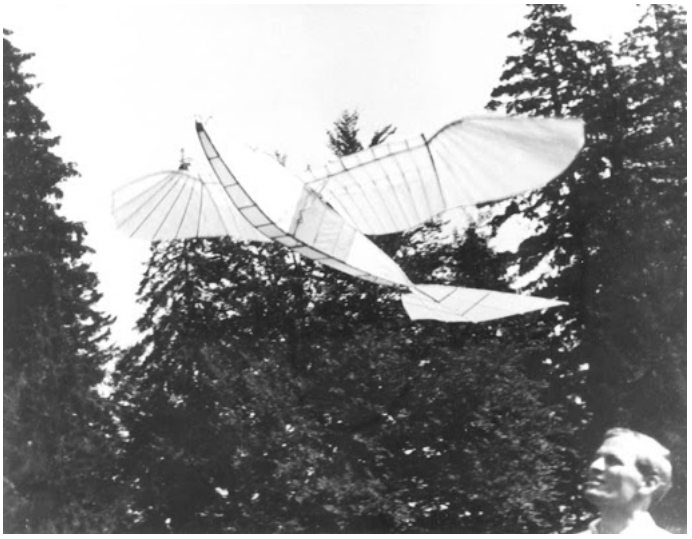


FIGURE 1.3: Prof. Erich von Holst model

were designed to be powered by compressed air or carbon dioxide. Despite being tailless, these ornithopters demonstrated steady flight, and their achieved climbing power remains noteworthy even by today's standards. In 1969, the Tim bird, credited to Albertini Prosper

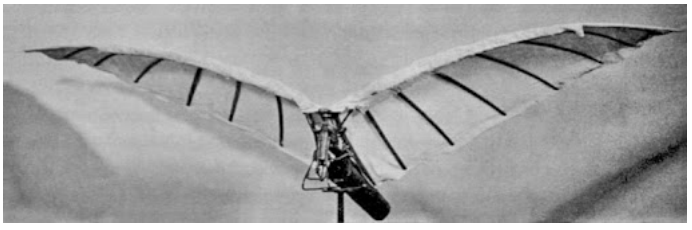


FIGURE 1.4: The flapping wing model of the Czech Cenek Chalupsky

and de Ruymbeke Gérard of France, marked a significant milestone as the first mass-produced rubber-powered flapping wing model figure 1.5. It featured simple membrane

flapping wings that introduced a novel and accessible design to a wider audience.

FIGURE 1.5: The Tim bird invented by Albertini Prosper and de Ruymbeke Gerard of France

1.3 State-of-The-Art Flapping Wing Flying Machines

1.3.1 The Delfly

The Delfly[KARÁSEK ET AL., 2018] (fig.1.6), developed by TU Delft, is a Micro Air Vehicle (MAV) with a bi-wing design, operated remotely. Demonstrates the ability to execute straight, horizontal, and hovering flights. Equipped with an onboard camera, the DelFly provides an excellent overview of its surroundings. In particular, it can identify targets and objects through vision technique. Additionally, a smaller version called the DelFly Micro, weighing only 3 grams with a wing span of 100mm, has been created by the team.

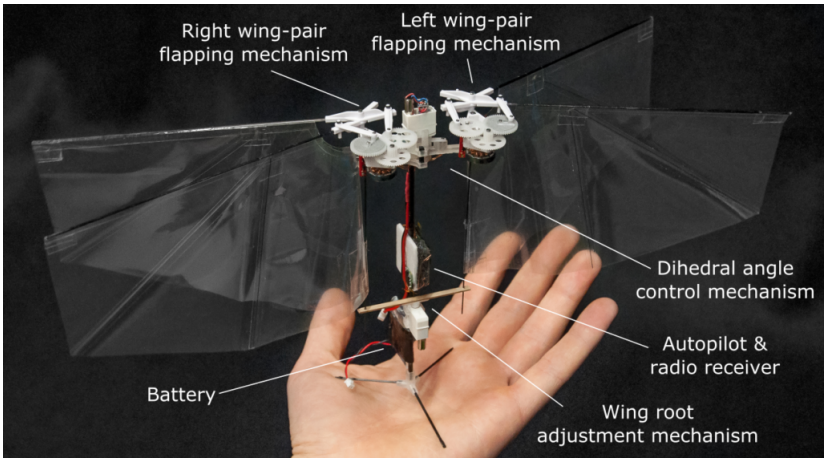


FIGURE 1.6: TU Delft, DelFly MAV

1.3.2 The Nano Hummingbird

The Nano Hummingbird(fig.1.7), also known as the Nano Air Vehicle (NAV), is a diminutive, remote-controlled aircraft crafted by AeroVironment Inc[HUM] . Developed in the United States, this aircraft not only resembles a real hummingbird in its body structure but also emulates its flight patterns. It achieves speeds of 17 km/h (4.7 m/s) and adeptly navigates in three axes of motion. With a wingspan spanning 160mm and a total flying weight of 19 grams, encompassing essential flight components such as batteries, motors, communication systems, and the video camera payload, the Nano Hummingbird showcases remarkable agility. Its capabilities include vertical ascent and descent, lateral movement in both directions, forward and backward flight, clockwise and anticlockwise rotation, along with the ability to hover steadily in midair.

FIGURE 1.7: Humming bird by AeroVironment

1.3.3 The Festo Swift Bird

Festo's BionicSwift [FESTO] (fig.1.8) is a bio-inspired robotic bird designed to explore advanced aerodynamics and autonomous flight technologies. This light weight flying robot (weighing only 42 grams) mimics the swift's flight, utilizing articulated wings with flexible, overlapping lamellae that resemble real feathers. This design allows for passive wing adjustments during flight, optimizing lift and drag for increased flight efficiency. The BionicSwift navigates autonomously within a defined airspace using an ultra-wideband (UWB) indoor vision system.So, it has a lot of room for improvement to make it capable of outdoor environment with robust controller.

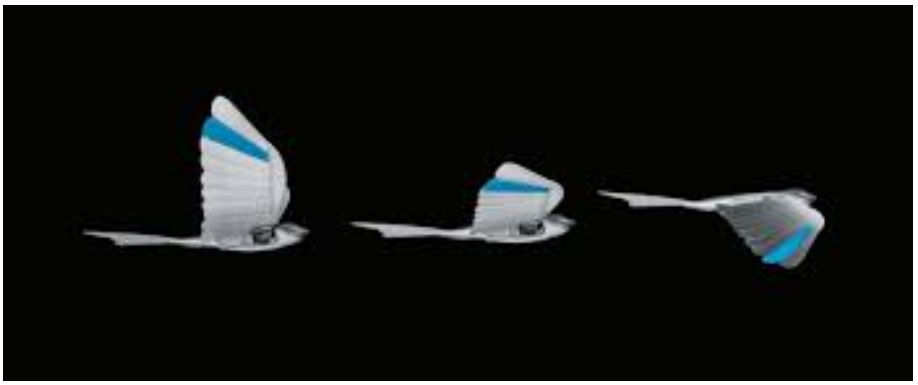


FIGURE 1.8: Swift Bird by Festo

1.3.4 The Silver Bat -India

In 2002, K. Nanda Kumar introduced the Silver Bat[K.NANDAN, 2002](fig.1.9), India's inaugural ornithopter. With a wingspan of 54 inches (1.37 m), this groundbreaking creation is propelled by an electric motor and operated through radio control. Figure (1.9) captures the ornithopter in flight, showing the early prototypes of its development stages. The successful flight of the Silver Bat earned it a place in the Limca Book of Records, recognizing it as a remarkable achievement.

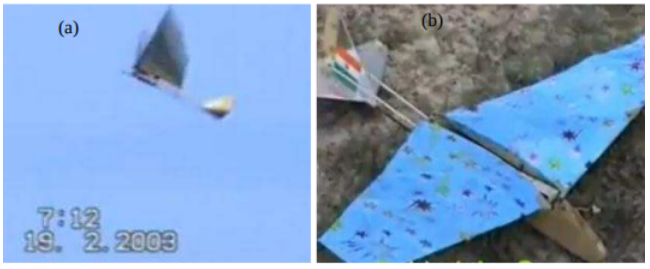


FIGURE 1.9: The first ornithopter of India (a) The Silver Bat in flight (b) One of its earlier prototype by Mr K Nanda Kumar

1.3.5 IITK Bird - India

Joydeep Bhowmik[BHOWMIK, 2017] and his team at IIT K have made significant contributions to ornithopter(fig.1.10) development, creating notable models such as Cleo and Raven. The Cleo, with a 160 cm wingspan and a weight of 450 grams, stands out for its flapping frequency of 4-6 Hz. Equipped with an onboard camera and flight controller, Cleo is manually controlled using a radio controller from a ground station. Its development, alongside other bird models, garnered attention in 2012.

On the other hand, the Raven showcases remarkable flight dynamics, featuring a minimum flapping frequency of 3.5 Hz and maintaining a consistent cruise speed of 5 m/s. The design principles are rooted in the theoretical framework outlined in the author's thesis, with precision improving as the prototype's dimensions increase. Initial projections suggested a power requirement of 9 watts, but during the actual cruise flight, it consumed 11 watts of electrical power. Notably, powered by a 500 mAh, 20 C Lithium polymer battery, the Raven achieves an impressive flight duration of up to 25 minutes.

FIGURE 1.10: IIT K bird development

1.3.6 Ultra-light Ornithopter

This study highlights recent advancements in small-scale flapping-wing micro aerial vehicles (MAVs), showcasing their expanded capabilities for autonomous flight control. The focus is on a 13-gram ornithopter[BAEK ET AL., 2011](fig.1.11) capable of autonomously flying towards a target without external assistance. The researchers developed compact control electronics weighing 1.0 gram, incorporating a microcontroller, inertial and visual sensors, communication electronics, and motor drivers for autonomous flight control.

To enhance efficiency, a simplified aerodynamic model of ornithopter flight was devised, reducing the complexity of the control system. The study emphasizes the use of onboard sensing and computation, presenting successful flight control towards a target. An innovative dead-reckoning algorithm was introduced to address temporary target loss, especially relevant for visual sensors with a narrow field of view. Notably, the 28 cm wingspan ornithopter achieved a landing accuracy within a 0.5 m radius from the target in

over 85% of attempts (N = 20), showcasing the effectiveness of the developed autonomous flight control system.

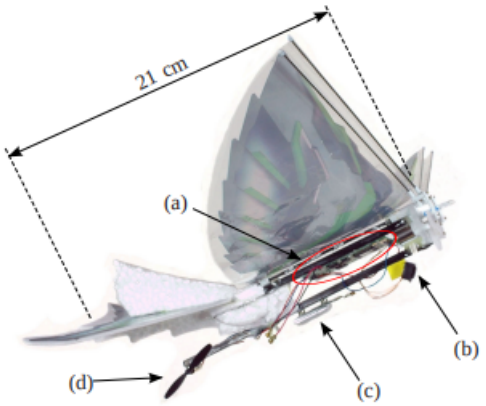


FIGURE 1.11: 13 gram ornithopter by University of California Berkeley

1.3.7 Autonomous Perching of Ornithopter

The article[ZUFFEREY ET AL., 2022] published in the Nature Communications in 2022 presents research on the autonomous perching capabilities of ornithopters, which are flapping-wing aerial vehicles inspired by birds and insects .The authors demonstrate successful experimental perching(fig.1.12) of ornithopters on branches, though the conditions were simplified to ease controller design .They note that optimized bio-inspired flight paths could improve perching reliability and reduce impact forces .This work contributes to the development of advanced perching capabilities for ornithopters, which could enable new applications for these bio-inspired aerial vehicles.

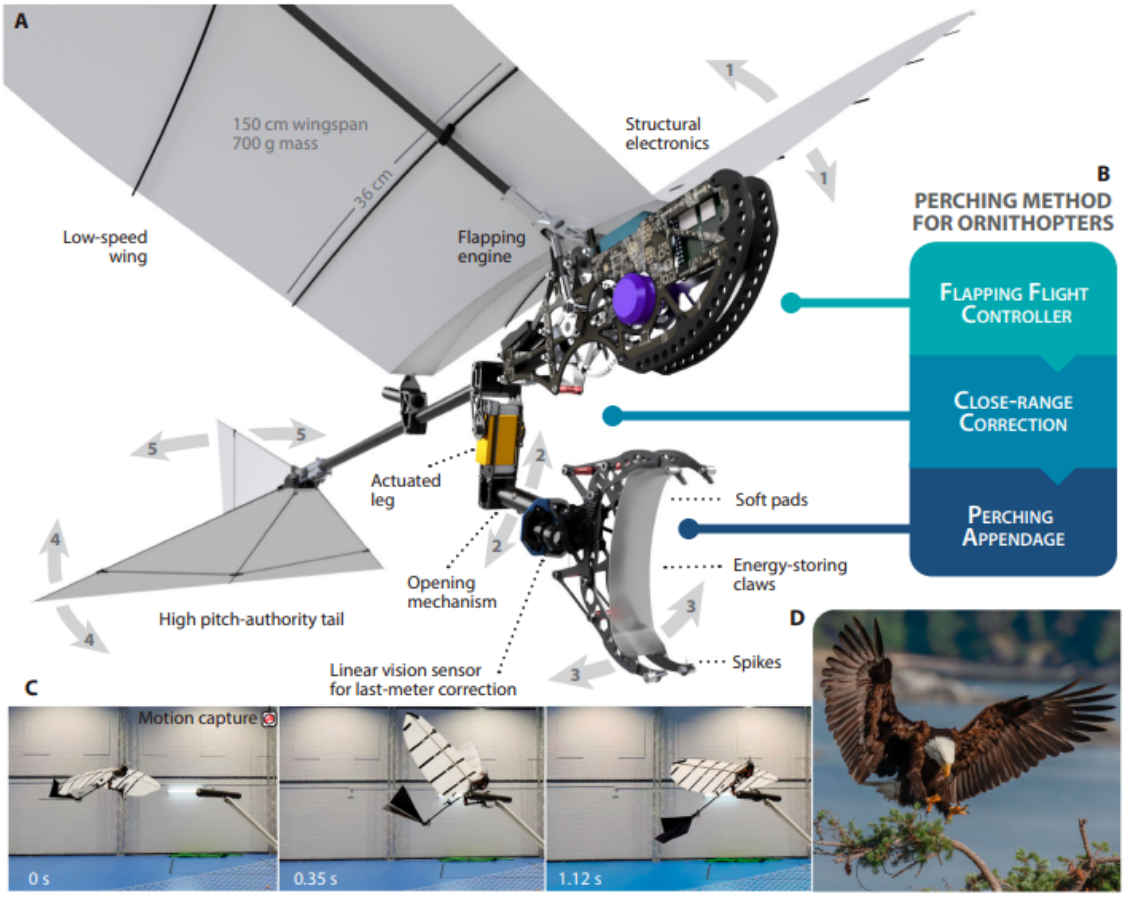


FIGURE 1.12: Overview and demonstration of the perching method

1.3.8 Quadcopter Control desing using RL

The cited paper[PANERATI ET AL., 2021] addresses this by introducing an open-source, OpenAI Gym-like simulator focused on multiple quadcopters. It utilizes the Bullet physics engine, aiming to bridge the gap by offering multi-agent and single agent simulations, vision-based reinforcement learning, and realistic physics modeling. The authors demonstrate its potential for both control theory (trajectory tracking, multi-robot flight) and reinforcement learning (stabilization) tasks.

The paper shows the capability of RL, and how it can be implemented in an aerial vehicle like the quadcopter.

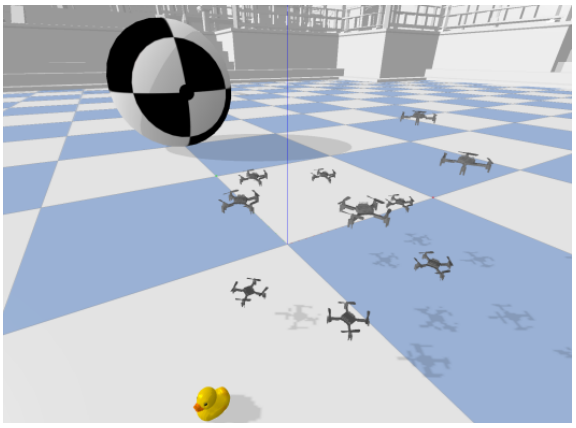


FIGURE 1.13: Training Multi-Quadrotor rendering in Gym Environment

1.3.9 Autonomous Helicopter flight via RL

The research paper[KIM ET AL., 2003] discusses policy invariance under reward transformations, focusing on reinforcement learning and policy search methods for large Markov Decision Processes (MDPs) and Partially Observable MDPs (POMDPs). The authors emphasize encoding symmetries directly into the model rather than requiring algorithms to learn them from scratch, highlighting the importance of model identification in helicopter body coordinates rather than spatial coordinates for effective control. The paper delves into the use of locally-weighted regression for state and action inputs, as well as the application of algorithms like gradient ascent and random-walk for weight optimization, emphasizing the need for appropriate scaling of derivatives and standard heuristics to ensure stability in the learning process. The paper further supports the possibility of using RL based control for an ornithopter.

FIGURE 1.14: Autonomous Helicopter with RL control

1.3.10 Design and Movement of a Three Leg Chair-Like Robot using RL

The paper [Inoue et al., 2024b] focuses on designing a chair-type asymmetric tripedal low-rigidity robot inspired by a three-legged chair character. The research delves into the robot's body design and gait generation, emphasizing aspects like stand-up motion, rewards, weights, and reset conditions during training. The study involves training multiple agents over epochs to learn standing up from different initial postures. Additionally, the research utilizes a physics simulator and specific algorithms for robot learning.
Author shows how a real problem can be formulated learning using RL. The paper clearly depicted the steps from building model and training in simulated Gym environment and then physical world directly. Which is very much essential for ornithopter learning algorithm.

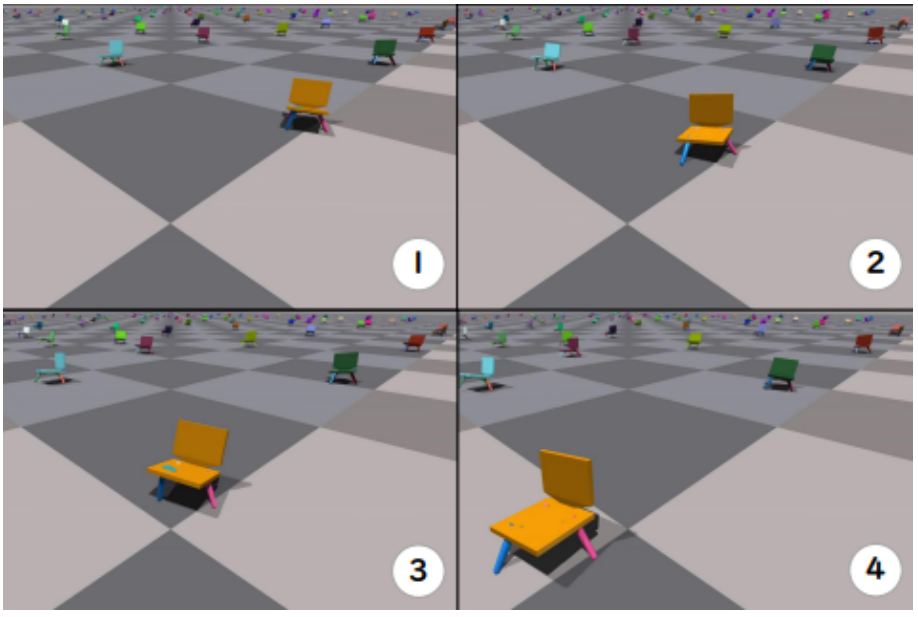


Figure 1.15: Training robot in simulator to learn walking

1.4 Organization of Thesis

The present work aims at improvement of an ornithopter model base on previous ornithopter Cleo build by joydeep Bhowmik [Bhowmik, 2017]. In the thesis we implemented different strategy to improvement the model to make it autonomous. Such as

- Modification of an ornithopter with different tail configurations. In Chapter 2, we will also discuss how to build a full-scale ornithopter from scratch.
- Improving wing efficiency of the ornithopter. In Chapter 3, we will discuss a novel perforated wing design and demonstrate how it can significantly increase lift and thrust.
- Developing a novel bio-inspired mechanism for generating diverse wing tip motion to enhance ornithopter agility (Chapter 4).
- Making the ornithopter autonomous. Chapter 5 covers data collection for developing a data-driven controller.
- Developing a Recurrent Neural Network-based control system for autonomy based on flight data (Chapter 6).
- Discussing the roadmap for building a virtual platform in OpenAI's GYM environment to train the ornithopter autonomously using Reinforcement Learning (Chapter 7).

Chapter 2

Development of Ornithopter and Measurement of Flight-Data

2.1 Introduction

The vast expanse of the sky has become a stage for observation, with drones playing a leading role in aerial surveillance. Quadrotors, the ubiquitous drones of our time, provide a cost-effective and efficient method to monitor vast areas. However, their limitations become apparent in specific applications like espionage, wildlife observation, and extended surveillance. In these scenarios, discretion, endurance, and agility are paramount, and here, a fascinating contender emerges - the ornithopter, a bio-inspired drone mimicking the flight of birds.

