Integrating Convolutional and Recurrent Networks for Synthetic-Data-Based PAV Navigation

A THESIS

submitted to the

INDIAN INSTITUTE OF TECHNOLOGY INDORE

in partial fulfillment of the requirements for

the award of the degree

of

MS(Research)

By

Prajjval Dhiman



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

May 2025

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **Integrating Convolutional and Recurrent Networks for Synthetic-Data-Based PAV Navigation** in the partial fulfillment of the requirements for the award of the degree of **MS(Research)** and submitted in the **Department of Computer Science and Engineering, Indian Institute of Technology Indore,** is an authentic record of my own work carried out during the time period from August 2023 to May 2025. The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

24 May 2025

Signature of the Student with Date

(Prajjval Dhiman)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

24/5/25

Signature of Thesis Supervisor with Date

(Dr. Gourinath Banda)

Prajjval Dhiman has successfully given his MS(Research) Oral Examination held on *15 June 2026*

15/06/26

Signature of Thesis Supervisor

Date:

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my heartfelt gratitude to a number of persons who in one or the other way contributed by making this time as learnable, enjoyable, and bearable. At first, I would like to thank my supervisor **Dr. Gourinath Banda**, who has been a constant source of inspiration during my work. Without his constant guidance and research directions, this research work could not be completed. His continuous support and encouragement has motivated me to remain streamlined in my research work.

I am thankful to my lab mates, **Apoorv Ambade** and **Krish Agrawal**, for their valuable support throughout my project. Their helpful feedback and suggestions greatly contributed to improving my work at various stages. I am also grateful to **Dr. Ranveer Singh**, HOD of Computer Science and Engineering for his help and support.

My sincere acknowledgement and respect to **Prof. Suhas Joshi** , Director, Indian Institute of Technology Indore for providing me the opportunity to explore my research capabilities at Indian Institute of Technology Indore.

I would like to express my heartfelt respect to my parents for their love, care and support they have provided to me throughout my life.

Finally, I am thankful to all who directly or indirectly contributed, helped and supported me.

Prajjval Dhiman

Abstract

Urban Air Mobility (UAM) promises transformative solutions for urban transportation by leveraging Personal Aerial Vehicles (PAVs) to reduce congestion, travel time, and environmental impact. Manual piloting in PAVs is discouraged due to the high traffic density of PAVs and their high operating speeds. Hence, PAVs should be equipped with an integrated Autonomous Navigation Control System (ANCS), which eliminates the need for the rider to be a skilled pilot. This paper presents an ANCS for PAVs, integrating a hybrid Convolutional Neural Network (CNN) and Gated Recurrent Unit (GRU) architecture to enable real-time obstacle avoidance and path planning. The system processes sequential RGB-D sensor inputs to predict 3D displacement vectors, ensuring safe manoeuvres around static and dynamic obstacles. A low-pass filtering technique smooths trajectories, enhancing passenger comfort and flight stability. Toward this end, an image dataset was created by setting up an end-to-end simulator framework. This dataset includes various categories of RGB, Depth Perspective, Infrared, etc., images labelled with positional or orientation data taken. This dataset is used to train deep learning models to figure out new coordinates for the vehicle to generate the route. The core of our approach utilizes deep learning to process the image data and predict an optimal navigation path by determining new positional coordinates for the vehicle. We present the dataset creation process and our proposed deep learning model architectures for realizing the autonomous navigation control system. The proposed model can safely drive the PAV across simulated surroundings, as shown by the results, which highlight the potential for real-world applications in urban air transportation. This research advances the safety and reliability of autonomous PAVs for scalable UAM deployment. Our experiments demonstrate the efficacy of the developed autonomous module. Evaluated in a high-fidelity Unreal Engine-AirSim simulation, the ANCS achieves a 96% intervention success rate and 100% navigation success across diverse urban scenarios, outperforming CNN-LSTM baselines in collision avoidance.

List of Publications

1. **P. Dhiman**, A. Ambade, K. Agrawal, and G. Banda. *Deep Learning-Based Autonomous Navigation for PAVs in Urban Airspaces via Synthetic Dataset Generation Framework*, (Accepted)

Contents

List of Publications iii

List of Figures vii

List of Tables ix

List of Abbreviations and Acronyms xi

1 Introduction 1

1.1 Contributions of this work 3

2 Literature Review 7

2.1 Preliminaries 11

3 Methodology 15

3.1 Dataset 15

3.2 Proposed Model Architecture 17

3.2.1 CNN for Spatial Feature Extraction 18

3.2.2 GRU for Temporal Sequence Modeling 19

3.2.3 Fully Connected Layers for Displacement Regression 20

3.3 Control System Integration 21

3.4 Post-Prediction Smoothing for Flight Stability 22

4 Experimentation and Result 25

4.1 Experimental Design 25

4.2 Software Stack 25

4.3 Virtual Environment 26
4.4 Experimental Results 27
4.4.1 Training Strategy and Hyperparameters: 27
4.4.2 Navigation Success Rate 27
4.4.3 Collision Rate 28
4.4.4 Dynamic Response to Obstacles 28
4.4.5 Intervention Success Rate 28
4.4.6 Environment Adaptability 29
4.4.7 Path Efficiency and Flight Smoothness 29
4.4.8 Model Comparison 30