Ornithopters offer unique advantages that transcend mere curiosity, positioning them as viable alternatives for specific surveillance needs. Imagine a silent observer, its flapping wings blending seamlessly with the natural soundscape, unnoticed amidst its feathered counterparts. This unparalleled camouflage ability stems from the ornithopter's design, making it ideal for sensitive situations where discretion is crucial. But the benefits extend beyond acoustics. Imagine an aerial platform that consumes significantly less power than its counterparts, allowing for extended missions without frequent battery recharges. This remarkable energy efficiency, inspired by nature's masterful design, becomes invaluable in remote areas or time-sensitive scenarios. Moreover, the biomimetic design grants ornithopters exceptional agility and maneuverability. Unlike the rigid movements of quadrotors, these machines mimic the graceful acrobatics of birds, navigating complex environments with ease. This nimbleness proves invaluable in scenarios requiring precise maneuvers, such as

navigating densely forested areas or inspecting intricate structures. However, unlocking the full potential of the ornithopter requires overcoming significant challenges. Recreating the intricate aerodynamics of bird flight within a mechanical system demands understanding the complex, unsteady airflow generated by flapping wings. Furthermore, modeling the dynamic wing deformations that occur during flight adds another layer of complexity. These intricate aspects pose hurdles in developing efficient control algorithms, especially for achieving autonomous flight.
However, in this chapter, we will focus on the design principles of bird flight. At the end, we will conduct manual flight tests of ornithopter to collect data aimed at building a neural network-based, data-driven controller.

2.2 Development of Ornithopter

The first model built is called BluBird (fig.2.1), is developed to replicate the earlier bird [Bhowmik, 2017], We used miller sheet of thickness 0.5 micron with wingspan of 160 cm, used styrofoam for ribs and two stage gear reduction, with 4HZ flapping frequency. After the first trail, and learning from the failure we attempted developing a new version of ornithopter.

Figure 2.1: BluBird

Figure 2.2: Chidiya

The Chidiya (fig.2.2) is the hindi word for bird, call our second ornithopter model Chidiya. In Chidiya we used three stage gear reduction with 150 cm of wingspan and V-tail configuration. Chidiya is designed to operate at flapping frequency between 3Hz to 6Hz. The overall weight of Chidiya is 450 grams with battery(35 grams).

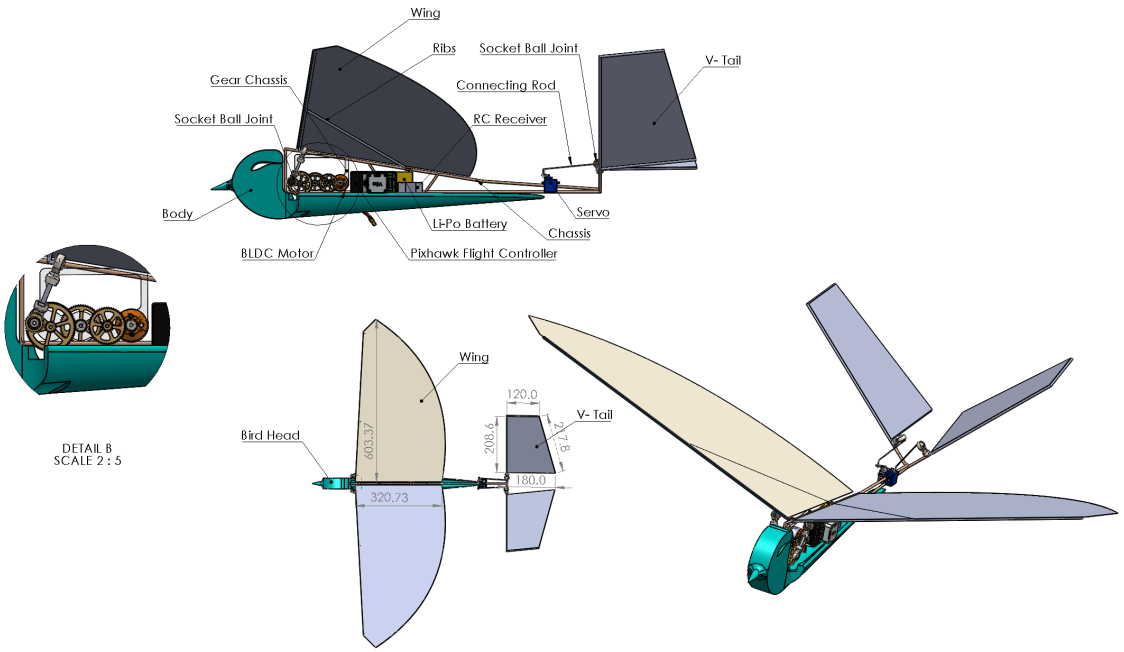


FIGURE 2.3: Chidiya CAD Assembly Model

2.3 Fabrication of Ornithopter Model

In our pursuit of creating an optimal aerial vehicle model, our primary focus is on achieving a lightweight construction while ensuring the durability of materials. To address this, we've chosen a high-density styrofoam for the fuselage, providing a balance between weight and structural strength. Carbon fiber tubes are strategically incorporated for reinforcement.

The wing design is a meticulous process, based on research BHOWMIK ET AL. [2013],we have adopted a semi-elliptical shape as shown in figure 2.4. For wing material, we've opted for a laminated low-density polythene to enhance wear and tear resistance during high-frequency flapping. In the previous model, we often encountered gear tooth failure due excessive torque jump from one gear stage to another. To overcome this challenge, a three-stage gear reduction system has been implemented to gradually enhance torque and

withstand load, but at the same time we compromised with power efficiency of ear train, due to friction loss.

Our gears are manufactured in-house using 3D printing technology, experimenting with materials like PLA, ABS, and Nylon. PLA exhibited stiffness but proved brittle, making it susceptible to bending stress. Moreover, its tendency to generate heat at higher speeds led to wear and tear. Consequently, we optimized the placement of PLA gears and introduced Nylon gears for higher rotation capabilities due to their superior wear resistance and tolerance for elevated temperatures. The gears, with a 0.8 module, were integrated into the gearbox with multiple stages.

The gearbox (figure2.3 in 'section B') is driven by a Brushless DC motor connected to a 16-teeth pinion which, through a carbon fiber shaft and bearing, eventually engages a 64-tooth gear. Total gear ratio of gear train is $40.5 : 1$. To convert rotary motion to flapping motion, a 4-bar mechanism is employed. A Universal joint is utilized to ensure smooth motion transmission from the gear shaft to the wing, maintaining a harmonious flapping motion.

For the fuselage, we have chosen high-density styrofoam throughout, providing structural integrity. To further reinforce the gearbox, a carbon fiber tube is incorporated, connecting it to both the fuselage and wing, ensuring a cohesive and durable design.

We have adopted a 90°V-Tail configuration with a trapezoidal shape for navigation and control. The movement of the tail is facilitated by specially crafted linkage rods and joints. These linkages are directly controlled by a servo motor for precise and responsive maneuvering.

2.4 Design Principle of Wings

Designing the wings for a flapping-wing unmanned air vehicle (FWUAV), poses unique challenges since both lift and thrust are generated solely through the flapping cycle of the wings. This distinguished from conventional aircraft design necessitates careful consideration of factors like low Reynolds numbers (typically ranging from $5 * 10^4$to 10^4) and the dynamic nature of flapping motion.

In contrast to established approaches in traditional aircraft design, the process for flapping-wing design lacks sophistication. Our methodology involved an iterative loop of theoretical

and experimental design, fabrication, testing, and refinement, proving to be the most realistic and effective pathway.

Drawing inspiration from a scale law derived from biological statistics by [Kang and Shyy, 2013], our conceptual design incorporated initial parameters like wing span, wing area, wing loading, aspect ratio, and beat frequency. This approach was essential due to the absence of a well-defined method for flapping-wing design.

Preliminary design involved experimental and theoretical studies [Yang et al., 2018b], exploring common features of unsteady low Reynolds number aerodynamics in flapping wings. Researchers highlighted the significance of both cord-wise and span-wise deflection in thin membrane-like wings for lift and thrust production during the flapping motion.

Research [Yang et al., 2018b] shows that increasing the taper ratio and reinforcing the stiffness of the inner part leads to an increase in the lift coefficient, while flexibility in the trailing edge of the wing increases the thrust coefficient. This suggests that the inner part of the wing primarily contributes to the generation of lift, while the outer part is crucial for thrust, dominated by deformations.

Consequently, an effective flapping-wing design should encompass a sufficiently large lifting surface in the inner part to generate lift and a flexible trailing edge in the outer part for substantial thrust. Achieving a preferable taper ratio is facilitated by shaping the planform into an approximately trapezoidal configuration as shown in figure2.4.

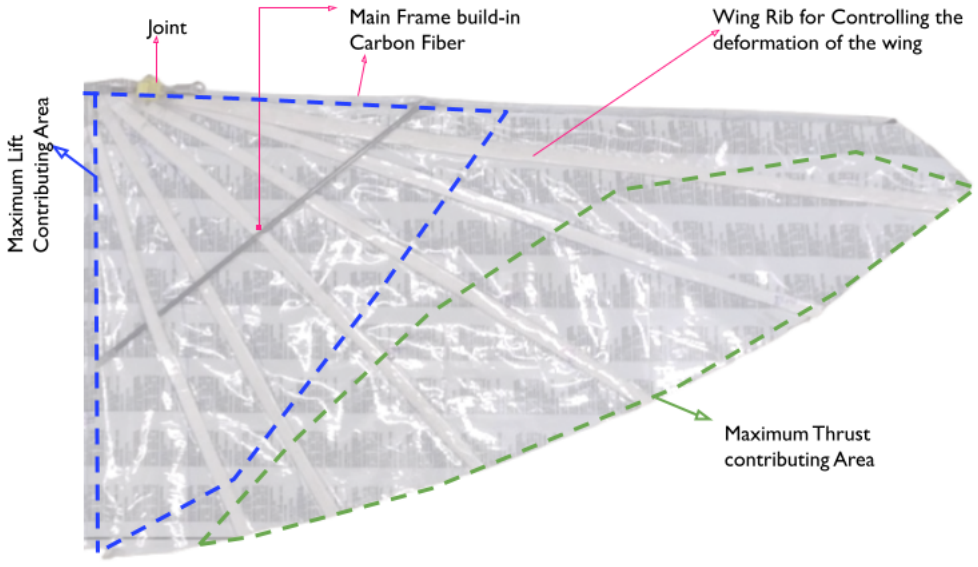


Figure 2.4: Chidiya's Wing Articular

2.5 System Integration of a Full-Scale Model

Upon completing the manufacturing process for all components, we proceeded to assemble them into a cohesive unit. The flapping wing movement is integrated with the carbon fiber chassis, and the wings are set in motion through an attached linkage system. This linkage is connected to 3D joints, directly linked to high-torque gears. These gears are part of a built-in gear train, ensuring smooth movement with the assistance of bearings to minimize friction. The rotary motion is powered by a pinion gear driven by a Brushless DC (BLDC) motor. Control over the BLDC motor is facilitated by an Electronic Speed Controller (ESC), drawing power from a Li-Po (Lithium Polymer) battery.

For comprehensive data logging of control input and output parameters, we have incorporated the Pixhawk PX4 Flight Controller. Additionally, two external servo motors are employed to manage the 90-degree V-Tail configuration. To establish communication with the ground station, a Fr-Sky X8r 8-channel multi-protocol receiver is utilized, allowing remote control and monitoring of the aerial vehicle.

Table 2.1: Ornithopter Building Components

Component	**Description**
Fuselage Material	High-density styrofoam
Wing Design	Semi-elliptical shape with laminated low-density polythene
Gear Material	PLA, ABS, Nylon (0.8 module)
Gearbox	Three-stage gear reduction with 3D printed gears
Motor	Brushless DC (BLDC) motor
Power Source	Li-Po (Lithium Polymer) Battery
Flight Controller	Pixhawk PX4
Receiver	Fr-Sky X8r 8-channel multi-protocol receiver

Table 2.2: Chidiya's (ornithopter) Specification

Component	**Description**
Total mass	450 g
Wingspan	150 cm
Flap frequency	3-6 Hz
Speed	5-8 m/s
Endurance	40 min
Operating Range	Ideally 25 km
Power Source	Li-Po battery, 2S
Control	Manual

FIGURE 2.5: Chidiya with human scale Model

FIGURE 2.6: Flight testing at 80 feet height

Chapter 3

Development of Bio-Inspired Perforated Flapping Wings

3.1 Introduction

Humans have long been fascinated by the incredible efficiency and manoeuvrability of bird and insect flight. Fixed-wing aircraft and quadcopter drones have failed to successfully imitate natural fliers in terms of efficiency and control. Therefore, biomimicry was explored, leading to the development of ornithopters [Bhowmik et al., 2013]. Due to the utilisation of airfoil technology, ornithopters are made to be exceedingly light and can produce lift with a lot less power than quadcopter propellers [Sunada and Tsuji, 2013] but their periodic flapping motion makes the problem inherently unstable [Jiao et al., 2021], thereby increasing complexity. However, this complexity is necessary to achieve exceptional manoeuvrability [Liu et al., 2016]. Therefore, with careful research and optimisation, ornithopters have the potential to outperform current UAVs, which is the motivation behind this study. The physics behind the flapping wings of birds and insects and their mechanisms to achieve net positive lift are crucial in understanding the design of ornithopters. During a single flapping motion, an upstroke and a downstroke occur. The downstroke generates positive lift and thrust since resistance due to air against the downward motion leads to an upward force. In a similar manner, the upstroke produces negative lift as air resistance against the upward motion generates a downward force [Ellington, 1984]. Also, during the upstroke, birds try to mitigate the drag as much as possible. To achieve an overall positive lift, birds have evolved two main mechanisms to reduce the negative lift generated during the upstroke.

(I)Articulated wing: Several birds bend their wings during the upstroke to lower the effective area of the wing perpendicular to the upward motion, thereby reducing downward force and minimising form drag. The feathering action of the wing increases the angle of attack, which decreases the negative lift [Banerjee et al., 2011]. They fully extend their wings during the downstroke, increasing positive lift [Jiao et al., 2021]. Such a wing design is called an articulated wing and the process is called active morphing (Fig 3.1 and Fig 3.2).

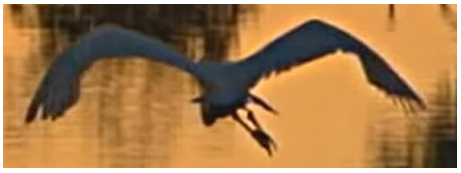


Figure 3.1: Articulated Wing During upstroke

Figure 3.2: Extended Wing During downstroke

(II) Twisting of feathers: Some birds use their wings as a check valve by twisting their feathers. Negative lift is decreased by the feathers' twisting and apertures created during the upstroke, making the wing permeable and a greater volume of air passes through. During the downstroke the feathers straighten and congregate, thereby obstructing airflow through the wing and increasing positive lift [Dvořák, 2016] (Fig: 3.3 and Fig: 3.4).

Figure 3.3: Twisting of feathers during upstroke

Figure 3.4: Twisting of feathers during downstroke

Articulated wing designs and active morphing have already been applied to several existing ornithopters using various linkages and mechanisms such as the SmartBird [Send et al., 2012] , SlowHawk2, Pterosaur Replica [Zakaria et al., 2016] etc. Chen et al. [2018] studied the lift enhancement of a flapping rotary wing by a bore-hole design and observed an increment of up to 40%. However, the use of an elastic film cover on the lower surface of the hole would make the design and manufacturing rather complicated and provide erratic results. Further, there is no observed enhancement in thrust. Yang et al. [2017] developed a feather-covered flapping wing which led to a significant gain in the lift as compared to a traditional polyester membrane wing. However, again, there is no observed increment in thrust and mass production of this design is rather arduous due to the requirement of real feathers. The main aim of this project is to develop a less complicated and simple-to-manufacture novel design that mimics the function of feathers as check valves, with more air passing through the wing during the upstroke and less air passing through during the downstroke, thereby generating net positive lift. We also aim to augment forward thrust using our design.

In this chapter, we propose a novel approach to ornithopter wing design aimed at improving lift and thrust generation while reducing negative lift during the upstroke. To achieve this, the wing design mimics the twisting of feathers in birds during the upstroke, which allows air to pass through the wing and reduces negative lift. The proposed wing design includes multiple converging holes that increase the volume of air passing through them, resulting in an increase in the outlet air velocity during the upstroke. Additionally, aligning the holes with the direction of flight allows for the generation of positive thrust during the upstroke. The effectiveness of the proposed wing design was confirmed through experimental results obtained using a six-component force transducer.
The experimental results showed a significant improvement of 12.88% and 12.44% in lift and thrust gain coefficients respectively as generated by the proposed wing design, demonstrating its potential for the development of more efficient and effective ornithopters. This approach provides a new direction for ornithopter wing design and offers potential for applications in various fields.

3.2 Experiment Methodology

3.2.1 Novel Wing Design

The current study suggests a novel design for improving positive lift during a wing's downstroke and decreasing negative lift and increasing positive thrust during its upstroke. The wings of this design as shown in Figure 3.6 incorporate several converging holes, with a larger hole opening on the upper side of the wing. During the upstroke, a larger volume of air passes through the holes due to the greater inlet area, and the outlet air velocity increases, similar to the working principle of converging nozzles in incompressible flow. Conversely, during the downstroke, a smaller inlet area causes the wing to obstruct the flow of a substantial amount of air, bolstering positive lift and hence generating net positive lift in a complete cycle. The resulting design is anticipated to enhance the wing's lift-to-drag ratio and aerodynamic performance. We fabricated various profiles of converging holes, however in order to facilitate easier manufacturing, we decided to design converging holes with straight surfaces, in the shape of truncated cones. Depending on the required top or bottom diameter, the equation of a curve of the type y = m*x is rotated about an axis to produce the desired profile for the converging holes. To further increase the effectiveness of the design, the holes have been aligned at a certain angle to the direction of flight, rather than perpendicular to it. This configuration minimises negative lift while producing positive thrust on the upstroke. In particular, the inclined conical holes cause the air to escape at a specific angle during the upstroke when the outlet air velocity is high, increasing positive thrust. Our study focuses on the use of conical holes inclined at 45 degrees to the direction of flight. The holes were made on 130 x 80 x 5 mm PLA plates (Fig 3.5). To ensure consistency in our experiment, we maintained the total number of holes and the symmetry of holes on each plate identical. We have studied three configurations such as,

- Flow passed through a converging hole, which resembles the quasi-steady upstroke of a flapping wing
- Flow passed through a diverging hole, which resembles the quasi-steady downstroke of a flapping wing
- Flow obstructed by an identical flat plate without holes, as a benchmark problem (Fig.3.5)

For each case, we find the drag force and side force for each configuration.

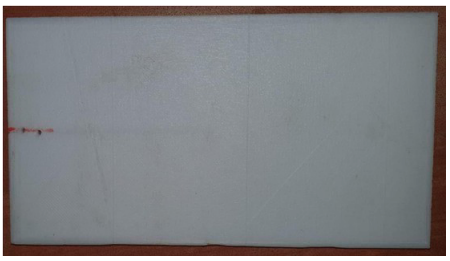


FIGURE 3.5: Flat Plate of size (130x80x5 mm)

FIGURE 3.6: Tilted Converging Hole

3.2.2 Transmission Flapping Mechanism

A flapping mechanism was developed to test the performance of perforated wings under wind tunnel conditions, simulating the flight of an ornithopter. The objective was to design and create a four-bar linkage that could provide a lightweight, high-power flapping mechanism using gears, cranks, bushings, and levers. A single crank-two rocker mechanism with four gears was eventually chosen due to its great efficiency and robustness. However, the stability of the flapping flight was found to be adversely affected by the asymmetrical movement of the flapping mechanism. When the wings flapped, this disparity created severe lateral and directional motion wobbling. In order to overcome this problem, a four-bar linkage was meticulously devised. The flapping mechanism, with an average phase lag between two wings of 2.0 degrees and a flapping amplitude of 73 degrees, is presented in Fig 3.7.

3.2.3 Experimental Setup

In order to reduce the complications associated with the flapping setup, we devised an alternate and rather simpler method. Instead of using the flapping mechanism, we placed the perforated wings directly in the wind tunnel, clamping them perpendicular to the direction of air velocity.

When a flat surface flaps symmetrically about one of its edges, the relative velocity of the surrounding fluid with respect to the surface at any instant of time is perpendicular to the surface. The only difference is that while flapping, this relative velocity is not constant throughout the wing. It increases linearly across the length of the wing, according to the relation $v = r\omega$, where 'r' is the distance from the axis of rotation and ω is the angular velocity, as shown in Fig 3.7.

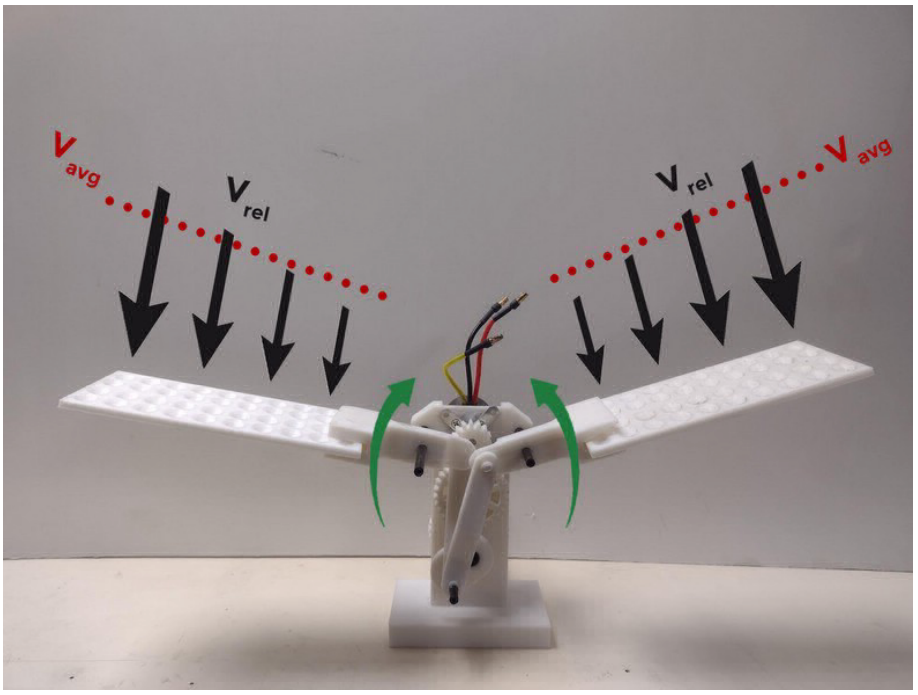


FIGURE 3.7: Twisting of feathers during downstroke

However, the purpose of this study is to validate the claim of increased thrust and lift generated by the proposed wing design. Hence, a simpler experiment is designed, where a plate with the above-mentioned nozzle-shaped holes is placed in a wind tunnel with either converging or diverging holes facing the wind. The forces along and perpendicular to the plate are measured. The converging side facing the wind emulates the condition of upstroke. If this configuration shows less force along the flow, in comparison to the diverging side facing the wind configuration, it will imply less negative lift in a flapping configuration. The same design would certainly work under the flapping conditions of ornithopters too. To obtain quantitative readings of the forces generated by the perforated wing, the model is secured to an ATI Gamma load cell with the necessary attachments. The load cell has calibrated ranges (±) and measurement uncertainty (95% confidence level), of 65N and 0.75% in Fx, 65N and 1.25% in Fy, 200N and 0.75% in Fz, 5N-m and 1.00% in Tx, 5N-m and 1.00% in Ty, 5N-m and 1.50% in Tz, respectively. The digital manometer was precisely calibrated before the experiment, and the pitot tube was perfectly aligned. With three different wing configurations and the mount separately, experiments were performed. The experimental setup (Fig 3.9,Fig 3.8, Fig 3.11,Fig 3.12,and Fig 3.10) and the parameters are unaltered. According to our interests, we have measured the force in the flow direction (drag) and the force perpendicular to the flow (side force) with different configurations of the plate. This force data was acquired by the installed load cell

and subsequently transformed into the plot depicted in section Experimental Validation 3.3.3.

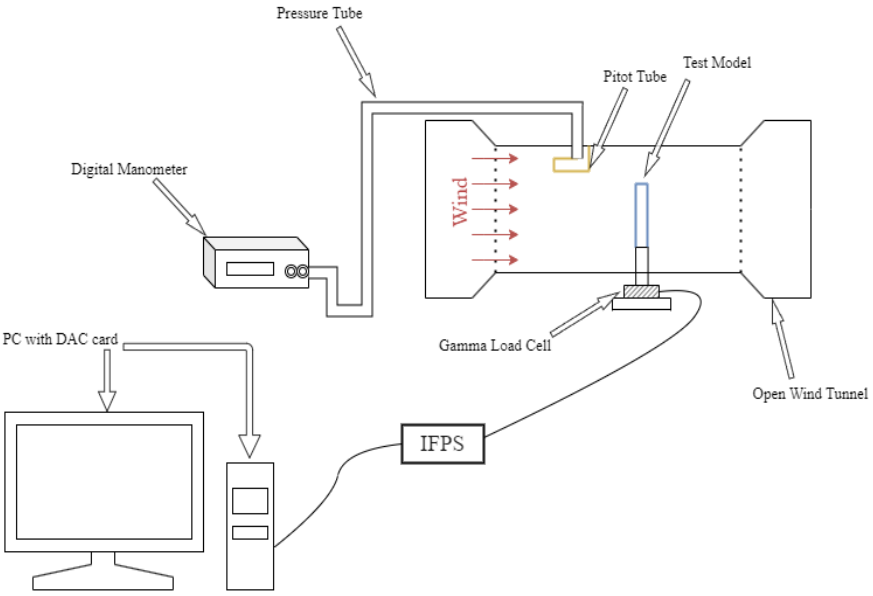


FIGURE 3.8: Schematic of the Wind Tunnel experimental setup

FIGURE 3.9: Experimental Setup with Wind Tunnel comprising of a 30x40 cm square test section

FIGURE 3.10: Setup inside the Wind Tunnel without the wing

3.3 Results

3.3.1 Theoretical Validation

The present study utilises Reynolds Transportation Theorem to determine the force experienced by a plate through a two-dimensional control volume analysis. The analysis is conducted for two scenarios, namely diverging and converging hole configurations, which are analogous to the thrust and lift forces generated during the downstroke and upstroke

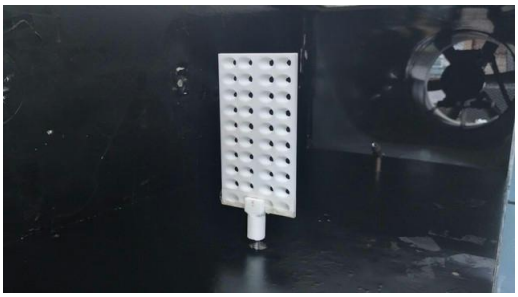


FIGURE 3.11: Perforated wing fastened inside the wind tunnel, perpendicular to the direction of airflow through the converging section

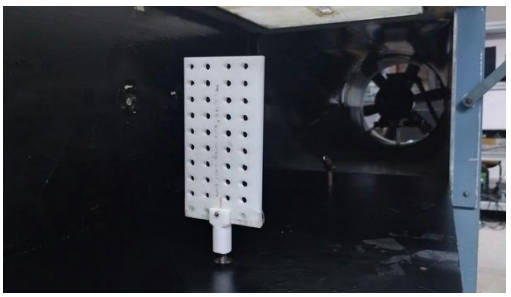
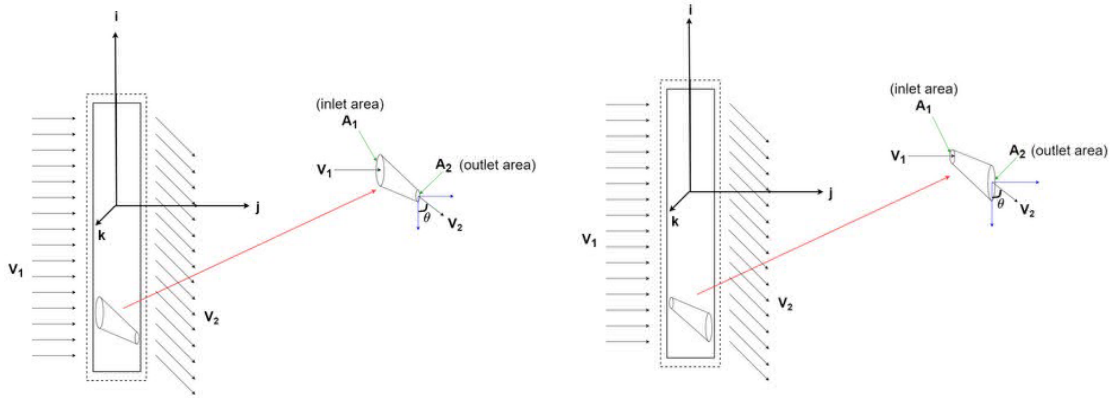


FIGURE 3.12: Perforated wing fastened inside the wind tunnel, perpendicular to the direction of air flow through the diverging section

of a flapping wing respectively. To provide a better understanding of the theoretical calculations, a schematic of the airflow through the wing is depicted in Fig 3.13



(a) Left (b) Right

FIGURE 3.13: schematic of the top view of the wing, in the case of flow through converging section (left) and diverging section (right), respectively. A closer view at the perforation with directions of the inlet and outlet velocity is shown for better understanding (θ= 45°)

With a given parameter using momentum conservation equation for 2D [3.1 and 3.2] potential flow, we get the theoretical value for different cases ([DHARAMBIRPODDAR]) as shown in table 3.1.

$$F_y = -\rho V_1^2 A_{\text{total}} + N\rho V_2^2 A_2(\sin^2\theta) \tag{3.1}$$

$$F_x = -N\rho V_2^2 A_2 \sin\theta\cos\theta \tag{3.2}$$

3.3.2 Simulation Validation

A steady-state 3D pressure-based simulation was conducted using ANSYS Fluent (Fig 3.14) with an unstructured mesh of 0.005 millimetres to verify experimental results. The simulation used the SIMPLE algorithm for pressure-velocity coupling and included a velocity inlet, outflow boundary, and stationary no-slip wall condition. A free stream velocity of 4 m/s was used. The drag force and side force have been calculated with different configurations as shown in Table 3.1.

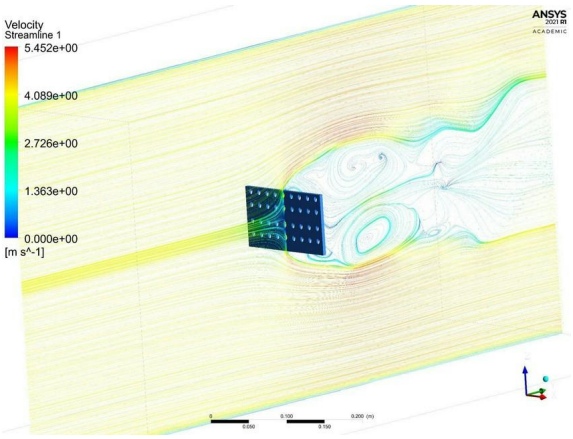


FIGURE 3.14: Isometric view of a flow

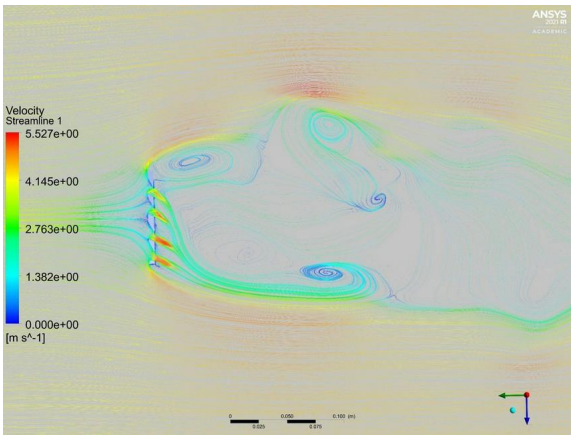


FIGURE 3.15: Flow along converging section

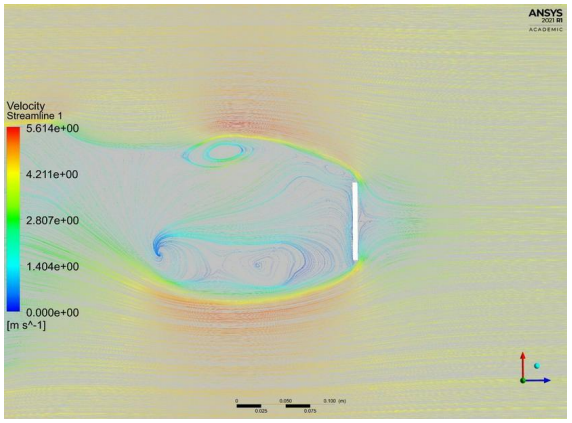


FIGURE 3.16: FLow on a flat plate without holes

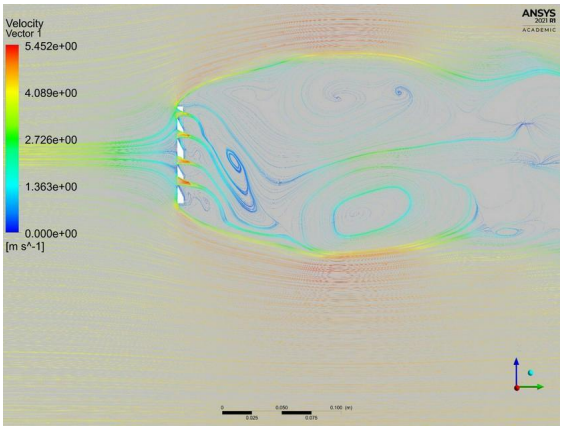


FIGURE 3.17: Flow along diverging section

3.3.3 Experimental Validation

In the experimental setup, various velocities (2m/s, 4m/s, and 6m/s) and configurations were used to investigate the drag force and side forces. The drag force is measured in

the flow direction, while the side force is measured perpendicular to the flow direction. Two configurations were compared: (i) flow from a larger inlet area to a smaller hole area, which represents the upstroke of a flapping wing, and (ii) flow from a smaller inlet area to a larger hole area, which represents the downstroke of a flapping wing. To facilitate comparison with other results, only the data obtained at a velocity of 4m/s has been shown in Fig 3.20, Fig 3.21 and its corresponding mean value is represented in Table 3.1. Also to qualitatively investigate the flow pattern behind the specimen, a smoke flow visualisation technique is used as shown in Fig 3.18 and Fig 3.19.

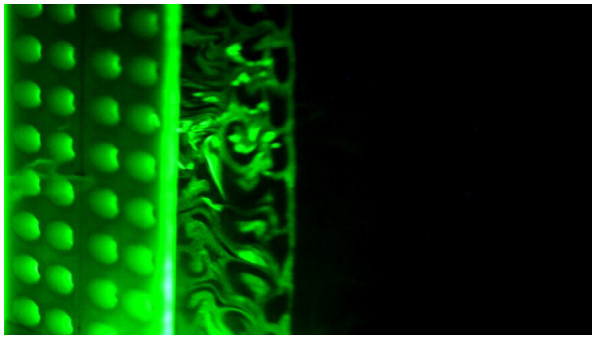


FIGURE 3.18: Isometric view from upstream side

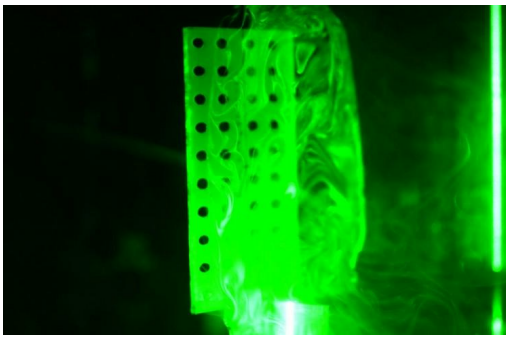


FIGURE 3.19: Isometric view from downstream side

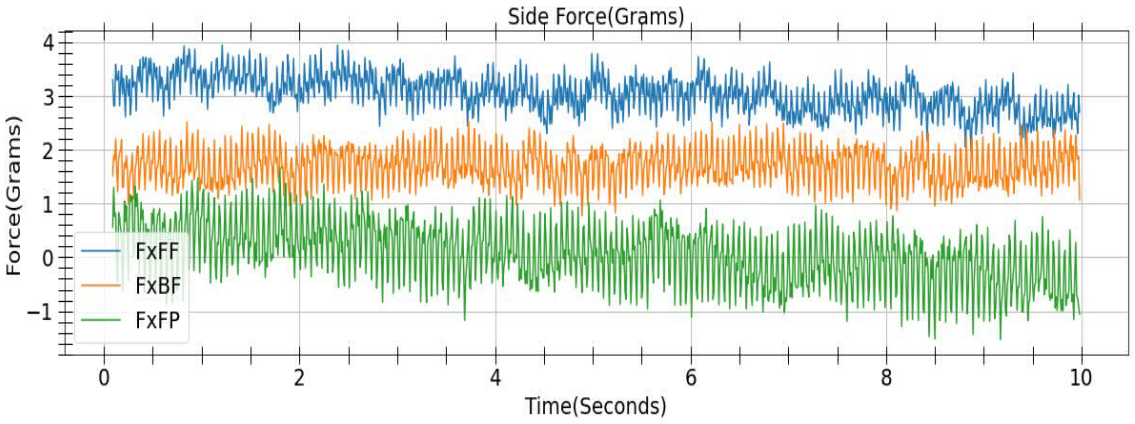


FIGURE 3.20: Acquired side force in time series data from wind tunnel experiment

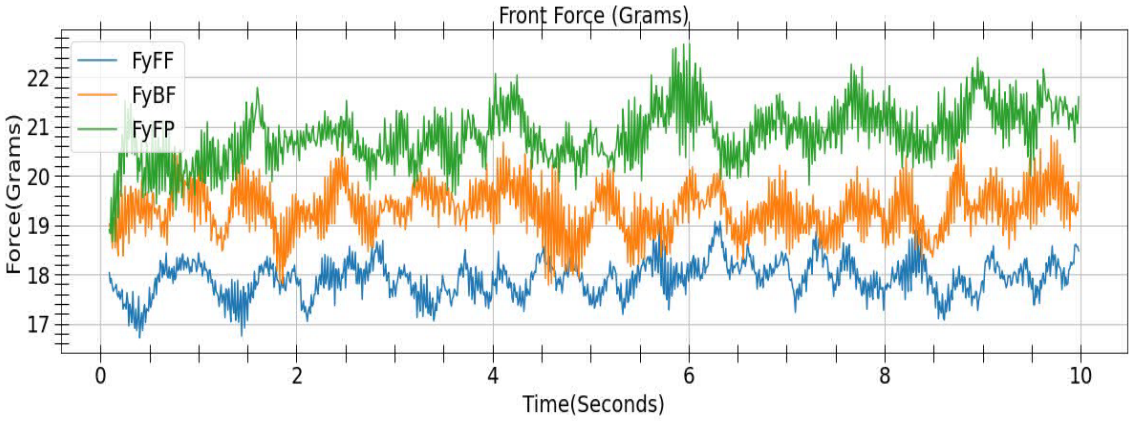


FIGURE 3.21: Acquired drag force in time series data from wind tunnel experiment

3.3.4 Discussion and Conclusion

The results (as shown in Table 3.1) indicate that there is a positive difference between the drag forces of FyBF and FyFF, which corresponds to net positive lift over a complete flapping cycle at a quasi-static condition in a quasi-steady flow. Similarly, the side forces FxBF and FxFF are positive and in the same direction, which corresponds to net thrust force in flapping motion at a quasi-static condition in a quasi-steady flow, and hence, improvement in net thrust force over a cycle of flapping wing motion. In order to measure the advantages of each case we define non-dimensional gain coefficients i.e. lift gain coefficient and thrust gain coefficient as -

$$\text{Clg} = \frac{\text{Net lift gain in a cycle with respect to benchmark case}}{0.5 \cdot A \cdot \rho \cdot v^2}$$

$$\text{Clg} = \frac{|(F_y\text{FP} - F_y\text{FF}) - (F_y\text{FP} - F_y\text{BF})|}{0.5 \cdot \rho \cdot A \cdot v^2} = \frac{|F_y\text{BF} - F_y\text{FF}|}{0.5 \cdot \rho \cdot A \cdot v^2} \tag{3.3}$$

$$\text{Ctg} = \frac{\text{Net thrust gain in a cycle with respect to benchmark case}}{0.5 \cdot A \cdot \rho \cdot v^2}$$

$$\text{Ctg} = \frac{|(F_x\text{FP} - F_x\text{FF}) - (F_x\text{FP} - F_x\text{BF})|}{0.5 \cdot \rho \cdot A \cdot v^2} = \frac{|F_x\text{FP} - F_x\text{BF}|}{0.5 \cdot \rho \cdot A \cdot v^2} \tag{3.4}$$

TABLE 3.1: Forces measured in theoretical, experimental, and simulation methods respectively

Results	FxFF	FxBF	FxFP	FyFF	FyBF	FyFP
Theoretical	3.9202	0.07115	0	16.7948	20.6438	20.715
Experimental	3.044	1.6953	0.0558	17.9193	19.3158	20.8282
Simulation	1.06	0.507	0.088	14.4638	15.284	16.01

TABLE 3.2: Lift and thrust gain coefficients calculated

Results	Theoretical Gain	Simulation Gain	Experimental Gain
Clg	35.5 %	7.56 %	12.88 %
Ctg	35.5 %	5.1 %	12.44 %

The outcomes of theoretical calculations, numerical simulations, and experimental results often exhibit discrepancies due to certain simplifications and assumptions made in each approach. In theoretical calculations, the absence of flow parallel to the face of the plate, the assumption of 2D potential flow, and irrotational flow are common causes of variation from experimental results. Similarly, in numerical simulations, the use of unstructured coarse grids can lead to deviations from actual results while reducing computational complexity. Hence, it is important to consider these factors while interpreting the results obtained from theoretical and numerical approaches. The repeatability of the experiment was assessed and consistent results were obtained for three iterations. However, the present study is just an initial test of our hypothesis of using perforated wings in ornithopters. Hence, we chose a simple rectangular plate with holes and performed the experiment by fixing the plate in a direction perpendicular to the airflow, then flipping it by 180° about the vertical axis and repeating the procedure. Therefore, we have tested the plate for particular configurations during the upstroke and downstroke separately.

In future scope of research, we aim to test different wing structures and aerofoils with a similar perforated design for different wind velocities, by flapping the wing in a wind tunnel. We also plan to incorporate turbulent models, structured grids and adaptive mesh in simulations to obtain more accurate results, which could potentially match the experimental observations. A comprehensive study on the same will help us optimise the wing design to attain maximum increment in lift and thrust for practical applications.