5 Conclusion and Future Works **33**

Bibliography **39**

List of Figures

2.1 High-level Framework 12

3.1 Personal Aerial Vehicle (PAV) 16
3.2 Proposed PAV Control GRU Architecture 18
3.3 Overview of the Proposed ANCS Framework 21

List of Tables

2.1 Comparison of Proposed Framework with State-of-the-art Methods for Obstacle Avoidance and Navigation 9

3.1 Non-Collision Dataset Information 17

4.1 Impact of Smoothing on Trajectory Quality 30

4.2 Comparison of PAV Control Architectures 31

List of Abbreviations and Acronyms

PAV Personal aerial vehicle

ANCS Autonomous navigation and control system

SDD Simulation-driven development

UAV unmanned aerial vehicle

UAS Unmanned air system

CPS Cyber-Physical System

ATM Aerial traffic management

UAM Urban air mobility

QGC QGroundControl

PAVeDS Personal Aerial Vehicle Dataset

ROS Robot operating system

Chapter 1

Introduction

Urban transportation systems worldwide face escalating challenges, including crippling traffic congestion, excessive carbon emissions, and inefficient land use. Effective transportation infrastructures are necessary to provide market access and foster mobility [1]. However, a significant challenge these systems face is traffic congestion. The issue causes slower speeds, longer travel times, and a decrease in the general quality of life [2, 3]. According to the U.S. Department of Transportation, about 6.7 billion US gallons of gasoline go to waste annually because of traffic congestion. Minor occurrences, such as a crash or sudden braking of an automobile, can cause a chain reaction that eventually leads to permanent traffic slowdown over time [4]. Personal Aerial Vehicles provide an efficient means for urban transportation without all these problems.

Most contemporary transport methods are not environmentally friendly. Personal road transport faces numerous challenges. Urban Air Mobility (UAM) includes making such a concept come true with safe, efficient air transportation over urban areas using advanced aircraft like electric Vertical Take-Off and Landing. It aims to reduce ground traffic congestion and provide rapid, point-to-point travel for passengers and goods within cities [5, 6]. These systemic inefficiencies demand disruptive solutions—Personal Aerial Vehicles (PAVs) represent a transformative leap forward. PAVs are autonomous electric aircraft designed for urban air mobility (UAM), offering point-to-point aerial transit without runways. Unlike traditional drones or piloted he-

licopters, PAVs integrate AI-driven navigation, vertical takeoff/landing (eVTOL), and passenger-grade safety to revolutionize short-distance travel. However, enabling autonomy in dense urban environments—with dynamic obstacles (e.g., buildings, other PAVs) and strict safety requirements—remains a critical unsolved problem. UAM envisions integrating air taxis, cargo drones, and emergency services into city transportation networks [7]. The implementation of Personal Aerial Vehicles (PAVs) can mitigate these difficulties. PAVs will be able to change traveling as they allow for direct point-to-point or door-to-door flights, and cut fuel usage in densely populated areas, and meet today's expectations of time effectiveness. NASA has also gone further in supporting PAV technology by hosting challenges focused on encouraging innovation and cooperation in the design of PAVs [8]. The Federal Aviation Administration (FAA) intends to modernize forms of the Next Generation Air Transportation System (NextGen), which will be ready in 2025. This new transport system aims to reduce environmental impacts while providing faster and more convenient alternatives than traveling by road.

Over the past few decades, aerial transportation has been increasingly popular for long-distance travel, notably between cities and nations, with the purpose of transporting a greater number of passengers and big payloads [9].On the other side, the PAV concept is focused on travelling short distances or renting vehicles for one to four people. A PAV is an aircraft designed for passenger transport for the betterment of urban air mobility [10]. In comparison to traditional UAVs or normal aircraft, PAVs operate at decidedly slower speeds and altitudes, typically within 80 to 150 meters. This makes them ideal for short-range, low-altitude flight within urban environments.

Safety is the primary concern with PAVs, as they are classified as passenger-class vehicles intended to transport humans [11]. They must, therefore, be much more heavily regulated than UAVs, including many fail-safes and redundancy devices in their design. These regulations ensure that PAVs are safe, reliable, and can operate in crowded airspace without posing a risk to passengers or the general public [12, 13].

Removal of human control, hence reducing risks such as fatigue and misjudgment, is meant to improve safety. Preferable would be an autonomous piloting system capable

of making decisions in the manner of humans with much more accuracy [14]. Such a system would guide the vehicle from source to destination and would avoid obstacles by precise manoeuvring and real-time adjustments. Using advanced sensors and machine learning techniques, the autonomous agent would continuously sense its environment, predict dangers, and make optimal decisions. This approach ensures the maximum safety, completely ruling out human error while improving the reliability [15].