Chapter 4

Development of Bio-Inspired Mechanism to Generate Various Wingtip Motions

4.1 Motivation

Have you ever watched a bird or insect in flight? They can change how they fly in an instant – chasing prey with bursts of speed or glide effortlessly to save energy on long journeys. This amazing ability comes from the way they can actively articulate the wings and also change their shape. In this chapter, we will explore how to build flying robot's wings that can articulate their wings, that results wing tip motions for different flying conditions; e.g., in hovering, wingtip follows path of figure eight, while in forward flight the tip path will be elliptic in nature. Our goal is to learn from nature, copying the incredible ways birds and insects fly.

Here our motivation comes from the abilities of natural flyers like dragonflies. We are inspired to understand and replicate their efficient wing movements to improve robotic flight. By mimicking how these creatures adapt their wing motions, we can build robots or ornithopters that are both agile and efficient in the air.

4.2 Introduction

Dragonflies, renowned for their intelligence and remarkable flight capabilities, serve as a compelling source of inspiration for the development of smart and efficient aerial vehicles. In this research, we embark on the creation of a dragonfly-inspired air vehicle, driven by the exceptional aerodynamic efficiency exhibited by these insects.

Central to our endeavor is the meticulous study of dragonfly wing kinematics, a crucial determinant of their flight skills. Extracting wing kinematics from natural videos [Geographic, 2013],[National Geographic, 2015], [TED, 2016] and [National Geographic, 2017] depicting dragonfly flight, we analyzed specific trajectories through an exhaustive literature review and direct observation. These trajectories were graphically represented, revealing distinct shapes corresponding to the motion patterns of the dragonfly's wing tips as shown in figure 4.1, figure 4.2.

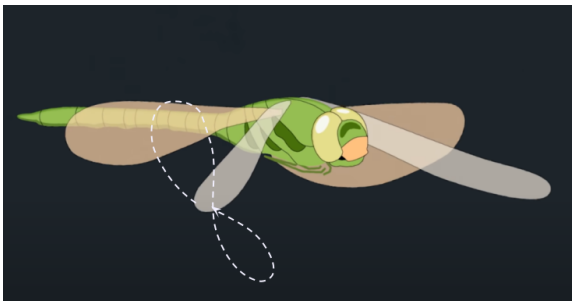


Figure 4.1: Schematic; Wing tip trajectory of forward flight

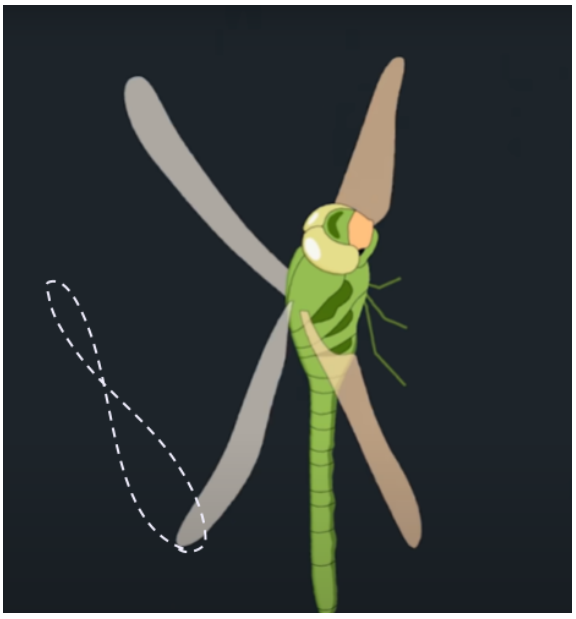


FIGURE 4.2: Schematic; Wing tip trajectory of backward flight

Researchers found that the figure-8 wingtip motion, used by insects like dragonflies and hummingbirds, is crucial for generating maximum lift and achieving hovering capabilities [LEYS ET AL., 2016]. This motion creates two loops, whose relative sizes change with flight type. In backward flight, the loops are equally sized [BODE-OKE ET AL., 2018], while forward flight displays a large difference, with one loop significantly smaller [WANG ET AL., 2003]. Inspired by this, we've built a proof-of-concept model capable of replicating these distinct loop patterns. By adjusting the phase difference parameter (eq. 4.7), we can easily switch between the desired wing patterns.

This chapter encompasses the construction of a mathematical model of kinematics to generate various flight loops, followed by the manufacturing of a dragonfly wing and kinematic experiment setup. We compare the locus of the wingtip obtained from high speed camera visualization with the mathematical model.

4.3 Development of Mechanism Model for Mimicking Various the Wing Tip Motions

To convert rotary motion to flapping motion, four-bar and five-bar linkage mechanisms are commonly used. To achieve more complex mechanisms like those found in hummingbirds

in hovering motions , where spin axis motion is also included, different types of gear trains and gear mechanisms are employed YANG ET AL. [2018a].

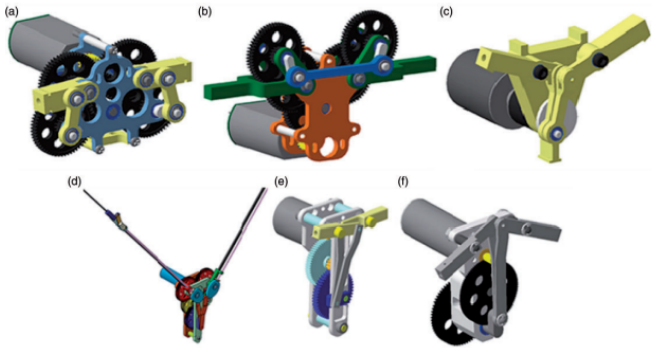


FIGURE 4.3: Various types of mechanism to convert rotary to flapping motion

The limitation of these mechanisms is that they cannot be easily altered from one wingtip motion to another. Therefore, we are attempting to build a novel mechanism that not only generates wingtip motion but also allows for alterations with minimal changes to the mechanism.
We used Linkage® software to generate five-bar linkage mechanisms for trial and error. Then, we refined these designs by modeling them mathematically.

4.3.1 Mathematical Modeling

The kinematics modeling for the mechanism is model mathematically as discussed in algorithm 1. The mathematical equation is developed based on the figure 4.4

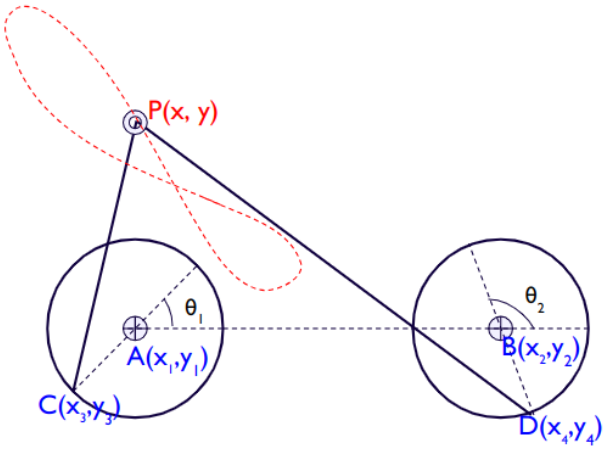


FIGURE 4.4: Schematic diagram of Kinematics Modeling

$$x_3 = x_1 + R \cdot \cos(\theta) \quad \Longrightarrow \quad x_3 = x_1 + R \cdot \cos(\omega t) \tag{4.1}$$

$$y_3 = x_1 + R \cdot \sin(\theta) \quad \Longrightarrow \quad y_3 = x_1 + R \cdot \sin(\omega t) \tag{4.2}$$

$$x_4 = x_2 + R \cdot \cos(\theta + \phi) \quad \Longrightarrow \quad x_4 = x_2 + R \cdot \cos(\omega t + \phi) \tag{4.3}$$

$$y_4 = x_2 + R \cdot \sin(\theta + \phi) \quad \Longrightarrow \quad y_4 = x_2 + R \cdot \sin(\omega t + \phi) \tag{4.4}$$

$$(x - x_3)^2 + (y - y_3)^2 = l_1^2 \tag{4.5}$$

$$(x - x_4)^2 + (y - y_4)^2 = l_2^2 \tag{4.6}$$

Using the equations mentioned above, the locus $P(x,y)$ of the intersection between two circles x_1 and x_2 is calculated for each time t with different ϕ, where ϕ is the phase angle between the two wheels. The ϕ can be varied based on uses by manually or autonomously.

$l_1 = \overline{CP}$ and $l_2 = \overline{PD}$ represent the connecting links of the mechanism, and $R = \overline{AC}$ is the radius of the wheel. We solved the equation using MATLAB® and extracted the locus theoretically with varying phase angles ϕ.

Algorithm 1 Locus of P(x,y) of Linkage Calculation

procedure CALCULATE TRAJECTORY(ϕ, ω, L, R_1, R_2, L_1, L_2, t_{start}, t_{end}, t_{step}) ▷ Parameters: linkage geometry, angular velocity, time range, step size
  Define symbolic variables x, y, t
  Define crank trajectories:
  $X1(t) = R_1 \cos(\omega t - \pi)$
  $Y1(t) = R_1 \sin(\omega t - \pi)$
  $X2(t) = R_2 \cos(\omega t + \phi - \pi) + L$
  $Y2(t) = R_2 \sin(\omega t + \phi - \pi)$
  Define floating link constraints:
  $(x - X1(t))^2 + (y - Y1(t))^2 = L_1^2$
  $(x - X2(t))^2 + (y - Y2(t))^2 = L_2^2$
  $t\_vals \leftarrow t_{start} : t_{step} : t_{end}$ ▷ Generate time values
  Initialize empty lists $x\_intersect$, $y\_intersect$
  for t in t_vals **do**
    sol = SolveSystem($(x - X1(t))^2 + (y - Y1(t))^2 = L_1^2$,
      (x - X2(t))^2 + $(y - Y2(t))^2 = L_2^2$)
    for i in 1 to 2 **do** ▷ Handle two possible solutions
      if $sol.x(i)$ is real **then**
        $x\_intersect.append$(double(subs(sol.x(i), t)))
        $y\_intersect.append$(double(subs(sol.y(i), t)))
      end if
    end for
  end for
  return $x\_intersect$, $y\_intersect$
end procedure

4.3.2 Kinematics of Mechanism

After formulating the mathematical model, than constructing a physical mechanism for experimental validation. Employing Linkage® software, we simulated the mechanism and derived the following dimensions: $\overline{AC} = 30\,\text{mm}$, $\overline{CP} = 106\,\text{mm}$, $\overline{PB} = 185\,\text{mm}$, $\overline{BD} = 30\,\text{mm}$, and $\overline{AD} = 140\,\text{mm}$. The angles θ_1 and θ_2, formed by the lines $\overline{AC}$ and $\overline{BD}$ respectively with respect to the horizontal axis, were crucial parameters. In accordance with our established definition.

$$\theta_1 - \theta_2 = \phi \tag{4.7}$$

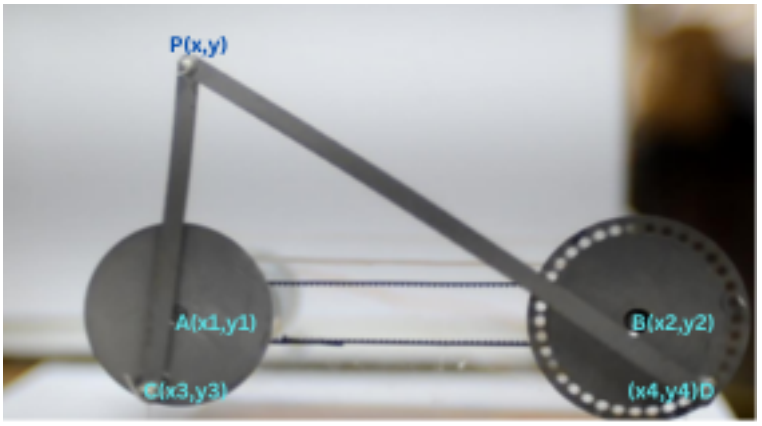


FIGURE 4.5: Kinematics Mechanism

4.4 Fabrication of Model

The design and fabrication of a bio-inspired dragonfly wing mechanism was developed at the UAL, IIT Kanpur laboratory. The mechanism and accompanying wing model aims to replicate the complex flight dynamics of a dragonfly.

The mechanism's foundation is a five-bar linkage system with two degrees of freedom. To streamline operation, one degree of freedom is constrained using a belt system. Stainless steel was chosen for the linkages to ensure robust structural integrity.

The mechanism is powered by a 12V, 1000 RPM DC motor. This motor drives a primary wheel connected to a driven wheel via a toothed belt. The driven wheel incorporates precisely spaced holes at 10°intervals, facilitating adjustments to the mechanism's movement patterns.

The wing model fig.4.7 meticulously replicates the structure of a species of dragonfly, Cordulegastridae wing (fig.4.6). We carefully traced the major wing articulations from a detailed image and reproduced them using 3D printing technology. This approach yields a flexible wing structure that closely mimics the movement of a real dragonfly wing. Mylar sheet was attached to the 3D printed structure to complete the realistic wing, achieving a final wingspan of 160 mm. After that the all the components is assembled (fig.4.8) together for experiment.

To ensure meticulous precision and dimensional accuracy, we employed a laser cutter for the manufacturing of all mechanism and wing components. This methodology guarantees the highest level of detail and consistency throughout the fabrication process.

FIGURE 4.6: Real Dragon Fly Wing

FIGURE 4.7: Fabricated Artificial Wing

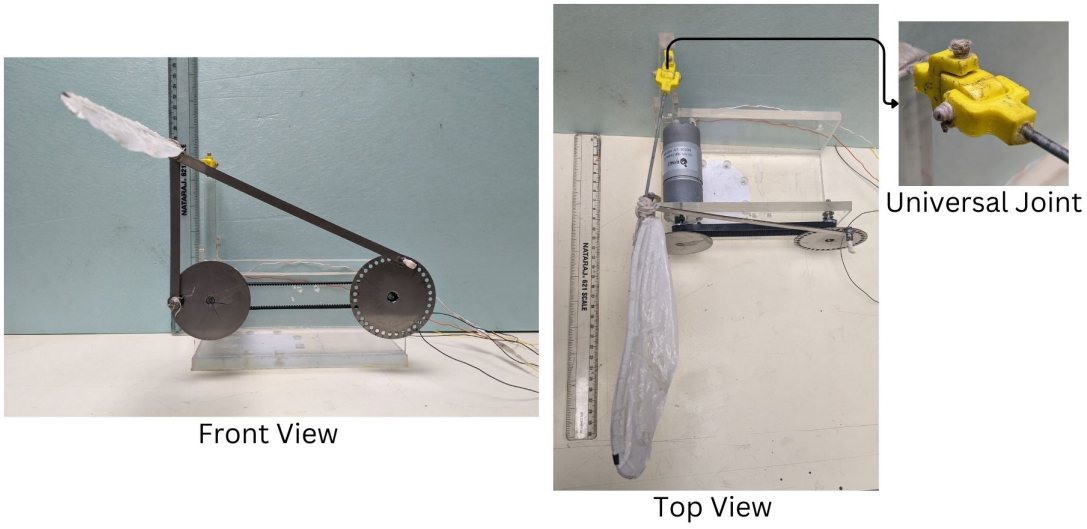


FIGURE 4.8: Assembled Experiment Setup

4.5 Comparison of Wing Trajectory Obtained from Mathematical Models with Experimental Results

The MATLAB code included entering the equations of X3 and X4 as per the values obtained from our linkage mechanism. The only thing varied in the following plots is the phase difference, that is

$$\phi = \theta_1 - \theta_2 \tag{4.8}$$

We changed the phase angle from 0 degree (0 radian) to 60 degree (1.05 radian) and observed the trajectories.