Cutting-edge deep learning methods, coupled with data from several sensors on the PAV, can be used to create a sophisticated autonomous piloting system [16, 17]. This system would analyze real-time sensor data from LiDAR, cameras, and IMUs to achieve a good perception of the surroundings. It would then produce the accurate control signals that will be relayed to PAV's onboard computer. With these signals, the PAV would navigate carefully by dynamically updating its position, orientation, and velocity. We call this API as ANCS.

The focus of this thesis is to develop an Autonomous Navigation Control System (ANCS) intended to navigate the PAV along its flight path. Innovative deep learning model architectures are employed to accurately avoid obstacles en route throughout the entire journey. These architectures utilize advanced deep learning methodologies fused with real-time sensor data, ensuring reliable decision-making during flight. A virtual simulation environment has been created with Unreal Engine [18] in conjunction with AirSim [19], facilitating comprehensive generation of dataset, assessment, and optimization of the ANCS across many simulated scenarios. This strategy seeks to improve the overall safety and efficacy of UAM systems.

1.1 Contributions of this work

Driven by the imperative for safe, scalable, and cost-effective development of autonomous PAVs in intricate urban airspaces and varying traffic patterns/densities, this thesis proposes a simulation-based, synthetic dataset-driven ML-based navigation module approach. The main contributions of this dissertation are:

- **Custom High-Fidelity Simulation Environment:** A vendor-agnostic PAV simulation environment comprising Unreal Engine and AirSim, which is instrumented with the Piloting engine "PX4" and ground control via "QGroundControl". This setup facilitates flexible recreation of diverse urban scenarios (varying building layouts, foliage, other PAV-traffic and weather) without real-world risk.

- **Synthetic Dataset Creation via Manual Flight Interventions:** By piloting the PAV through collision and near-miss manoeuvres in simulation, we collected a rich image-dataset/s (RGB, depth, infrared), which covers a range of rare and safety-critical events.

- **Hybrid CNN + GRU Navigation Model:** We designed a two-stage deep-learning network that fuses spatial features (five-layer CNN) with temporal context (stacked GRU), enabling accurate prediction, manoeuvring and real-time path planning.

- **Integrated Autonomous Navigation Control System (ANCS):** The CNN–GRU predictor is embedded into a live control loop—using depth thresholds for dynamic obstacle intervention and AirSim's command APIs to generate low-latency waypoint commands on the fly.

- **Lightweight Trajectory Smoothing for Passenger Comfort:** We introduce a low-pass filter on predicted coordinates to eliminate jerks, striking an optimal balance between responsiveness and flight smoothness without perceptible lag.

- **Comprehensive Simulation-Based Validation & Baseline Comparison:** Extensive experiments demonstrate 100% route completion and 4% collision rate across multiple urban maps and traffic scenarios. We did comparison with pure-CNN and CNN +LSTM variants to further image highlight the superior safety and reliability of our hybrid approach.

These contributions collectively establish a robust, safety-first framework for autonomous PAV navigation, paving the way for future work on real-world deployment

and sim-to-real transfer.

The rest of the thesis is organized as follows: Chapter 2 discusses related work relevant to our study; Chapter 3 details our proposed autonomous piloting agent; Chapter 4 presents the experimental evaluation of our approach; and finally, Chapter 5 provides a discussion of the findings and concludes the thesis.

Chapter 2

Literature Review

This section covers an extensive literature review that focuses on the most relevant contributions to the field of autonomous navigation, obstacle detection and avoidance, and machine learning in these innovative technologies. PAVs are an area of rapid research in development [10], but few specific studies have focused on these vehicles. Based on our analysis, there is a considerable deficiency in specific research work towards the development of autonomous PAVs, especially within cities, where a more dynamic and complex navigation problem exists. PAVs are closely related to UAVs, and thus comparing them helps highlight the unique challenges and requirements of UAM. In this section, we shall describe how our work covers all the gaps found in previous studies.

Prior research, like that conducted by Pearson *et al.* [20], used a multi-task regression-based learning (MTRL) approach for UAV navigation in forest environments. This approach primarily uses waypoint navigation with North, East, Down (NED) coordinates to navigate around stationary obstacles like trees. Although the MTRL approach is effective for forest scenarios, it is less suitable for the rapid decision-making required in metropolitan areas. Furthermore, previous work has been limited to confined areas, whereas our approach is adaptable to both confined and expansive spaces. By integrating deep learning techniques for dynamic control and constant obstacle detection, our system is specifically designed to excel in UAM.

We investigated the same concepts that guided this study and put them into ap-

plication, this time using deep learning approaches to address the problem of UAV navigation systems with emphasis on object detection and avoidance in real-time. While Mandloi *et al.* [21] managed to use CNNs properly for visual perception in forest areas, we have modified it for cases where the UAV is surrounded by complex urban structures, which require elevation changes. Besides this, the developed virtual environment built with the aid of Unreal Engine and AirSim served as a simulator which assisted in the assessment and the improvement of the pilot agent's performance under different flying circumstances. This allows our model to have direct access to real sensor information, which is quite important in those areas where GPS information cannot be obtained. In general, our work expands the scope of machine learning approaches by integrating advanced neural networks along with a hybrid integration of CNN and GRU that provides a solution for the specific problems of the urban air mobility systems.

Kouris *et al.*[22] demonstrate a self-supervised approach based on CNN to enable UAVs to navigate indoor environments with stationary obstacles autonomously. Their approach predominantly focuses on distance-to-collision predictions using a monocular camera, which excels in manoeuvring around stationary obstacles. However, it does not take into account dynamic factors such as moving objects and multi-UAV interactions. Their system is also designed for miniaturized and enclosed spaces, making it more difficult to scale and be effective in larger and more intricate spaces. These studies serve as a good basis for this work, except that this work tries to enhance the development of a flying agent for PAVs in urban traffic conditions in simulation.

In contrast to Kouris *et al.*[22] our design employs a hybrid model that combines CNN with GRU. This enables the PAV to navigate through, circumvent or even fly over static and dynamic as well as countless normal objects, enabling it to make rapid decisions in complex navigation. Consequently, it allows for rapid decision-making in complex navigation and path planning—capabilities that were not present in the previous model. Further, our approach gives rise to a virtual simulated universe, thereby providing complete experimentation in different cities, which enhances normalization and tolerance to autonomous flight in such areas.

Table 2.1: Comparison of Proposed Framework with State-of-the-art Methods for Obstacle Avoidance and Navigation

Article	**Obstacles Type**			**Method**		**ML Type**		**Destination Reaching**	**Global / Local Planner**	**Demonstration Method**	
	Static Object	**Other Moving PAVs**	**Multiple Obstacles**	**Prediction**	**Avoidance**	**Supervised**	**RL**			**Simulation**	**Demonstration Tool**
Pearson *et al.* [20]	✓	No	No	✓	✓	✓	No	No	Local	✓	AirSim + Unreal Engine
Mandloi *et al.* [21]	✓	No	No	No	No	✓	No	No	Global	✓	ROS-gazebo
Kouris *et al.* [22]	✓	No	✓	✓	✓	✓	No	No	Local	No	Testbed
Zhang *et al.* [23]	✓	No	✓	✓	No	No	✓	✓	Local	✓	Simulator not mentioned
Amer *et al.* [24]	✓	No	✓	No	✓	✓	No	No	Local	✓	AirSim + Unreal Engine
Proposed Method	✓	✓	✓	✓	✓	✓	No	✓	Local	✓	AirSim + Unreal Engine

Zhang *et al.* [23] proposes a strategy for adaptive collision avoidance in the case of several UAVs operating over urban terrain. However, this approach mainly overlooks the potential of deep reinforcement learning. This reliance on reinforcement learning can prolong the training process, and their method is predominantly aimed at collision avoidance between UAVs rather than effectively navigating through diverse and dynamic obstacles, including mobile entities. On the other hand, in our work, we apply a deep learning based method, which enables PAVs to operate in core urban environments, addressing both static and dynamic obstacles. By utilizing deep learning techniques, our model anticipates and mitigates various risks in urban environments, significantly reducing the lengthy training periods commonly associated with reinforcement learning. In addition to that, our approach ensures that PAVs are able to reach targets, ensuring collision-free motion, incorporating a more comprehensive resource than rerouting and increasing speed methods considered earlier, which only ensure collision-free motion without ensuring a trajectory planning approach.

Amer *et al.* [24] proposes an innovative technique for autonomous drone navigation

along predefined paths using onboard visual input; their model generates pitch, yaw, and roll commands based on fixed patterns of movement. While efficient for repetitive tasks such as package delivery, it primarily applies to stable environments with minor alterations.

Our work seeks to fill the gap brought about by the nature of UAM scenarios for PAVs that are highly dynamic and chaotic, as well as interact with both static and dynamic targets. Unlike the traditional approach of following certain routes, our CNN, LSTM and GRU-based system can flexibly adjust to subsequent environmental changes. This adaptability enhances the robustness of our approach, surpassing the performance of purpose-built systems that are designed for specific tasks. With regard to these gaps and advances, our approach proposes to expand the area of autonomous PAV navigation in more complex aerial urban scenarios.

Table 2.1 presents a comparative overview of various state-of-the-art methods for obstacle avoidance and navigation, highlighting the capabilities of each in handling different obstacle types, learning paradigms, and simulation environments. Most existing approaches, such as those by Pearson *et al.* [20] and Mandloi *et al.* [21] handle static obstacles well but are limited in managing dynamic entities like other PAVs or multiple concurrent obstacles. While supervised learning is commonly employed, reinforcement learning remains underutilized. Only Zhang *et al.* [23] explicitly uses RL, and few methods ensure complete destination reaching capabilities. Moreover, many works focus on local planning, with global planning seen only in Mandloi *et al.* [21] The proposed method stands out by supporting all obstacle types (static, dynamic, and multiple), integrating both prediction and avoidance modules, and ensuring destination reaching through a local planner. Additionally, the proposed framework is validated using the AirSim simulator integrated with Unreal Engine, providing a realistic and interactive environment for robust testing. This comprehensive feature set underscores the proposed method's superior adaptability and completeness compared to existing solutions. Our ANCS for PAVs enhances the capability to navigate from source to destination while avoiding obstacles. The ANCS employs novel machine learning based models to predict safe waypoints and generate manoeuvring commands,

effectively managing static and dynamic obstacles throughout the flight. Trained on a custom dataset tailored for PAV dynamics and rigorously tested in a virtual environment powered by Unreal Engine and AirSim, this method significantly improves the security and reliability of UAM systems.