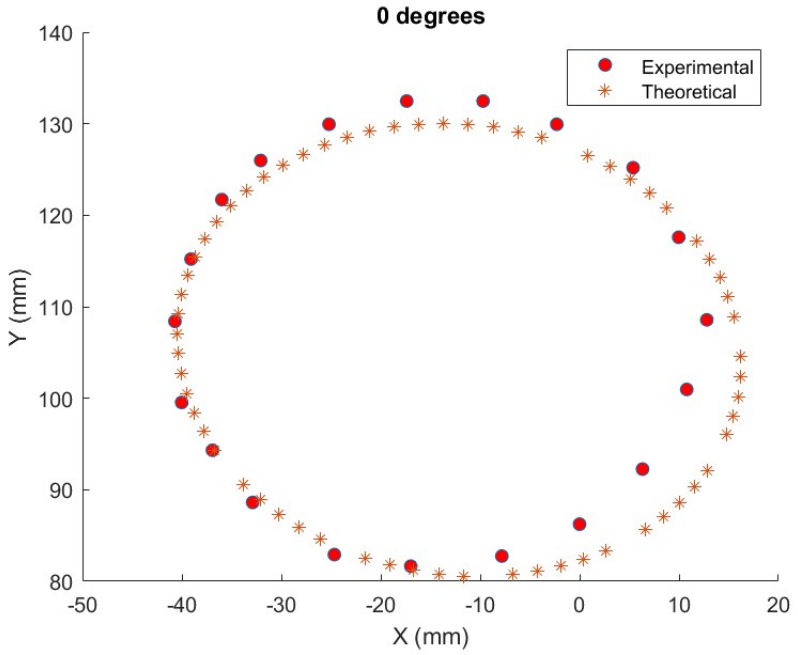


FIGURE 4.9: Comparison between experimental and theoretical at ϕ=0°

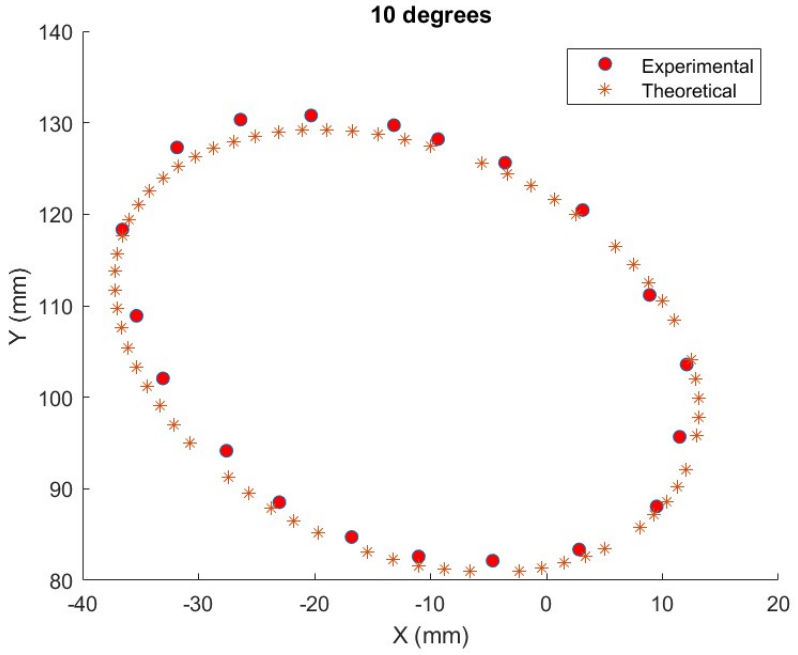


FIGURE 4.10: Comparison between experimental and theoretical at ϕ=10°

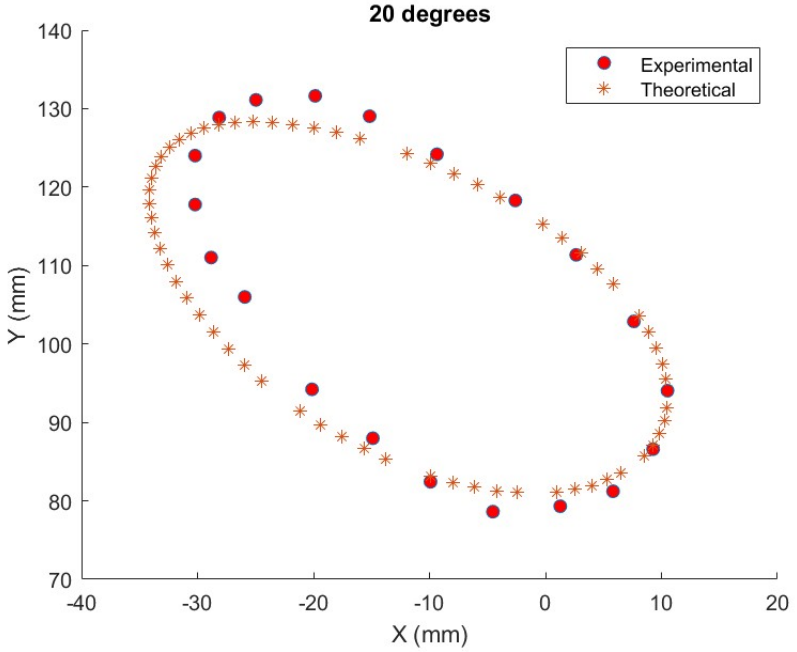


FIGURE 4.11: Comparison between experimental and theoretical at ϕ=20°

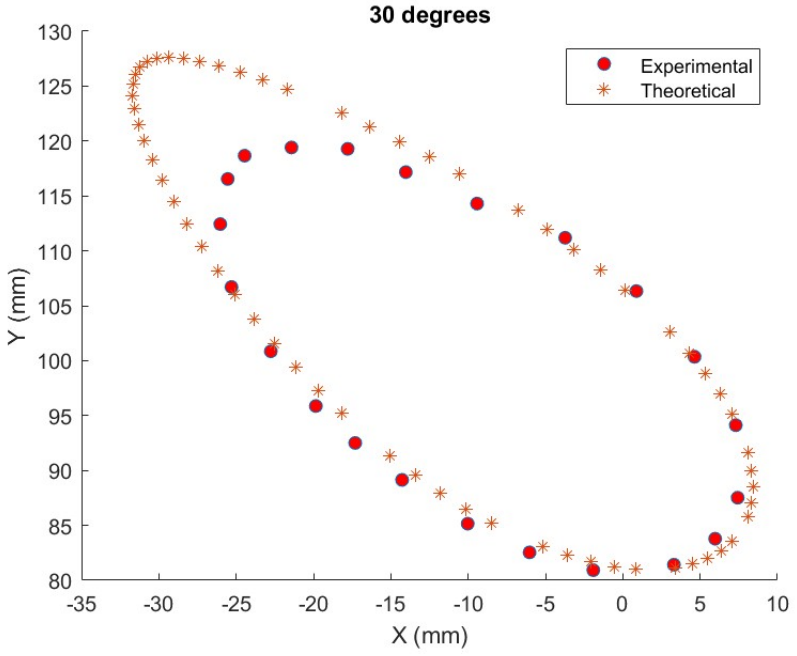


FIGURE 4.12: Comparison between experimental and theoretical at ϕ=30°

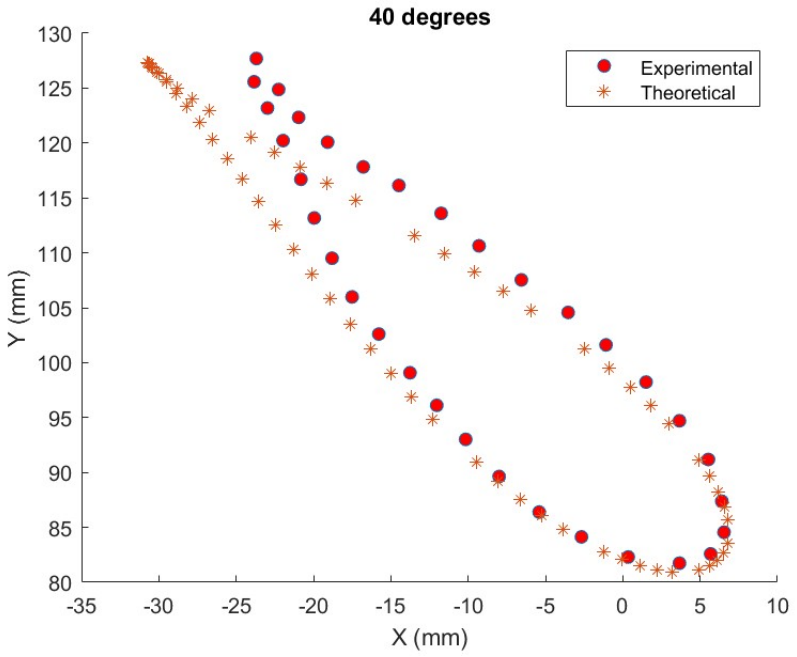


FIGURE 4.13: Comparison between experimental and theoretical at ϕ=40°

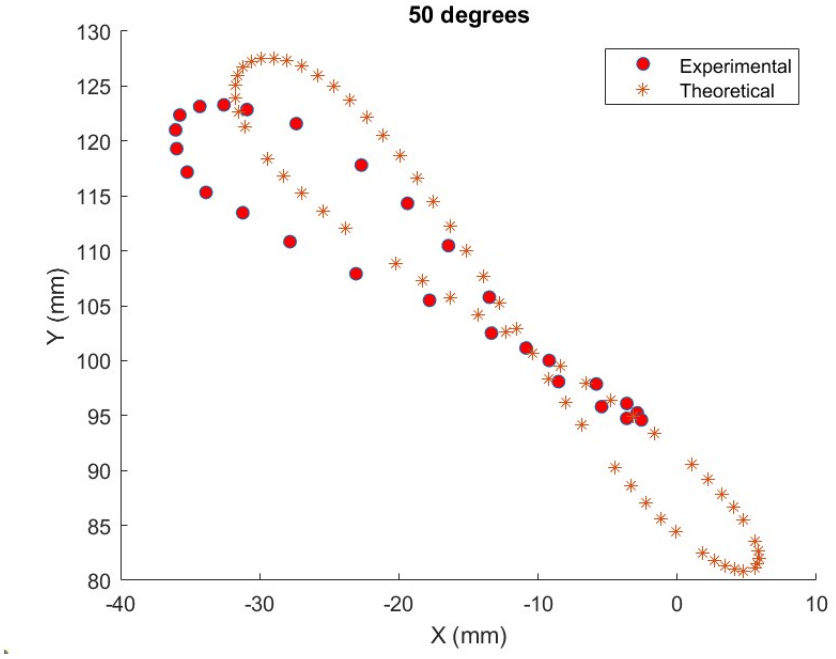


FIGURE 4.14: Comparison between experimental and theoretical at ϕ=50°

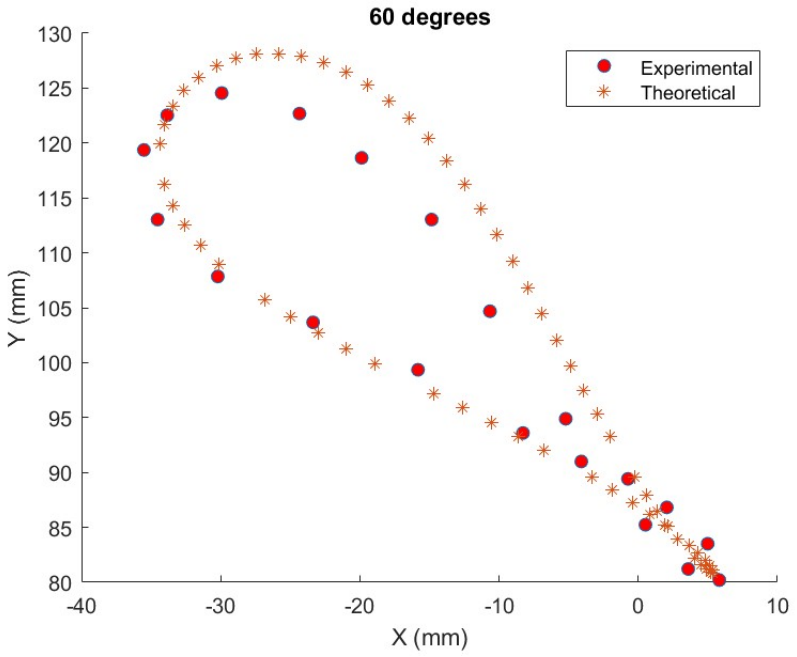


FIGURE 4.15: Comparison between experimental and theoretical at ϕ=60°

The comparison shows that the experimental and theoretical results closely align within the ϕ range of 0°to 30°. However, at higher angles, deviations occur between the experimental and theoretical results due to the three-dimensional nature of the mechanism.

4.6 Conclusion

In conclusion, our study demonstrates that a mechanism can be generate various wingtip motion using the five-bar linkage mechanism. We confirmed the comparison of trajectory between mathematical modeling and a fabricated mechanism. Notably, our findings reveal that by adjusting the phase difference (ϕ), the wing tip trajectory motion can be shifted, offering versatility crucial for enhancing agility or power efficiency depending on the purpose. Importantly, this versatility can be achieved with a single mechanism, adding to its practicality and efficiency.

Our goal is to use the mechanism and shift the phase angle ϕ autonomously in real-time to enhance the ornithopter model, making it a more robust flying vehicle. Moving forward, further validation of the mechanism through force measurement with wind tunnel experiments is essential. This vision is to advancing the field of aerial robotics through innovative mechanisms and experimental development.

Chapter 5

Data Cleaning and Refinement for Data-Driven Model

5.1 Introduction

In recent years, researchers have made impressive strides in the study of ornithopters, or mechanical birds, successfully piloting manually controlled versions of these fascinating flying machines. However, despite these accomplishments, there are still crucial areas within ornithopter research that require further exploration and refinement.

One major challenge lies in developing a mechanism that accurately mimics the complex aerodynamics of natural bird flight, especially considering the unique flapping motion of ornithopters. Understanding and characterizing this motion is essential for progress in the field. Additionally, the diverse sizes and shapes of flapping wings present a challenge for modeling, especially when it comes to creating control algorithms for autonomous flight.

While some researchers have made attempts at achieving autonomous flight in ornithopters, many of these efforts have been limited to small-scale or hobby-grade models. However, important research by Bhowmik et al. [2014] has highlighted the importance of identifying specific flapping frequencies that optimize ornithopter performance based on their weight and wingspan.

In response to these challenges, our research takes a novel approach. Instead of relying solely on physics-based modeling, we propose integrating real-time data collection during flight. By using an onboard flight controller, we aim to capture the intricate dynamics

of ornithopter flight by mapping input data against output data. We plan to utilize Deep Neural Networks (DNNs) and Recurrent Neural Networks (RNNs) to construct a dynamic model that not only captures the complexities of ornithopter flight but also enables autonomous control.

5.2 Collection of Data

In the process of extracting real-time data from our ornithopter during manual flight of around 15 minutes, we utilized the Pixhawk® flight controller, which is equipped with an STM 32-bit microcontroller unit. In addition to the built-in flight controller sensors such as the IMU and gyroscope sensor, we also incorporated a GPS sensor to gather data for further post-processing and training purposes. The Pixhawk flight controller logs data in ulog format, encapsulating all the information regarding the ornithopter's performance and controller data sent from the ground station. Following each flight, this data is downloaded for the purpose of building data-driven system identification and eventually developing a controller for the ornithopter. The Mission Planner (software platform from ArduPilot®) ground controller and PX4 flight review serve as integral components in simplifying the extraction process, providing a user-friendly interface for accessing recorded flight parameters.

5.2.1 Key Parameters Recorded

The collected data encompasses crucial parameters essential for system identification and control algorithm development. These parameters include roll, pitch, yaw, altitude, latitude, longitude, and three actuator data such as c1-roll, c2-pitch, c3-throttle (PWM signals), recorded during flight experiments conducted with various sensor configurations.

5.2.2 Actuator Data Significance

Specifically, the actuators comprise two servo motors and one BLDC motor, each associated with distinct PWM signals: c1-roll, c2-pitch, and c3-throttle, respectively. Servo motors C1 and C2 cooperatively induce rolling and pitching motions of the ornithopter, whereas the BLDC motor governs the ornithopter's throttle, thereby influencing its flapping frequency.

5.3 Post-Processing the data

The raw flight data will be passed through following processes in order to make sense for neural network model; [Dharambir Poddar, 2024];

5.3.1 Resampling:

During flight data collection, sensors often produce readings at irregular intervals due to factors like asynchronous data logging or varying sampling rates. Irregular sampling can introduce complications into subsequent analysis and modeling tasks. Resampling addresses this by creating a new dataset with consistently spaced time intervals. This is achieved through interpolation techniques, and the procedure of data Up-sampling and Down-sampling is discussed in algorithm 2.

Linear Interpolation: Assumes a straight line connects adjacent data points. It's computationally efficient but can miss subtleties if the underlying signal changes rapidly.

Cubic Interpolation: Fits a smoother curve through data points, potentially capturing more intricate variations, but is more computationally intensive. As an example it shown in figure 5.2 and 5.1 performing up-sampling GPS data and down-sampling Gyroscope data respectively.

Sensors	Sampling Rate (Hz)	Resampling Rate (Hz)
GPS	1	10
Gyroscope	20	10
IMU	50	10
Transmitter	10	10

Algorithm 2 Data Resampling

function RESAMPLE($tparam, yparam$)
  $x \leftarrow df[tparam]$ ▷ Extract time data
  $y \leftarrow df[yparam]$ ▷ Extract data to resample
  $f_{linear} \leftarrow$ INTERPOLATE(x, y) ▷ Linear interpolation
  $f_{cubic} \leftarrow$ INTERPOLATE$(x, y,$ 'cubic') ▷ Cubic interpolation
  $last\_val \leftarrow df.loc[x.last\_valid\_index(), tparam]$ ▷ Final timestamp
  $x_{new} \leftarrow$ ARANGE$(0, last_val, last_val/5000)$ ▷ New time points
  $y_{linear} \leftarrow f_{linear}(x_{new})$ ▷ Resample using linear interpolation
  ▷ Optionally: $y_{cubic} \leftarrow f_{cubic}(x_{new})$
  $df_{sam}[yparam] \leftarrow y_{linear}$
  $df_{sam}[tparam] \leftarrow x_{new}$
end function

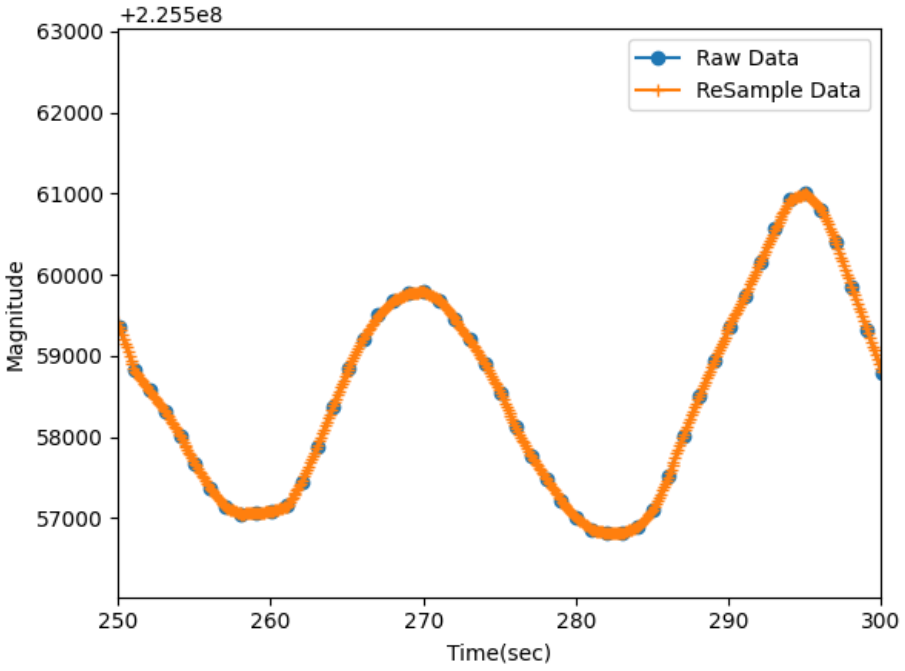


FIGURE 5.1: Up-Sampling GPS data

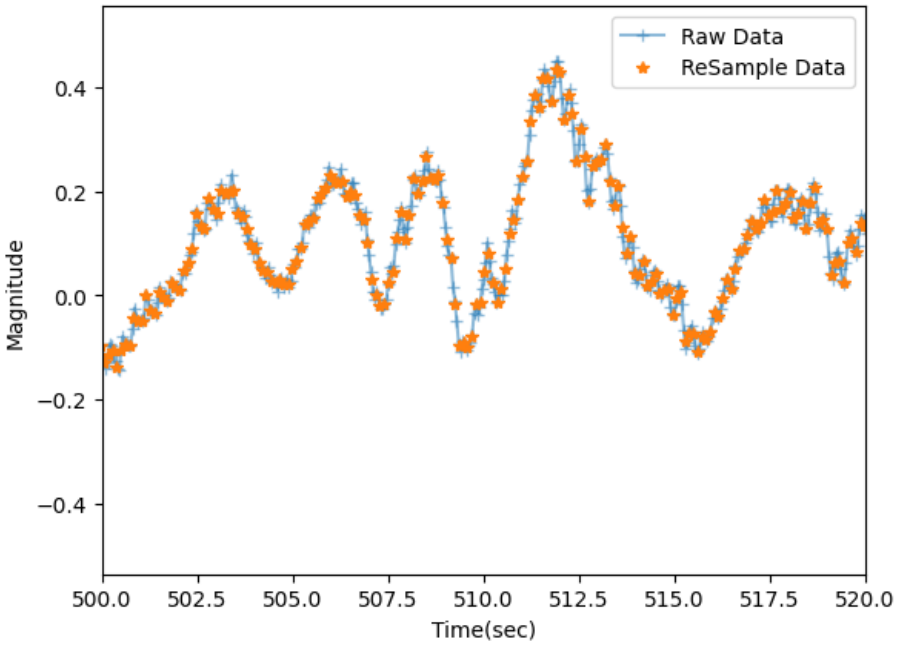


FIGURE 5.2: Down-Sampling Gyroscope data

5.3.2 Quaternion to Euler Conversion:

Orientation data is provided in quaternion format (q_0, q_1, q_2, q_3), which is compact but less intuitive to visualize. The customized `QuaternionToEuler` function algorithm 3 converts quaternions to more familiar roll, pitch, and yaw angles in degrees, making it easier to interpret the ornithopter's attitude.

Algorithm 3 Quaternion to Euler Angle Conversion

function QUATERNIONTOEULER(w, x, y, z)
  $ysqr \leftarrow y * y$
  $t0 \leftarrow +2.0 * (w * x + y * z)$
  $t1 \leftarrow +1.0 - 2.0 * (x * x + ysqr)$
  $X \leftarrow \text{degrees}(\arctan 2(t0, t1))$ ▷ Roll
  $t2 \leftarrow +2.0 * (w * y - z * x)$
  $t2 \leftarrow \max\{-1.0, \min\{t2, +1.0\}\}$ ▷ Restrict range
  $Y \leftarrow \text{degrees}(\arcsin(t2))$ ▷ Pitch
  $t3 \leftarrow +2.0 * (w * z + x * y)$
  $t4 \leftarrow +1.0 - 2.0 * (ysqr + z * z)$
  $Z \leftarrow \text{degrees}(\arctan 2(t3, t4))$ ▷ Yaw
  return $[X, Y, Z]$
end function

5.3.3 GPS spherical axis to 3D Cartesian Transformation:

For precision flight control and path planning, GPS coordinates (latitude, longitude, altitude) were transformed into a local cartesian coordinate system using the Universal Transverse Mercator (UTM) and WGS84.. The `gps2xy` function (shown below) implemented this conversion, ensuring compatibility with control algorithms and simplifying local calculations.

The `gps2xy` Function This function addresses these issues by:

1. **Coordinate Systems:** It defines two coordinate reference systems (CRS):

   - **WGS84**[Wikipedia contributors, 2024b]: The standard system used by GPS.
   - **UTM** [Wikipedia contributors, 2024a]: The Universal Transverse Mercator system, a projected coordinate system that divides the Earth into zones. UTM provides a locally flat, grid-like mapping.

2. **UTM Zone Selection:**
   The code reminds the user to select the appropriate UTM zone for their location.

Working within a single UTM zone minimizes distortions caused by projecting the Earth onto a flat map.

3. **Transformation:**
   A `pyproj`[Whitaker et al., 2022]transformer object handles the conversion between the WGS84 (GPS) and UTM coordinate systems. The always `xy=True` ensures the output is in (X, Y) format.

Output of `gps2xy` function: The function returns the X and Y coordinates within the selected UTM zone.

Importance of Local Flight:
In a UTM system, simple Euclidean geometry can be used to calculate distances and directions relevant to the vehicle's path within its local operating area. Many flight control algorithms are designed to work with Cartesian coordinates (X,Y,Z). This conversion provides compatible inputs. Also,visualizing flight paths and planning maneuvers becomes more intuitive on a local grid system.

5.3.4 Axis Offsetting and Time Averaging

X and Y coordinates are offset by subtracting their initial values, making subsequent calculations and plotting relative to the flight's starting position. A new 'time' column is generated with the average of all the column of resample time, to make uniformity.

5.3.5 Normalization of Data

To improve neural network training and prevent the dominance of any single feature, Min-Max normalization was applied. Each feature column was scaled to the range of 0 to 1 using the following algorithm 4.

$$\text{normalized\_value} = \frac{\text{original\_value} - \text{min\_value}}{\text{max\_value} - \text{min\_value}}$$

Importance in Feature Scaling for Neural Networks Training

- **Convergence:** Gradient descent-based training algorithms, common in neural networks, converge faster and more reliably when features are on similar scales.

- **Feature Importance:** Normalization prevents features with naturally large magnitudes from dominating the model, allowing the neural network to learn the relative importance of each feature.
- **Temporal Patterns:** To capture time-dependent patterns, time differences between consecutive data points were calculated. A new column, *time_diff*, was created and subsequently normalized, as follows in algorithm 4.

$$\text{time\_diff} = \text{dataFrame['time'].shift(-1)} - \text{dataFrame['time']}$$

Time differences can help the neural network learn patterns related to how quickly the system is changing, which can be important for control and prediction tasks.