In our work, we propose the PAVs' ANCS model with real-time capabilities, marking a significant advancement in autonomous flight technology, particularly for navigating complex urban environments.

2.1 Preliminaries

PAVs are autonomous aircraft, designed to be accessible and operable by a broad segment of the population [25]. They are seen as revolutionizing personal transportation because they reduce travel time substantially, increase the reach of mobility, particularly of cities and suburbs, and enhance general quality of life [26].

The essence of this thesis is to develop a piloting agent that will control the dynamics of the aircraft, making it completely autonomous throughout its journey by incorporating short-term path planning. This will utilize the aircraft's perception module and Flight Control System (FCS). This means that perception is critical to make sure the PAV can properly understand its environment and respond correctly. It makes autonomous navigation possible and aids in safe manoeuvring to avoid obstacles [27]. The FCS [28] ensures a stable flight because it balances altitude, pitch, roll, and yaw by sending commands from the piloting agent. We developed an internal end-to-end simulated framework to synthesize data set and show the functionality of ANCS module. This framework is explained in detail in work of Prasad *et al.* [29].

Figure 2.1 shows the architecture of the proposed pilot agent that integrates with the PAV. The perception module is made up of a camera that collects environmental data that is input into the ANCS. The Long-Term Planner (LTP) and the Short-Term Decision Maker (ST-DM) are the two salient components in the ANCS that interact with each other to produce navigation commands. The output of the ANCS, in the form of control signals such as pitch, yaw, and roll, is passed to the FCS. The

FCS modifies the rotors to provide stable, yet responsive flight dynamics. In this streamlined integration, the autonomous navigation and stability of the PAV can be ensured.

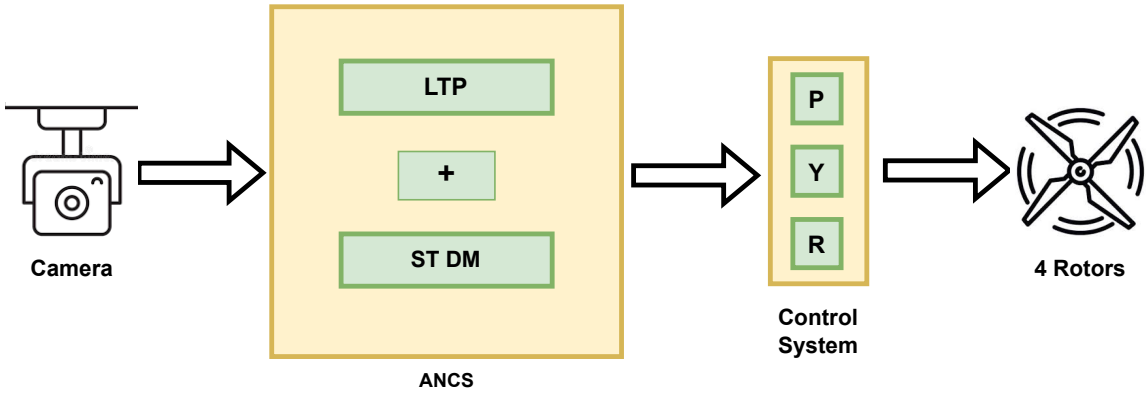


Figure 2.1: High-level Framework

Path planning [30, 31, 32] is generally divided into two main categories, long-term path planning (global planning) and short-term path planning (local planning).

Long-term path planning: It determines the overall trajectory from the starting point to the destination, focusing on selecting an efficient and feasible route across a larger area. This process considers known obstacles, no-fly zones, and restricted areas to ensure safe navigation.

Short-term path planning: It is responsible for real-time navigation and obstacle avoidance within the immediate vicinity of the PAV. It involves continuous, dynamic adjustments to circumvent unexpected obstacles that may appear along the designated route.

This work focuses on short-term path planning. Novel ML-based models are used to integrate this with the piloting agent. The perception module of PAV will capture images, while depth sensors measure distances to nearby obstacles, which will then be fed into this model for optimal navigation output. This output will be passed to the FCS to manoeuvre the aircraft appropriately, ensuring successful and safe navigation.

A **Convolutional Neural Network (CNN)** is an application-specific deep learning model for processing structured grid data, such as images [33]. In the case of PAVs, CNN is used to process visual inputs from the perception module, and therefore, the

aircraft is able to interpret its environment and make appropriate decisions for navigation. Extracting spatial hierarchies and features from the image data makes CNN an essential part in autonomy [34, 35].

While CNNs are effective in spatial feature extraction, autonomous navigation in dynamic environments also requires temporal understanding. To address this, **recurrent neural network (RNN)** architectures such as **Long Short-Term Memory (LSTM)** networks are leveraged for their ability to capture long-term dependencies in sequential data. In PAVs, LSTMs are especially beneficial in situations where decision-making hinges on context from previous events, such as following moving obstacles or interpreting flight history over time. By maintaining important data from previous time steps, LSTMs make navigation more stable and context-aware and hence are an important element in maintaining safety and responsiveness during autonomous flight [36].