Algorithm 4 Data Normalization

▷ Calculate range for attitude angles

$Roll_{max} \leftarrow \text{MAX}(df['roll'])$

$Roll_{min} \leftarrow \text{MIN}(df['roll'])$

$Pitch_{max} \leftarrow \text{MAX}(df['pitch'])$

$Pitch_{min} \leftarrow \text{MIN}(df['pitch'])$

$Yaw_{max} \leftarrow \text{MAX}(df['yaw'])$

$Yaw_{min} \leftarrow \text{MIN}(df['yaw'])$

▷ Calculate range for control signals

$C_{min} \leftarrow 900$ ▷ Adjust if needed

$C_{max} \leftarrow 2100$ ▷ Adjust if needed

▷ Normalize attitude angles

$$df['roll'] \leftarrow \frac{df['roll'] - Roll_{min}}{Roll_{max} - Roll_{min}}$$

$$df['pitch'] \leftarrow \frac{df['pitch'] - Pitch_{min}}{Pitch_{max} - Pitch_{min}}$$

$$df['yaw'] \leftarrow \frac{df['yaw'] - Yaw_{min}}{Yaw_{max} - Yaw_{min}}$$

▷ Normalize control signals

$$df['c1\_roll'] \leftarrow \frac{df['c1\_roll'] - C_{min}}{C_{max} - C_{min}}$$

$$df['c2\_pitch'] \leftarrow \frac{df['c2\_pitch'] - C_{min}}{C_{max} - C_{min}}$$

$$df['c3\_throttle'] \leftarrow \frac{df['c3\_throttle'] - C_{min}}{C_{max} - C_{min}}$$

▷ Additional normalizations as needed ...

5.3.6 Saving Data

Following post-processing[1], the data is saved in the required format for the subsequent development of a Neural Network-based system dynamics model and controller design.

TABLE 5.1: Normalized parameters used for further analysis as represented in columns

time	**roll**	**pitch**	**yaw**	**x**	**y**	**alt**	**c1Roll**	**c2Pitch**	**c3Throttle**

[1] `https://github.com/dharambirpoddar/Bird_Resampling_Refinement.git`

Chapter 6

Development of Controller using Recurrent Neural Network (RNN)

6.1 Introduction

Traditional control strategies for ornithopters often rely on meticulously hand-crafted models that capture flight dynamics, aerodynamics, and actuator responses. However, these models can be complex, time-consuming to develop, and may not fully generalize to changing flight conditions or variations in the robotic bird's design. To address these challenges, we investigate a data-driven approach for building a controller using Deep Neural Networks (DNNs) and Recurrent Neural Networks (RNNs).

We are primarily interested in harnessing the capabilities of RNNs. Due to their inherent ability to process sequential information and maintain an internal memory, RNNs have the potential to learn complex input-output relationships directly from pre-recorded flight data. This black-box approach aims to capture the ornithopter's behavior without the need for explicit mathematical modeling of its dynamics. However, we know that actuators c1-roll and c2-pitch influence attitude control, and the combination of c1-roll, c2-pitch, and c3-throttle addresses guidance, our RNN-based controller aims to learn these mappings autonomously.

Eventually, we test and validate our RNN model with real flight datasets, exploring the possibilities to replace classical control.

6.2 Feature and Target Selection

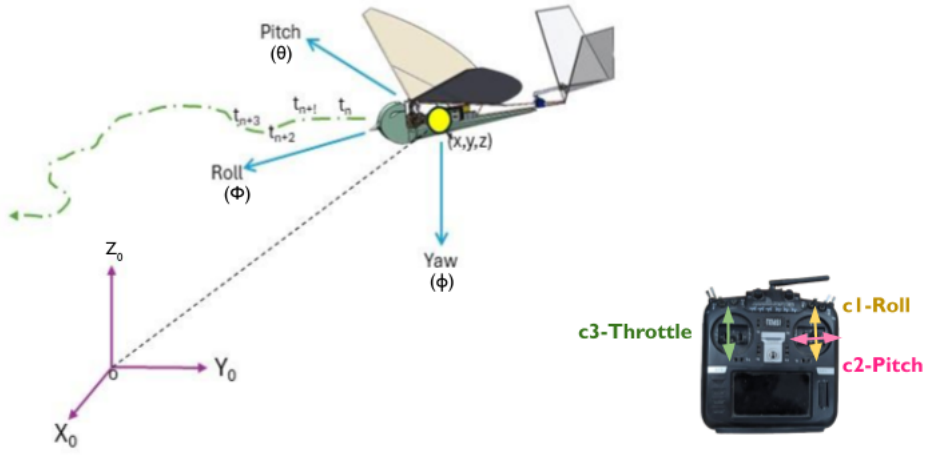


FIGURE 6.1: Schematic diagram of an instantaneous states against the instantaneous responses of c1-roll, c2-pitch and c3-throttle

After data preparation we are going to derive the controller using supervised DNN and RNN, as

- **X**: Normalised Sate variables, such as X=[x,y,z, ϕ, θ, ψ, $\triangle t$]. where positional value [x,y,z] is taken from inertial frame, and attitude is value [ϕ, θ, ψ, $\triangle t$]
- **y**: Normalized Target, which is control signals, that we want to predict, such as y=[$c1\_roll$, $c2\_pitch$, $c3\_throttle$]

6.3 Splitting Data for Training and Validation of the model

In ornithopter control system development, utilizing time-series data is paramount for capturing past behaviors and trends. Among various techniques, time-series data splitting is preferred. This method allocates earlier data for training and later data for validation, preserving temporal sequences crucial for accurate predictions in dynamic systems like ornithopters.

Time-Based Splitting:

Split the data chronologically. For instance, we use the first 75% for training and the remaining 25% for testing. This ensures the model is trained on data that follows the same temporal sequence as real-world scenarios.

Windowing:

Divide the data into fixed-size windows that slide along the timeline. Each window becomes a training instance. This approach is useful when dealing with very large datasets.

Walk-Forward Splitting:

Train the model on a specific window of data, then evaluate it on the next window. The window is then shifted forward for the next training iteration. This is particularly suitable for forecasting tasks where the model is continually updated with new data.

6.4 Working Principle of Neural Network in the context of DNN and RNN

At the core of this control system lies a deep neural network (DNN). DNNs excel at learning complex relationships between inputs and outputs from data, making them suitable for modeling the nonlinear dynamics of ornithopter flight. Let's delve into how a DNN functions:

6.4.1 Neurons and Layers

Neurons: The fundamental units of a DNN are artificial neurons. A neuron takes multiple inputs (X1, X2, ... Xn), multiplies each by a weight (w1, w2, ... wn), adds a bias term (b), and passes the result through a nonlinear activation function (f): output, y = f(w1*x1 + w2*x2 + ... + wn*xn + b);
Whereas, in our model inputs are X=[x,y,z, ϕ, θ, ψ, $\triangle t$] and output y=[$c1\_roll$, $c2\_pitch$, $c3\_throttle$]

Layers: Neurons are organized into layers. The input layer receives the ornithopter's state information X=[x,y,z, ϕ, θ, ψ, $\triangle t$]. Hidden layers process this information, with each layer's neurons computing new representations. The output layer produces the predicted control signals. Our ornithopter model comprises an input layer with 32 neurons for 7

input features, two hidden layers with 64 and 16 neurons utilizing 'ReLu' activation, and an output layer with 3 neurons employing 'linear' activation to predict control signals.

6.4.2 Activation Function [Goodfellow et al., 2016]

Linear combinations of inputs, as you'd find in a purely linear model, can only represent linear relationships. Activation functions introduce non-linearity, enabling neural networks to model complex, real-world phenomena like ornithopter dynamics. Also, The output of an activation function helps determine whether a neuron should "fire" (be activated) and to what extent, influencing the information flow within the network. There most commonly used activation functions are;

Sigmoid Function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Range: Smoothly squashes input values between 0 and 1.

Properties: Historically used but can suffer from vanishing gradients (very small gradients for large input magnitudes), which hinder the learning process.

Tanh Function:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Range: Maps input values between -1 and 1.

Properties: Often preferred over sigmoid due to being zero-centered, but still has the vanishing gradient issue.

ReLU (Rectified Linear Unit):

$$\text{ReLU}(x) = \max(0, x)$$

Range: Outputs 0 for negative inputs, and the input itself for positive values.
Properties: Computationally efficient and mitigates vanishing gradients for positive inputs, making it popular in deep networks. However, it can lead to "dying ReLU" (neurons stuck in 0 output state).

Leaky ReLU:

$$\text{Leaky ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{otherwise} \end{cases}$$

Range: Similar to ReLU, but with a small, non-zero slope for negative inputs.

Properties: Addresses the "dying ReLU" problem.

Linear Activation Function:

$$f(x) = x$$

Range: The output is the same as the input, covering the entire real number line.

Properties: The linear activation function is simply the identity function, meaning it doesn't introduce non-linearity. It's often used in output layers for regression tasks where the output range is unrestricted. Some times other linear activation function works well for output layer in regression problem. It works well in the modeling of ornithopter.

6.4.3 Learning: Backpropagation in the context of DNN

A DNN learns by minimizing a loss function (e.g., mean squared error). This quantifies the difference between predicted control signals and the true, desired control signals in the training data.

Backpropagation: Backpropagation plays a crucial role in training DNNs for control tasks, analogous to the role of error signals in classical control. During the feedforward pass, a DNN generates predictions. The difference between these predictions and the true values (from flight data) is the loss. Backpropagation systematically calculates gradients eq:6.1 with different layers for different neurons as figure 6.2, indicating how changes in

weights and biases will affect the loss. These gradients guide updates using an optimizer (like ADAM). The goal is to minimize the loss over iterations, implying that the DNN's predictions increasingly match the real control signals. This process allows us to train a model that can accurately predict control actions for unseen trajectories.

$$\begin{aligned} \delta_j^l &= \begin{cases} \frac{\partial L}{\partial a_j^L}\sigma'(z_j^L) & \text{if } l = L \text{ (output layer)} \\ \sum_k w_{kj}^{l+1}\delta_k^{l+1}\sigma'(z_j^l) & \text{otherwise (hidden layers)} \end{cases} \\ w_j^l &= w_j^l - \eta a_j^{l-1}\delta_j^l \\ b_j^l &= b_j^l - \eta\delta_j^l \end{aligned} \tag{6.1}$$

Where:

δ_j^l : Error signal of neuron j in layer l.

$\frac{\partial L}{\partial a_j^L}$: Partial derivative of the loss function (L) with respect to the activation a_j^L of output neuron j.

$\sigma'(z_j^l)$: Derivative of the activation function σ evaluated at the weighted input z_j^l of neuron j in layer l.

w_{kj}^{l+1} : Weight connecting neuron j in layer l to neuron k in layer $l+1$.

δ_k^{l+1} : Error signal of neuron k in layer $l+1$.

η : Learning rate.

a_j^{l-1} : Activation of neuron j in layer $l-1$.

b_j^l : Bias of neuron j in layer l.

Iterative Training: The training process involves feeding flight data, calculating errors, and updating parameters over many iterations. This allows the DNN to capture the intricate relationship between the ornithopter's state and the control signals needed to achieve stable and maneuverable flight.

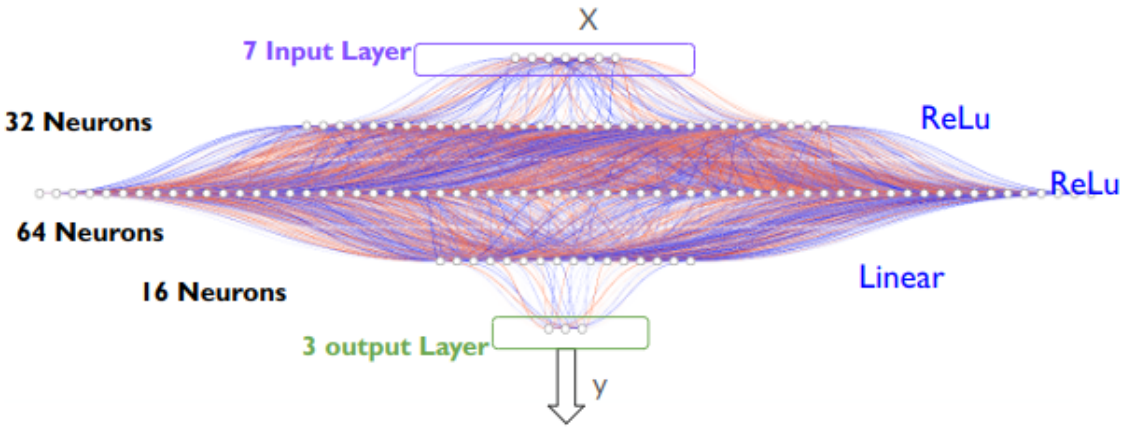


FIGURE 6.2: DNN Architecture for ornithopter control model

6.4.4 Learning: Backpropagation in the context of RNN

Backpropagation through time (BPTT)[BODEN, 2002] in RNNs adapts the core backpropagation algorithm to the time-dependent nature of Recurrent Neural Networks. Errors are calculated at each time step and propagated backwards through the network's structure and across time to guide weight updates with following equation (6.2). This allows RNNs to learn complex dynamics and long-term dependencies within sequential data. Much like how feedback loops are used in control systems [AGGARWAL, 2018] to correct deviations from a desired behavior, backpropagation in RNNs allows them to learn improved control strategies over time. This makes RNNs a powerful tool for controlling systems with nonlinear or poorly understood dynamics, particularly where system behavior evolves over time and is influenced by past states.

$$
\begin{aligned}
h_t &= \sigma(W_h h_{t-1} + W_x x_t + b_h) \\
\hat{y}_t &= \sigma(W_o h_t + b_o) \\
\delta_t &= y_t - \hat{y}_t \\
\frac{\partial E}{\partial W_o} &= \sum_t \delta_t h_t^T \\
\frac{\partial E}{\partial W_h} &= \sum_t \delta_t^T \sigma'(h_{t-1}) W_h^T \\
\frac{\partial E}{\partial W_x} &= \sum_t \delta_t^T \sigma'(h_{t-1}) x_t^T
\end{aligned}
\tag{6.2}
$$

Where;

x_t : Input at time t

h_t : Hidden state at time t

$\hat{y}_t$: Predicted output at time t

y_t : True target output at time t

W_h, W_x, W_o : Weight matrices for hidden-to-hidden, input-to-hidden, and hidden-to-output connections respectively.

b_h, b_o : Bias vectors for hidden and output layers, respectively.

σ : Activation function

E : The loss function (mean squared error)

Techniques to Address Backpropagation Challenges

- **LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) Networks**: These specialized RNN architectures incorporate gating mechanisms that control the flow of information within the network. This helps to mitigate vanishing and exploding gradients, allowing RNNs to learn effectively from longer sequences.
- **Gradient Clipping**: This technique limits the magnitude of gradients during backpropagation, preventing them from exploding and destabilizing the training process.
- **Truncated BPTT**: In some scenarios, backpropagation is limited to a fixed number of time steps instead of considering the entire sequence. This reduces computational complexity and helps address vanishing gradients for very long sequences.

6.4.5 Optimization (ADAM) [Goodfellow et al., 2016]:

An optimizer (like Adam) uses the gradients to update the weights and biases, nudging them in a direction that reduces the error.
ADAM maintains two moving averages of gradients, namely the first moment (m_t) and the second moment (v_t). These moving averages are computed using exponential decay:

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$$

where: - g_t is the gradient of the loss function with respect to the parameters at time step t. - β_1 and β_2 are the exponential decay rates for the first and second moments, typically set to values close to 1 (e.g., 0.9 and 0.999, respectively).

These moving averages are then used to update the parameters:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t} + \epsilon} \cdot m_t$$

where: - θ_t represents the parameters of the model at time step t. - η is the learning rate. - ϵ is a small constant (e.g., 10^{-8}) added to prevent division by zero.

ADAM also incorporates momentum by including a term that exponentially decays the past gradients. The update rule with momentum can be written as:

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$$

ADAM dynamically adjusts the learning rate for each parameter based on the estimates of the first and second moments of the gradients. This adaptiveness helps accelerate convergence and makes ADAM well-suited for optimizing neural networks.

6.5 Preference of RNN over DNN

Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) networks, are a type of neural network architecture designed for processing sequential data. Unlike traditional feedforward neural networks, RNNs have connections that loop back on themselves, allowing them to maintain information over time. LSTM networks, a variant of RNNs, address the vanishing gradient problem often encountered in standard RNNs. They achieve this by introducing memory cells and gating mechanisms that regulate the flow of information. This enables LSTMs to capture long-term dependencies in sequential data while avoiding the issue of vanishing gradients [Amidi and Heuritech].

In Figure 6.3, the process of passing control input data through time sequencing is illustrated. Each unit of operation comprises three hidden layers, with 32, 64, and 16 LSTM unit cells stacked [Verma et al., 2018], as depicted in Figure 6.4.

Within each unit cell, illustrated in Figure 6.5, information is filtered through various gates [Amidi and Heuritech]:

1. The Update gate Γ_u determines the relevance of past data at the current time step.
2. The Relevance gate Γ_r decides the extent to which previous information should be retained.
3. The Forget gate Γ_f signals the necessity to erase data that is no longer relevant.
4. The Output gate Γ_o dictates the amount of information to reveal to the next cell.

These gates collectively regulate the flow of information through the LSTM unit, enabling effective processing of sequential data.

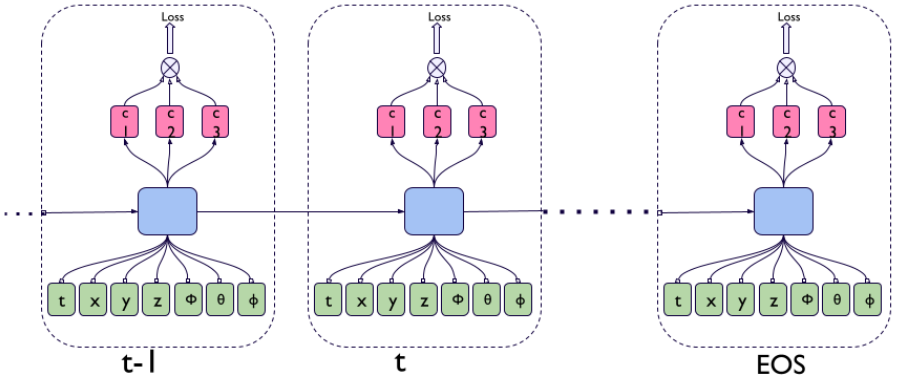


Figure 6.3: LSTM Time Sequence stacked

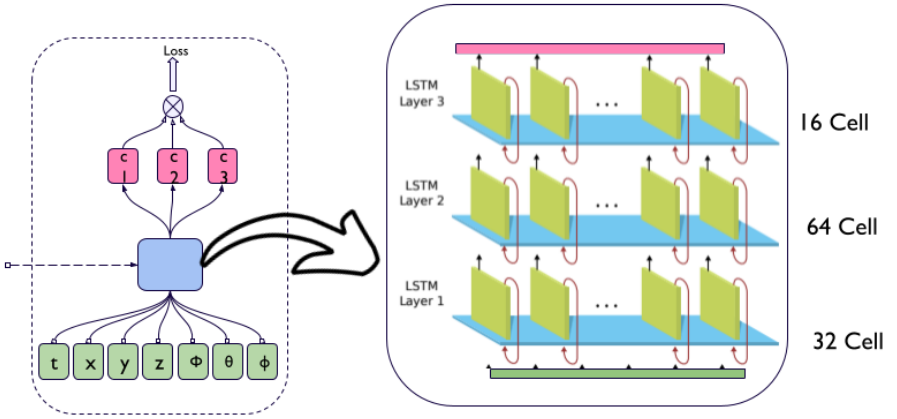


Figure 6.4: LSTM stack of neurons

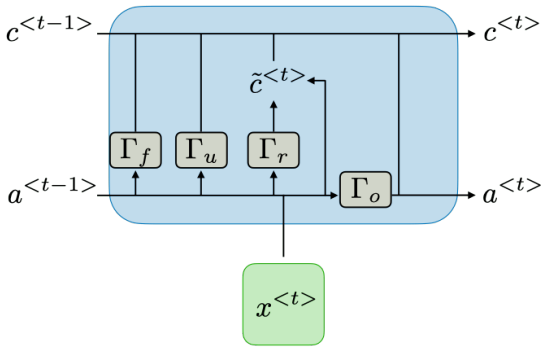


FIGURE 6.5: LSTM Unit Cell (Source: Stanford.edu)

$$
\begin{aligned}
c_t &= \Gamma_f \odot c_{t-1} + \Gamma_u \odot \tilde{c}_t \\
\tilde{c}_t &= \tanh\left(W_c[\Gamma_r \odot a_{t-1}, x_t] + b_c\right) \\
a_t &= \Gamma_o \odot \tanh(c_t)
\end{aligned}
\tag{6.3}
$$

6.5.1 Limitations of DNNs for Time Series Data

Inability to Capture Long-Term Dependencies: DNNs struggle with sequential data where relationships between elements can be distant. In ornithopter control, past flight states (e.g., roll, pitch and yaw angles a few seconds ago) might influence the optimal control adjustments now. However, DNNs wouldn't effectively capture these long-range dependencies.

6.5.2 Advantages of LSTMs for Ornithopter Control

Learning Long-Term Dependencies: LSTMs excel at modeling these long-term relationships in sequential data. By remembering past flight states through their cell state, they can learn how past dynamics influence necessary control actions for stable flight.

Handling Variable-Length Sequences: Real-world flight data might have varying lengths due to external factors. LSTMs are built to handle sequences of different lengths, unlike DNNs which require fixed-size inputs.

Focus on Recent Information: LSTM gates prioritize information from recent time steps during processing, crucial for control systems where recent flight states have the most significant impact on current control decisions.

6.6 Controller Development using DNN and LSTM

We, pre-processed flight data for use in an offline, supervised machine learning regression task. Our goal is to generate open-loop control signals to guide an ornithopter along new trajectories. To this end, we compare Deep Neural Networks (DNNs) and Recurrent Neural Networks (RNNs) as learning techniques using Tensorflow[GOOGLE] Library. Due to the time-dependent nature of flight dynamics, RNNs, particularly those with LSTM units, are expected to excel in modeling these temporal relationships. The algorithm 5 and 6 explain as follows.

Algorithm 5 DNN-Based Ornithopter Control Model Development

▷ Data Preparation
$X \leftarrow df[['time\_diff\_norm','roll','pitch','yaw','x\_norm','y\_norm','alt\_norm']]$
$y \leftarrow df[['c1\_roll','c2\_pitch','c3\_throttle']]$
$training\_size \leftarrow \text{INT}(0.75)()\text{len}df$
$X\_train, X\_test \leftarrow df.loc[: training\_size, :], df.loc[training\_size :, :]$
$y\_train, y\_test \leftarrow df.loc[: training\_size, :], df.loc[training\_size :, :]$
▷ Neural Network Model Definition
$model \leftarrow \text{SEQUENTIAL}$
$model.add(\text{DENSE}(32, activation = 'relu', input\_shape = (7,)))$
$model.add(\text{DENSE}(64, activation = 'relu'))$
$model.add(\text{DENSE}(16, activation = 'relu'))$
$model.add(\text{DENSE}(3, activation = 'linear'))$
$optimizer \leftarrow \text{ADAM}(learning\_rate = 0.01)$
$model.compile(loss = 'mean\_squared\_error', optimizer = optimizer, metrics = ['accuracy'])$
▷ Model Training
$history \leftarrow model.fit(X\_train, y\_train, epochs = 250, batch\_size = 128, validation\_data = (X\_test, y\_test), verbose = 1)$
▷ Evaluation
$training\_loss \leftarrow history.history['loss']$
$validation\_loss \leftarrow history.history['val\_loss']$
... code for creating the loss plot ...
$score \leftarrow model.evaluate(X\_test, y\_test, verbose = 0)$

Algorithm 6 LSTM-Based Ornithopter Control Model Development

▷ Data Preparation
$X \leftarrow df[['time\_diff\_norm','roll','pitch','yaw','x\_norm','y\_norm','alt\_norm']]$
$y \leftarrow df[['c1\_roll','c2\_pitch','c3\_throttle']]$
$training\_size \leftarrow \text{INT}(0.75 \cdot)()\text{len}df$
$X\_train, X\_test \leftarrow df.loc[: training\_size, :], df.loc[training\_size :, :]$
$y\_train, y\_test \leftarrow df.loc[: training\_size, :], df.loc[training\_size :, :]$
▷ LSTM Model Definition
$model\_LSTM \leftarrow \text{SEQUENTIAL}$
$model\_LSTM.add(\text{LSTM}(32, return\_sequences = True, input\_shape = (X\_train.shape[1], 1)))$
$model\_LSTM.add(\text{LSTM}(64))$
$model\_LSTM.add(\text{DENSE}(16, activation =' relu'))$
$model\_LSTM.add(\text{DENSE}(3, activation =' linear'))$
$optimizer \leftarrow \text{ADAM}(learning\_rate = 0.01)$
$model\_LSTM.compile(loss =' mean\_squared\_error', optimizer = optimizer, metrics = ['accuracy'])$
▷ Model Training
$history\_LSTM \leftarrow model\_LSTM.fit(X\_train, y\_train, \ldots \ training\ parameters \ \ldots)$
▷ Evaluation
$training\_loss\_LSTM \leftarrow history\_LSTM.history['loss']$
$validation\_loss\_LSTM \leftarrow history\_LSTM.history['val\_loss']$
... code for creating the loss plot ...
$score \leftarrow model\_LSTM.evaluate(X\_test, y\_test)$

6.7 Evaluation of Models

After running the model, as expected, LSTM performs slightly better than DNN. Despite DNN having less complexity, it converges faster than LSTM. However, after 100 epochs, as shown in Figure 6.6, the loss of LSTM is lower than that of DNN.

Mean Squared Error (MSE) is one of the most common loss functions used for regression tasks, including predicting control signals in machine learning models. It calculates the average of the squared differences between predicted values and their corresponding true target values.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Where:

- n: Number of samples in your dataset.
- y_i: True target value (e.g., the correct control signal for a data point).
- $\hat{y}_i$: Predicted value by your model.

Magnitude of Errors: MSE directly amplifies larger errors due to the squaring operation. This means the model is strongly penalized for making inaccurate predictions.

Smoothness: MSE has a smooth gradient, which generally leads to well-behaved updates during the model training process.

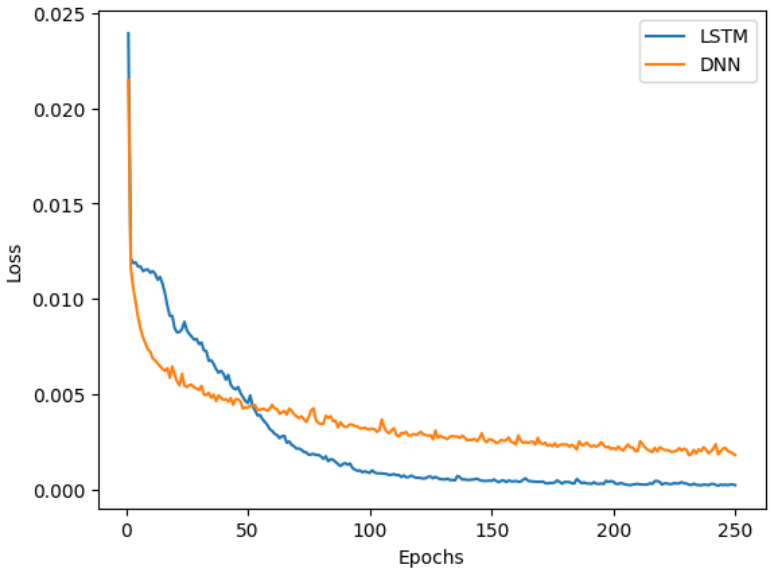


FIGURE 6.6: Loss comparison between DNN and LSTM

In the results, the test accuracy for DNN in evaluating the model is around 89%, while for LSTM, it is 94%. This indicates that the models accurately capture the intricate behavior of the data.

6.7.1 Testing Trained Model

To assess how well our trained AI model captures the input-output behavior of the ornithopter, we conducted an experiment. We commanded the ornithopter to follow a specific flight trajectory (defined by its states over time in the ML model) and recorded the required control inputs (c1-roll, c2-pitch, and c3-throttle). Our trained AI model then attempted to predict these control inputs based on the same state information.

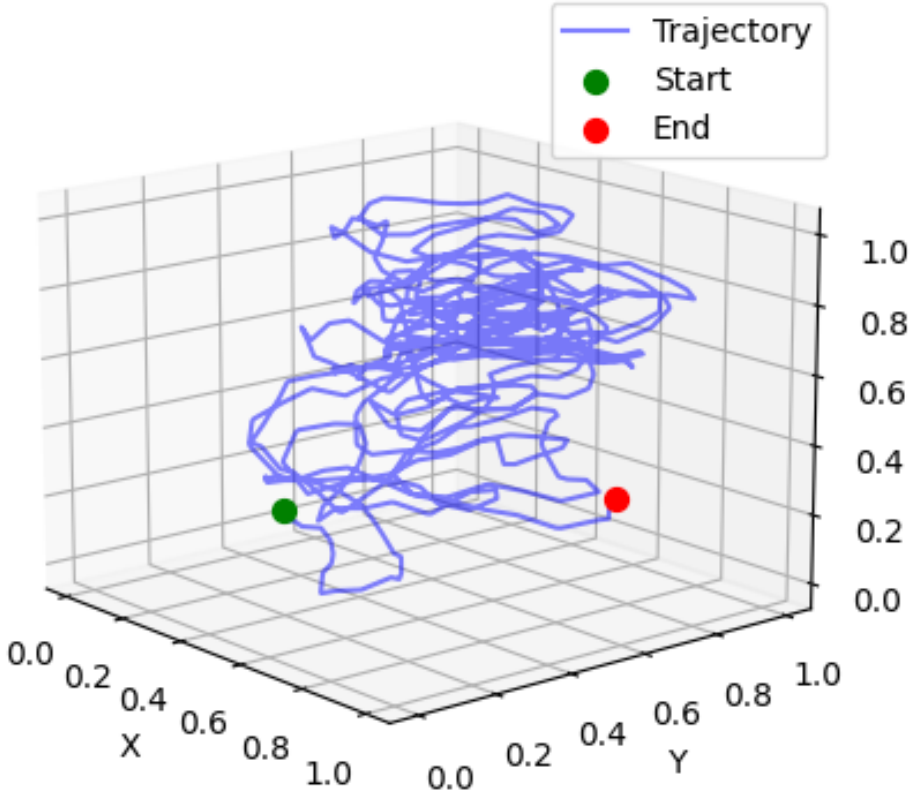


FIGURE 6.7: Training Trajectory(about 10 min flight time)

The provided figure illustrates the deviations between the AI model's predictions and the true control inputs used during the experiment. Achieving an accuracy of 94% demonstrates that the model has learned to successfully replicate the control actions needed to guide the ornithopter along the given trajectory.

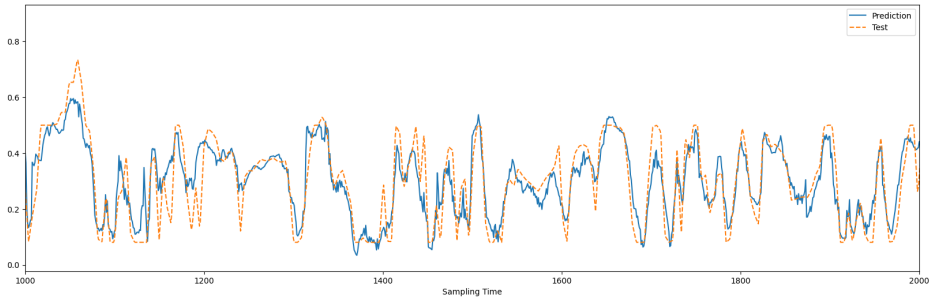


FIGURE 6.8: Control response; c1-roll

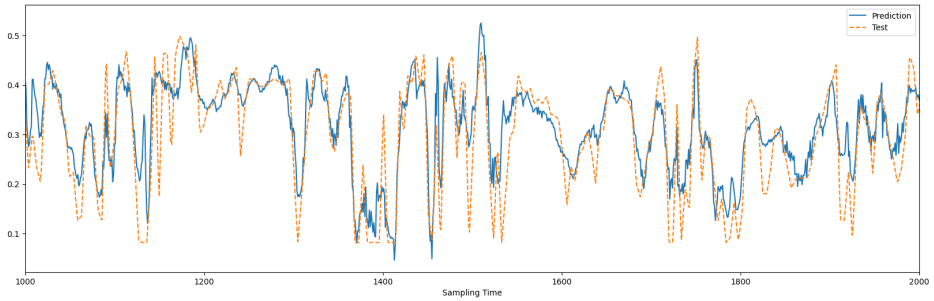


FIGURE 6.9: Control response; c2-pitch

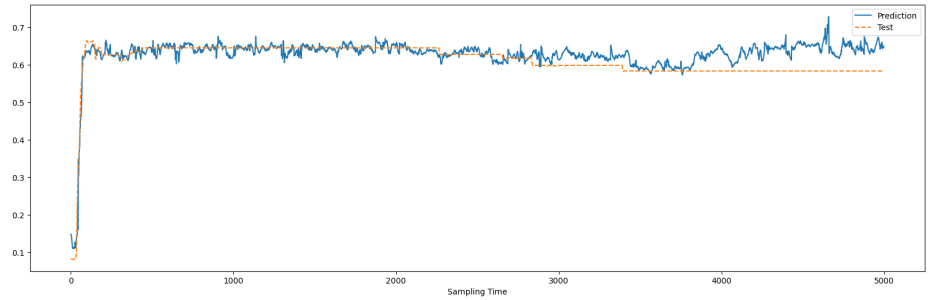


FIGURE 6.10: Control response; c3-throttle

6.8 Validation with different flight with same model

To evaluate our trained RNN(LSTM) model, we tested it on a completely new flight trajectory of the same ornithopter. The goal was to compare the model's predicted control signals (c1-roll, c2-pitch, c3-throttle) with the actual control signals used during the flight.

The flight data, consisting of approximately 8 minutes of flight time, was resampled at 10 Hz, resulting in 5000 sample points. For clarity, we plot a 1.6-minute segment (samples 3000-4000) of the results, comparing the predicted and actual control signals side-by-side.

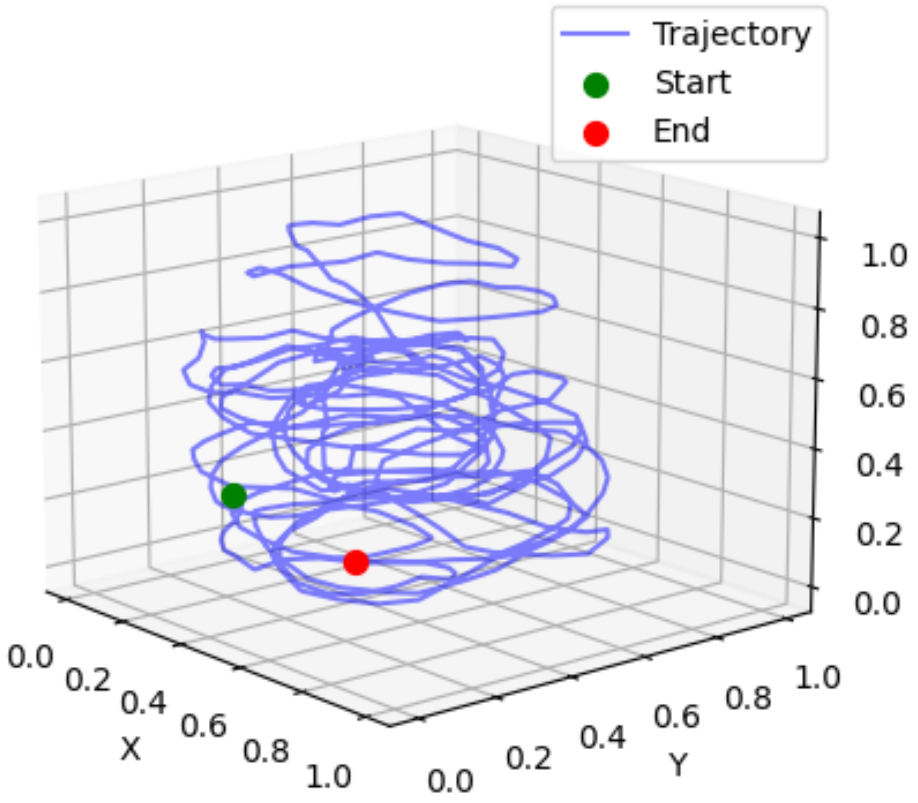


FIGURE 6.11: Testing Trajectory (about 8 min flight time)

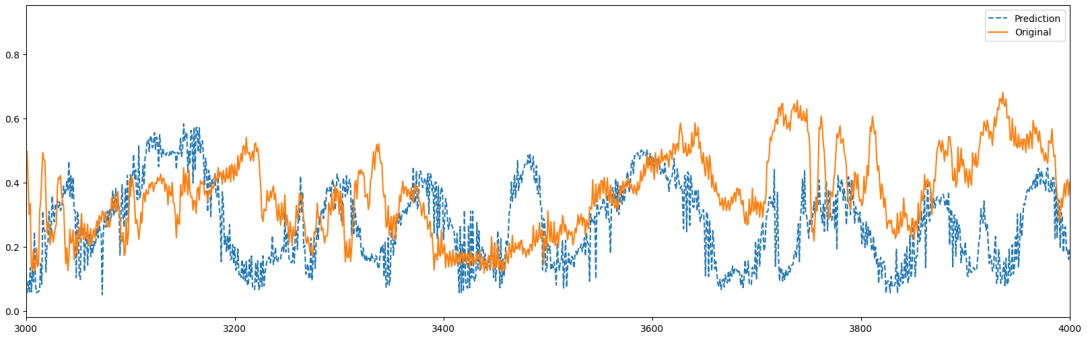


FIGURE 6.12: Controller c1-roll; Prediction vs Actual responses

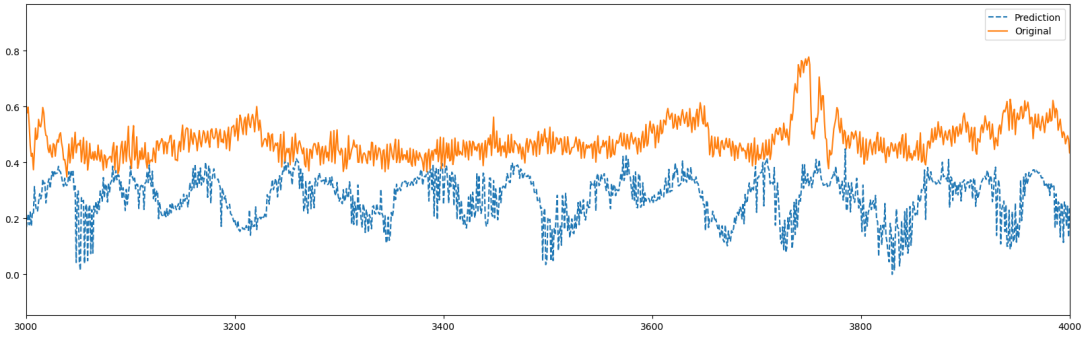


FIGURE 6.13: Controller c2-pitch; Prediction vs Actual responses

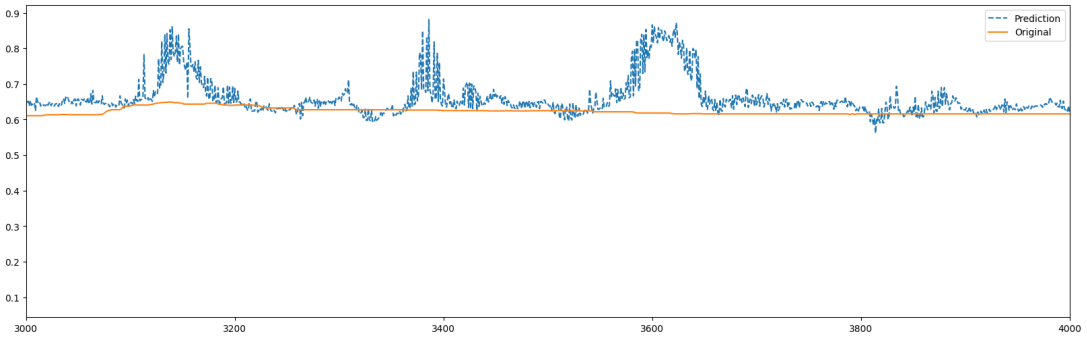


FIGURE 6.14: Controller c3-throttle; Prediction vs Actual responses

TABLE 6.1: RNN Model Prediction Errors on Time Series Data

Parameters	MSE	MAE	RMSE
C1-Roll	0.042	0.166	0.207
C2-Pitch	0.076	0.2412	0.276
C3-Throttle	0.014	0.086	0.118

Mean Squared Error (MSE): The MSE value suggests a low average squared deviation between predictions and true control signals. However, the practical implications of this error magnitude depend entirely on the specific control signals used for the ornithopter and the performance requirements of the flight control system. Understanding the units of the control signals and the allowable error margins is crucial for a meaningful interpretation of the MSE.

Mean Absolute Error (MAE): The MAE of values indicates that the model's control signal predictions deviate from the true values by an average of 0.164 units.

Root Mean Squared Error (RMSE): The RMSE values, being very close to the MAE, implies a lack of extreme outliers in the model's predictions. This suggests relatively consistent errors rather than occasional large deviations.

6.9 Potential Sources of Error

The observed offset between predicted and actual control data could stem from several factors:

1. Unmodeled Environmental Influences: Environmental disturbances such as wind gusts and varying wind speeds can significantly influence the ornithopter's flight dynamics. If these conditions are not included as parameters in the training model, discrepancies between predicted and real-world control responses are likely.

Possible Improvement: Incorporate a sensor capable of measuring 3D time-series wind velocity data to enhance the model's ability to adapt to changing environmental conditions.

2. Ornithopter Structural Imperfections: The delicate nature of the ornithopter model renders it susceptible to changes in flight behavior due to collisions or other events that cause weight distribution imbalances. These imperfections can introduce unpredictable variations in control responses.

Possible Improvement: Regular inspection and calibration of the ornithopter model might mitigate the impact of structural imperfections on prediction accuracy.

Chapter 7

Future Scope: Integration of Reinforcement Learning (RL)

7.1 Introduction

Imagine a future where ornithopters, those fascinating flapping-wing aircraft, can navigate through complex environments with ease, just like birds. We've made significant strides in designing control systems for them, using the technologies like Deep Neural Networks and Recurrent Neural Networks. But there's still work to be done.

We need these ornithopters to handle real-world challenges, like unexpected noises or obstacles that might throw them off course. That's where our vision for the future comes in. We want to teach these flying machines how to adapt and learn from their experiences using something called Reinforcement Learning.

The ornithopters learning to fly better and smarter by practicing in virtual environments, kind of like video games. We'll create these special simulated worlds using tools like OpenAI GYM and the Pybullet physics engine. In these digital arenas, ornithopters will try out different moves and learn from their successes and failures.

Our goal is to develop a control system that's not just smart but also resilient, capable of handling whatever challenges come its way. By embracing Reinforcement Learning(RL), we're paving the way for ornithopters to soar through the skies with confidence, even in the face of uncertainty.