Complementing the capabilities of LSTMs, **Gated Recurrent Units (GRUs)** offer a computationally efficient alternative for modelling temporal dependencies in sequential data. For PAVs, GRUs are utilized to process time-series data, such as sequences of sensor data or visual frames, enabling the system to learn motion patterns and predict changes in the environment [37]. Their comparatively less complex architecture than LSTMs allows for quicker training while retaining good performance in temporal awareness tasks such as trajectory prediction and short-term path planning.

Like any machine learning algorithm, CNNs, LSTMs and GRUs rely heavily on the quality and quantity of data used in training. Here, a custom-synthesised dataset is utilized, which has been especially designed for training the PAVs under multiple scenarios. The dataset contains images from the front camera of the PAV with corresponding geometric information from the perception module. This all-embracing data collection would enable the CNN to learn effectively in order to allow the PAV to navigate and react to the environment in real time.

Chapter 3

Methodology

In this section, we present the proposed methodology developed to address the challenges of urban transportation by leveraging Personal Aerial Vehicles (PAVs) to reduce congestion, travel time, and environmental impact. Our methodology for developing the Autonomous Navigation Control System (ANCS) integrates deep learning-based perception, real-time path planning, and flight control optimization within a high-fidelity simulation environment. Our methodology for developing the Autonomous Navigation Control System (ANCS) consists of four key components: (1) Dataset Collection and Augmentation, (2) Proposed Model Architecture, (3) Control System Integration, and (4) Post-Prediction Smoothing for Flight Stability.

In Unreal Engine, Figure 3.1 shows the replacement of AirSim's default drone model with the custom PAV model. The PAV components were combined into one mesh, and pivot points for the PAV body and propellers were fine-tuned to function in rotation and sensor placement. A static collision mesh was given to the final model, which allowed the simulation environment to realistically detect collisions, thus producing an appropriate PAV design suitable for the requirements set by the simulation.

3.1 Dataset

A comprehensive dataset is crucial for any machine-learning task, and this thesis employs a synthetic dataset designed solely for training PAVs. The simulation envi-



Figure 3.1: Personal Aerial Vehicle (PAV)

ronment was designed within Unreal Engine and AirSim, using which the dataset of metropolitan landscapes with obstacles such as skyscrapers, trees and other PAVs has been generated. The PAV was operated manually in this environment utilising PX4 SITL [38] flight controller software and QGroundControl (QGC) [39] as the ground control station, manoeuvring through complex scenarios including other PAVs functioning concurrently. This high-fidelity simulation, utilising Unreal Engine's real-time rendering, generated realistic scene simulations and enabled automatic annotation of sensor data.

The simulated environment enabled the gathering of rich sample data and sensor information, which is important for tasks such as obstacle detection and making decisions in real-time. The simulator enabled acquiring multi-sensor data that involves camera feeds, depth images, and LiDAR scans; this offers a complete dataset for training machine learning models. A synthetic dataset provides a scaleable and efficient framework for developing ANCS for PAVs.

The dataset collected for this study is collision-free, which means the PAV was able to pass over all structures and other PAVs during the experiment with no collision. It encompasses nearly 29,000 RGB high-resolution images, as demonstrated in Table 3.1, taken with a size of 768 by 432 pixels. Four different flights were used within this

Table 3.1: Non-Collision Dataset Information

Image Type	**Flights**	**Frames**	**Image Resolution**	**Size**
RGB	4	29,320	768px x 432px	17.2 GB

simulated urban environment; that means complete coverage of scenarios, as well as trajectory lines, would be covered. Table 3.1 provides key statistics about the Non-Collision Dataset used in the proposed modelpresents detailed information about the Non-Collision Dataset, which was collected to support tasks involving visual understanding in non-collision scenarios. The dataset consists of RGB images, meaning each frame contains full color information across red, green, and blue channels—essential for capturing rich visual details in the environment. Data was gathered over the course of 4 distinct flight sessions, which adds diversity and variability to the dataset, making it more robust for training and evaluation. In total, 29,320 image frames were recorded, offering a large sample size that can improve the generalization of machine learning models. Each image has a resolution of 768×432 pixels, providing a balance between image quality and computational efficiency. The complete dataset occupies 17.2 GB of storage, reflecting its high volume and quality. Together, these values indicate that the dataset is comprehensive and suitable for tasks such as object detection, navigation, or environmental analysis in drone or robotic applications. This dataset is also valuable because it has a plentiful visual context, offering a broad set of perspectives and situations for navigating complex, changing metropolitan environments.

3.2 Proposed Model Architecture

To enable real-time obstacle avoidance in dynamic urban environments, we propose a neural network combining convolutional and recurrent processing. The pro-

posed system focuses on short-term path planning to avoid obstacles and make safe manoeuvres. The main intention is to change the PAV's flight path dynamically to avoid static obstacles such as buildings, trees, and dynamic obstacles like other PAVs or moving objects. Using real-time information and fast decision-making techniques, the technology will surely ensure safe passage in a comprehensive urban environment. This thesis introduces a Deep learning-based ANCS for PAVs, designed to solve such problems wherein visual input is read from a PAV's front-mounted camera and transformed into coordinate positions.

The proposed architecture integrates a Convolutional Neural Network (CNN) with a Gated Recurrent Unit (GRU) to process sequences of RGB images and predict three-dimensional displacement vectors (d_x, d_y, d_z), as shown in Figure 3.2. This hybrid design ensures effective spatial feature extraction and temporal sequence modelling, resulting in accurate and reliable predictions. The input to the architecture is a

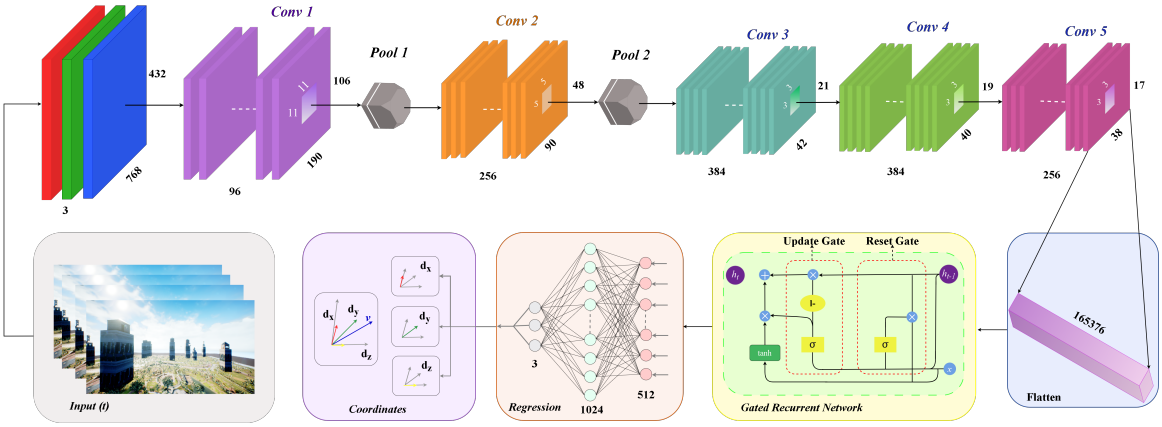


Figure 3.2: Proposed PAV Control GRU Architecture

sequence of t RGB images, each with dimensions $3 \times 432 \times 768$. Each frame is processed individually by the CNN. The detailed discussed is provided in Sections 3.2.1, 3.2.2, and 3.2.3.

3.2.1 CNN for Spatial Feature Extraction

The proposed CNN architecture serves as the feature extractor, processing each frame independently to capture spatial information. It consists of five convolutional

layers interleaved with nonlinear activation functions and max-pooling operations, which reduce dimensionality while preserving key spatial features. Each stream of the proposed CNN receives an input image of $3 \times 432 \times 768$, which denotes the number of channels, height, and width with a stride of 4, respectively. This captures large-scale spatial patterns, such as edges and textures. The first convolutional layer outputs a feature map of size $96 \times 106 \times 190$, followed by a max-pooling operation with a kernel size of 3×3 and a stride of 2, reducing the dimensions and retaining significant features. The second convolutional layer produces a feature map of size $256 \times 48 \times 90$, again followed by a similar max-pooling operation. This process continues through the third $384 \times 21 \times 42$, fourth $384 \times 19 \times 40$, and fifth $256 \times 17 \times 37$ convolutional layers, each followed by max-pooling layers with the same kernel size and stride.

The final convolutional layer produces a feature map of dimensions $256 \times 17 \times 38$, which is flattened into a vector of size $165,376$ for each frame. This vector encodes the spatial features essential for temporal modelling. By employing a combination of convolutional and pooling operations, the CNN effectively extracts spatial patterns, such as edges and textures, that contribute to robust feature representation for downstream processing.

3.2.2 GRU for Temporal Sequence Modeling

The Gated Recurrent Unit (GRU) is utilized to model the temporal dynamics present in the sequence of input frames. Feature vectors extracted by the CNN for each frame are fed sequentially into the GRU, which is configured to process and encode temporal dependencies effectively. Each input to the GRU is a flattened feature vector of size 165,376 representing the spatial features of a single frame. The GRU employs a hidden state of dimensionality 512, allowing it to capture and retain complex temporal patterns across the sequence. To further enhance its ability to model long-term dependencies, the architecture includes two stacked GRU layers, enabling deeper temporal representation.

As the GRU processes the feature vectors sequentially, it generates a hidden state at each time step. For the task of regression, only the hidden state from the final time

step is retained. This final hidden state encapsulates the temporal relationships and dependencies across the entire sequence of frames, providing a comprehensive temporal representation for downstream processing. By leveraging the GRU's ability to manage sequential data effectively, the architecture ensures robust temporal sequence modelling for accurate displacement prediction.