7.2 Agent

The learner and decision maker in RL called agent. Agent's goal is to take the action intelligently based on state and reward. Agent, environment and action can be analogous to controller, controller system (plant) and control signal [Sutton and Barto, 2018]in control problem. In the case agent is our ornithopter model as depicted in fig 7.1 precisely, controller(c1-Roll,c2-Pitch and c3-Throttle) actuation.

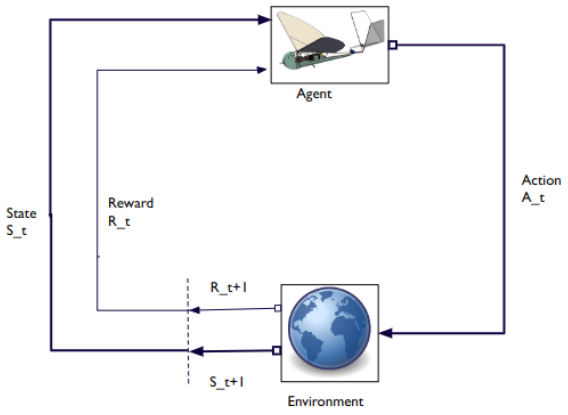


Figure 7.1: The agent environment interaction in a MDP

7.3 Environment

Sutton and Barto[Sutton and Barto, 2018] consider the environment to be anything outside of the agent itself. They emphasize that the agent and environment interact continuously, with the agent choosing actions and the environment responding to those actions and presenting new situations to the agent.

7.4 Building MDP for RL Framework

Reinforcement learning (RL) offers a unique way to train agents. Instead of giving explicit instructions, we set up an environment where the agent explores and learns through trial and error. The environment provides feedback in the form of rewards (or penalties), guiding the agent towards better decision-making. The Markov Decision Process (MDP) provides a powerful mathematical framework for modeling this type of problem, where

outcomes depend on both the agent's choices and some inherent randomness. Key components of an MDP include: States (S), Actions (A), Transition Probabilities (P), Rewards (R), and a Discount Factor (γ).

7.4.1 States (S)

The set of possible situations the agent can be in at a specific time-step in the environment. Specifically, way-point Information in time series: which includes Roll ϕ, Pitch θ, Yaw ψ: rotational orientation and x, y, z: Position in the 3D environment of the ornithopter can be consider as state.
s_t=[x_t, y_t, z_t, ϕ_t, θ_t, ψ_t]; where s_t $\in$S

7.4.2 Actions (A)

The set of possible choices the agent can make, given time step in order to interact with its environment.

In our frame work the ornithopter agent will take the action $\mathbb{A}$ based on state $\mathbb{S}$ and $\mathbb{R}$
c_t=[$c1_t$, $c2_t$, $c3_t$]; where c_t $\in$A

- Action Space($\mathbb{A}$):
  - Continuous: Allows for finer control (smooth adjustments).
  - Discrete: Allows for limited maneuverability.

7.4.3 Transition Probabilities (P)

Transition probabilities are key to capturing the inherent uncertainty and dynamic behavior of ornithopter model. While aerodynamics, inertia, and actuator limitations provide a deterministic baseline, accurately representing how the ornithopter moves from one state to another often requires incorporating randomness. These probabilities might model wind disturbances, slight delays in motor responses, or even sensor noise. By encoding these probabilities, reinforcement learning agent can learn to make robust decisions that account for the real-world unpredictability it will face when deployed.

7.4.4 Rewards (R)

The designed reward function aims to guide the reinforcement learning agent towards precise path tracking and efficient goal completion while maintaining flight stability.

The considered parameters are;
Path Tracking: Large positive reward for staying on the path, negative for deviating.
Goal Proximity: Positive for getting closer to the next waypoint.
Smoothness: Penalize jerky movements (large, sudden changes in control signals) and reward energy efficiency.
Waypoint Completion: Large positive reward when reaching a target waypoint. The reward function is defined as:

$$R(s_t, c_t) = w_1 \cdot (s_t, WP) + w_2 \cdot (s_t, wp_{\text{current}}) + w_3 \cdot (c_t, c_{t-1}) + w_4 \cdot (s_t, wp_{\text{current}})$$

$s_t \in \mathbb{S}$: The ornithopter's state at time step t (i.e., position and orientation)
$c_t \in \mathbb{A}$: The action taken by the agent at time step t (i.e., motor control signals)
WP : The set of waypoints defining the desired path
wp_{current} : The current target waypoint
$d(\cdot, \cdot)$: A distance function
w_1, w_2, w_3, w_4 : Weights determining the relative importance of each reward component.

7.4.5 Discount Factor (γ)

It indicates the level of significance attributed to immediate rewards versus future rewards.

$\mathrm{G}_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$, where $0 \leq \gamma \leq 1$.

7.5 Policy (π)

The policy(π) in RL, is a decision making strategy that agent uses to select the action based on it current state ($\pi(a/s)$). There are two main types of policies;

Deterministic Policy: The Policy directly maps a state to a specific control action.

$\pi(s_t) = c$; $c \in \mathbb{A}$ & $s_t \in \mathbb{S}$
In practical ornithopter models, this equation may not be applicable due to our limited understanding of the intricate dynamics involved in flapping motion.

Stochastic Policy: The policy defines a probability distribution over possible control actions, conditioned on the current state.

$$\pi(C|s_t) = P(C = [c_1, c_2, c_3]|s_t) \; ; \; c \in \mathbb{A} \; \& \; s_t \in \mathbb{S}$$

In ornithopter model RNN architectures can be used as stochastic policy. Where the RNN take input the current state s and directly suggest a probability distribution of action c. The primary goal of agent is to learn optimal policy π^* that maximizes the expected cumulative reward over time. Mathematically, this can be expressed as:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \cdot R(s_t, a_t)\right]$$

Where: $\mathbb{E}[\cdot]$ denotes the expectation; γ is the discount factor; $R(s_t, a_t)$ is the reward function providing feedback based on the state and action;

7.6 Create the custom OpenAI-GYM Environment

7.6.1 Building URDF model

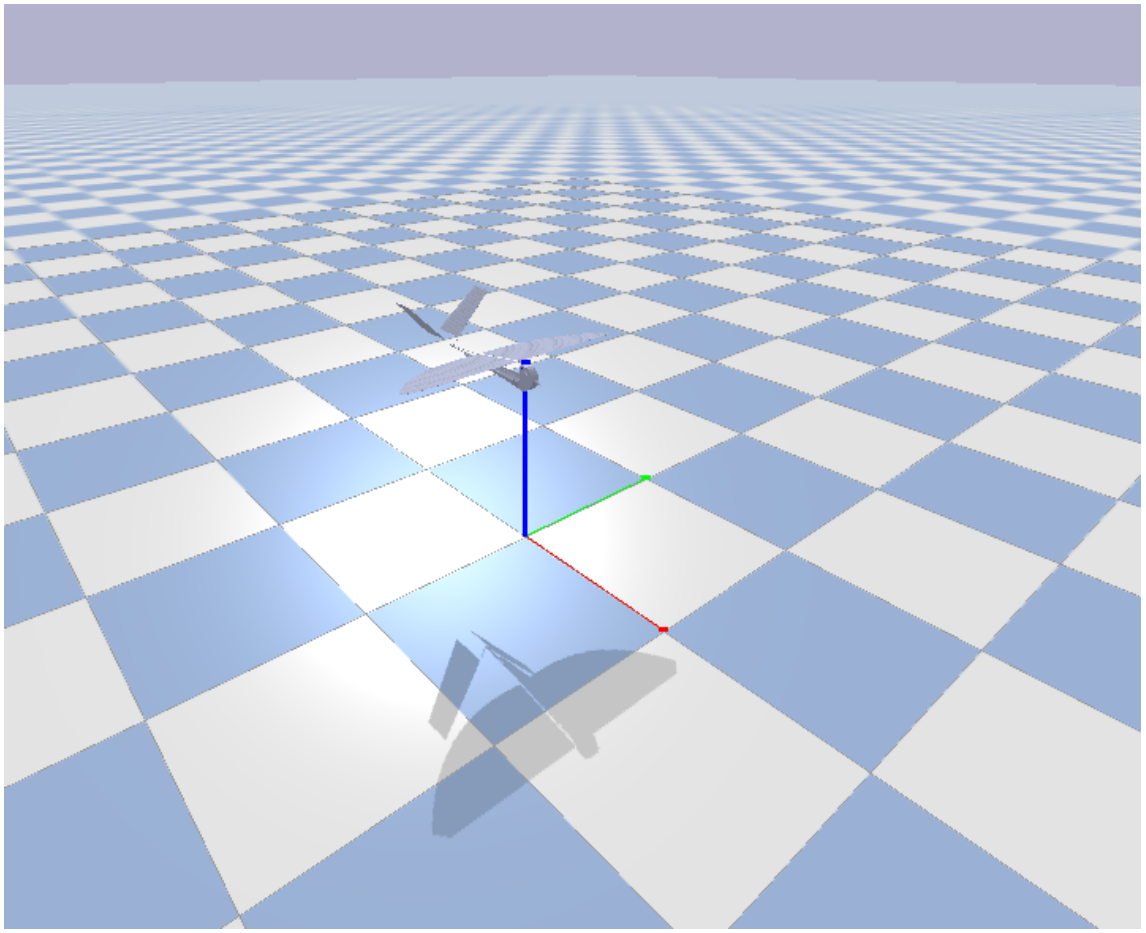


FIGURE 7.2: URDF ornithopter model, in Pybullet Environment

```
https://github.com/dharambirpoddar/BIrd_urdf.git
```

LISTING 7.1: URDF code for the bird.

The Unified Robot Description Format(URDF) is an XML-based language widely used in robotics to define the structure and properties of robots. Such as Joints, mass, Inertia tensors and collision information as shown in figure 7.2.

Reinforcement learning's emphasis on trial and error poses a significant challenge when training physical robots, especially those prone to accidents, like flying vehicles. Direct training runs the risk of severe damage, altering the robot's geometry and compromising the learning process. This is where URDF and physics simulators like PyBullet [BULLET

PHYSICS, Year accessed],GYM [OPENAI, Year accessed] become invaluable. By creating a URDF-based digital twin of the robot, we gain a safe virtual environment to explore complex flight dynamics. Simulators like PyBullet realistically model forces such as drag, downwash, thrust, and gravity. This allows the reinforcement learning agent to experiment with random control inputs, experience the consequences (both rewards and punishments), and refine its behavior without physical risk. This is like a digital twin for the robot, and in the learning phase, protecting the physical hardware while accelerating the policy development.

7.7 Test and Deploy

7.7.1 System configuration for Sim-to-Real

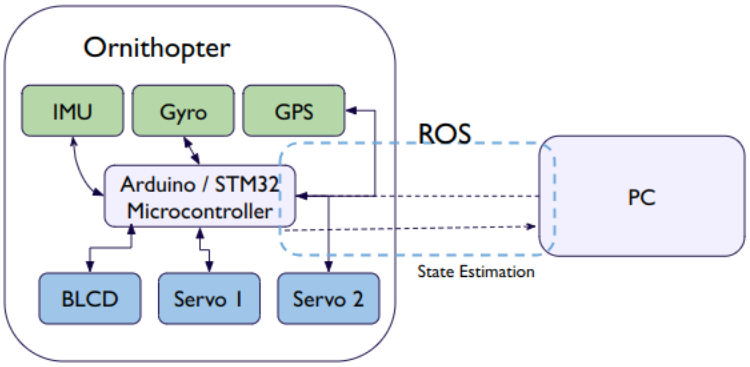


FIGURE 7.3: System Configuration

Once the ornithopter model achieves robust control within a simulated environment like OpenAI Gym or Gazebo, leveraging a physics engine that accurately captures real-world dynamics, we can transition to autonomous flight. To enable this, we outfit the ornithopter platform with a suite of hardware: BLDC motors for propulsion, servo motors for wing articulation, a high-fidelity IMU-GPS combo for accurate 3D pose estimation, and a capable microcontroller. ROS (Robot Operating System) will serve as the middleware to seamlessly bridge communication between the onboard microcontroller and a ground station. This architecture facilitates real-time exchange of critical data—control signals, IMU-derived quaternions for orientation representation, and GPS coordinates—empowering autonomous navigation and control as demonstrated in similar simulation-to-reality transfer studies [INOUE ET AL., 2024a].

7.8 Conclusion

This thesis embarked on a comprehensive journey towards building an autonomous ornithopter. Beginning with the fundamental design of the physical ornithopter, we explored various wing configurations to optimize efficiency. Experimental testing yielded a novel wing design that delivered a 13% increase in both lift and thrust, promising enhanced flight performance. To refine the ornithopter's kinematics, we employed data-driven techniques like DNNs and LSTMs. Ultimately, we transitioned to reinforcement learning to develop a robust control system, leveraging its adaptability in the face of complex dynamics.

While this work achieved significant advancements, the goal of a fully autonomous ornithopter remains an ongoing pursuit. Future directions include continuing to optimize the physical design, exploring more sophisticated reinforcement learning algorithms, and ultimately bridging the gap between simulation and real-world implementation.

Bibliography

Look up in the sky—it's a bird, it's a plane, it's a nano-hummingbird. https://www.avinc.com/resources/case-studies/view/look-up-in-the-skyits-a-birdits-a-plane-its-anano-hummingbird. Accessed: [insert date here].

C. C. Aggarwal. *Neural Networks and Deep Learning*. Springer, Cham, 2018. ISBN 978-3-319-94463-0. URL https://www.springer.com/gp/book/9783319944630.

S. Amidi and Heuritech. Cheatsheet - recurrent neural networks. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks. Accessed: Insert Date Accessed.

S. S. Baek, F. L. G. Bermudez, and R. S. Fearing. Flight control for target seeking by 13 gram ornithopter. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2674–2681. IEEE, 2011.

A. Banerjee, S. K. Ghosh, and D. Das. Aerodynamics of flapping wing at low reynolds numbers: force measurement and flow visualization. *International Scholarly Research Notices*, 2011, 2011.

J. Bhowmik. *Aerodynamics of Flapping Wing and Development of Ornithopter*. PhD thesis, IIT Kanpur, 2017.

J. Bhowmik, D. Das, and S. K. Ghosh. Aerodynamic modelling of flapping flight using lifting line theory. *International Journal of Intelligent Unmanned Systems*, 1(1):36–61, 2013.

J. Bhowmik, N. Raj, D. Das, and A. Abhishek. Measurement and analysis of aerodynamic forces of an ornithopter in free flight. In *Proceedings of International Conference on Intelligent Unmanned Systems*, volume 10, 2014.

A. T. Bode-Oke, S. Zeyghami, and H. Dong. Flying in reverse: kinematics and aerodynamics of a dragonfly in backward free flight. *Journal of The Royal Society Interface*, 15(143):20180102, 2018.

M. Boden. A guide to recurrent neural networks and backpropagation. *the Dallas project*, 2(2):1–10, 2002.

Bullet Physics. Bullet Physics SDK. `https://github.com/bulletphysics/bullet3`, Year accessed.

L. Chen, Y. Zhang, and J. Wu. Study on lift enhancement of a flapping rotary wing by a bore-hole design. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 232(7):1315–1333, 2018.

Dharambir Poddar. Bird Resampling Refinement. `https://github.com/dharambirpoddar/Bird_Resampling_Refinement.git`, 2024.

DharambirPoddar. BIW_ForceCalculation. URL `https://github.com/dharambirpoddar/BIW_ForceCalculation`.

R. Dvořák. Aerodynamics of bird flight. In *EPJ Web of Conferences*, volume 114, page 01001. EDP Sciences, 2016.

C. P. Ellington. The aerodynamics of hovering insect flight. i. the quasi-steady analysis. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 305 (1122):1–15, 1984.

Festo. Bionicswift. URL `https://www.festo.com/us/en/e/about-festo/research-and-development/bionic-learning-network/highlights-from-2015-to-2017/bionicswift-id_326830/`.

N. Geographic. How do birds fly? — national geographic, 2013. URL `https://www.youtube.com/watch?v=lOrH0Gbt8YM`. Accessed on 2024-05-07.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, Cambridge, MA, 2016. ISBN 978-0262035613. URL `https://www.deeplearningbook.org/`.

Google. Tensorflow. `https://www.tensorflow.org/`. Accessed: Insert Date Accessed.

S. Inoue, K. Kawaharazuka, K. Okada, and M. Inaba. Body design and gait generation of chair-type asymmetrical tripedal low-rigidity robot. *arXiv preprint arXiv:2404.05932*, 2024a.

S. Inoue, K. Kawaharazuka, K. Okada, and M. Inaba. Body design and gait generation of chair-type asymmetrical tripedal low-rigidity robot. *arXiv preprint arXiv:2404.05932*, 2024b.

Z. Jiao, L. Wang, L. Zhao, and W. Jiang. Hover flight control of x-shaped flapping wing aircraft considering wing–tail interactions. *Aerospace Science and Technology*, 116:106870, 2021.

C.-k. Kang and W. Shyy. Scaling law and enhancement of lift generation of an insect-size hovering flexible wing. *Journal of The Royal Society Interface*, 10(85):20130361, 2013.

M. Karásek, F. T. Muijres, C. De Wagter, B. D. Remes, and G. C. De Croon. A tail-less aerial robotic flapper reveals that flies use torque coupling in rapid banked turns. *Science*, 361(6407):1089–1094, 2018.

H. Kim, M. Jordan, S. Sastry, and A. Ng. Autonomous helicopter flight via reinforcement learning. *Advances in neural information processing systems*, 16, 2003.

K.Nandan. Footage of flapping flight, 2002. URL `https://www.youtube.com/watch?v=ZOxRaWae88k`.

F. Leys, D. Reynaerts, and D. Vandepitte. Outperforming hummingbirds' load-lifting capability with a lightweight hummingbird-like flapping-wing mechanism. *Biology open*, 5(8):1052–1060, 2016.

H. Liu, S. Ravi, D. Kolomenskiy, and H. Tanaka. Biomechanics and biomimetics in insect-inspired flight systems. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 371(1704):20150390, 2016.

National Geographic. Mystery of the Monarch Butterfly — National Geographic, 2015. URL `https://www.youtube.com/watch?v=cJJowVxiaRU`. Accessed on 2024-05-07.

National Geographic. How Do Animals Experience Pain? — National Geographic, 2017. URL `https://www.youtube.com/watch?v=iJi61NAIsjs`. Accessed on 2024-05-07.

OpenAI. OpenAI Gym. `https://github.com/openai/gym`, Year accessed.

J. Panerati, H. Zheng, S. Zhou, J. Xu, A. Prorok, and A. P. Schoellig. Learning to fly—a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7512–7519. IEEE, 2021.

W. Send, M. Fischer, K. Jebens, R. Mugrauer, A. Nagarathinam, and F. Scharstein. Artificial hinged-wing bird with active torsion and partially linear kinematics. In *Proceeding of 28th Congress of the International Council of the Aeronautical Sciences*, volume 10. Brisbane, Australia, 2012.

S. Sunada and K. Tsuji. Advantages of an ornithopter versus an airplane with a propeller. *TRANSACTIONS OF THE JAPAN SOCIETY FOR AERONAUTICAL AND SPACE SCIENCES*, 56(5):277–285, 2013.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018.

TED. How do birds learn to sing? — Partha P. Mitra — TED Institute, 2016. URL `https://www.youtube.com/watch?v=Lw2dfjYENNE`. Accessed on 2024-05-07.

S. Verma, G. Novati, and P. Koumoutsakos. Efficient collective swimming by harnessing vortices through deep reinforcement learning. *Proceedings of the National Academy of Sciences*, 115(23):5849–5854, 2018.

H. Wang, L. Zeng, H. Liu, and C. Yin. Measuring wing kinematics, flight trajectory and body attitude during forward flight and turning maneuvers in dragonflies. *Journal of Experimental Biology*, 206(4):745–757, 2003.

J. Whitaker, M. M. Blaschek, and D. Rylov. pyproj: Geodesic computations, projections and coordinate system transformations. `https://pypi.org/project/pyproj/`, 2022.

Wikipedia contributors. Universal Transverse Mercator coordinate system – Wikipedia. `https://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system`, 2024a. [Online; accessed 5-May-2024].

Wikipedia contributors. World Geodetic System – Wikipedia. `https://en.wikipedia.org/wiki/World_Geodetic_System`, 2024b. [Online; accessed 5-May-2024].

W. Yang, B. Song, et al. Experimental investigation of aerodynamics of feather-covered flapping wing. *Applied bionics and biomechanics*, 2017, 2017.

W. Yang, L. Wang, and B. Song. Dove: A biomimetic flapping-wing micro air vehicle. *International Journal of Micro Air Vehicles*, 10(1):70–84, 2018a.

W. Yang, L. Wang, and B. Song. Dove: A biomimetic flapping-wing micro air vehicle. *International Journal of Micro Air Vehicles*, 10(1):70–84, 2018b.

M. Y. Zakaria, H. E. Taha, and M. R. Hajj. Design optimization of flapping ornithopters: The pterosaur replica in forward flight. *Journal of Aircraft*, 53(1):48–59, 2016.

R. Zufferey, J. Tormo-Barbero, D. Feliu-Talegón, S. R. Nekoo, J. Á. Acosta, and A. Ollero. How ornithopters can perch autonomously on a branch. *Nature Communications*, 13 (1):7713, 2022.