3.2.3 Fully Connected Layers for Displacement Regression

The architecture of the fully connected layers plays a crucial role in transforming the hidden state output of the GRU into a format suitable for predicting displacement in three-dimensional space. Following the output from the GRU, the first fully connected layer (FC1) comprises 1024 neurons, utilizing a Rectified Linear Unit (ReLU) activation function. This configuration enables the model to effectively map the 512-dimensional output from the GRU to a significantly higher-dimensional feature space. The application of the ReLU function not only facilitates non-linearity but also enhances the learning capacity of the model by allowing it to better capture complex patterns in the data.

Following FC1, the second fully connected layer (FC2) is structured with three neurons corresponding to the displacement components (d_x, d_y, d_z). This final layer is responsible for processing the transformed features from FC1 and producing outputs representing the displacement in three-dimensional space. The direct link between the output neurons and the displacement components allows for a straightforward interpretation of the model's predictions, ensuring that the three-dimensional displacement vector is generated to align with the desired regression output.

By combining fully connected layers with the GRU, this architecture efficiently leverages both spatial and temporal features. The integration of FC1 and FC2 within a CNN-GRU framework culminates in a robust and versatile model that addresses challenges in dynamic environments. Such a design not only enhances predictive accuracy but also ensures that the model can adapt to complex scenarios, providing a solid foundation for displacement regression tasks.

3.3 Control System Integration

The proposed method implements a CNN-GRU-based hybrid model to predict the coordinates of a PAV when an obstacle is detected in its path. Once trained, the model can forecast the positional coordinates d_x, d_y and d_z in the spatial plane (X, Y, Z) based on future visual inputs from the front-facing camera. These predicted coordinates are updated in real-time as the PAV navigates its environment. As illustrated in Figure 3.3, the control flow of the PAV begins with the vehicle moving to its initial position, where the source and destination coordinates are provided as input. The control system of the PAV receives these coordinates and converts them into control signals that adjust the motors and flight actuators, thereby regulating fundamental flight parameters, such as altitude, direction, speed, and rotation, essential for precise manoeuvres in urban aerial environments. The design procedure of the PAV

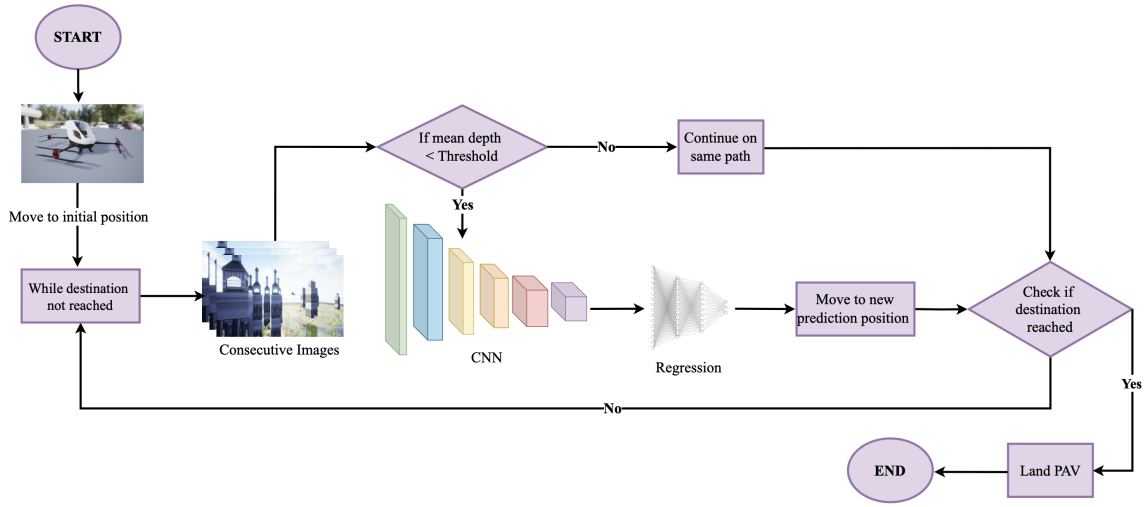


Figure 3.3: Overview of the Proposed ANCS Framework

Autonomous Navigation Control System is summarised in Algorithm 3.1. The depth camera captures depth images that help calculate the mean depth of the captured image. This mean depth is compared to a set threshold value. If the mean depth is lower than this threshold, it signals that an obstacle is nearby. In this case, the system activates the proposed CNN-GRU-based model, prompting the PAV to move to a new predicted position that is free of obstacles. After reaching this safe location,

the PAV recalculates its path to the destination, considering the obstacle it detected, and adjusts its route as needed. On the other hand, if the mean depth is greater than the threshold, it indicates a clear path ahead, allowing the PAV to continue following its planned route without any changes. The integration of RGB and depth images into the system enhances obstacle detection and navigation capabilities. The depth images provide information about the relative distances of surrounding objects. Continuous checks of the depth values ensure that the PAV maintains a safe distance from potential obstacles. This approach guarantees effective navigation in complex urban environments by integrating real-time obstacle detection and continuous path recalculation, ultimately facilitating the PAV's autonomous manoeuvring.

The system depends on RGB images, depth images, and PAV control commands that are provided by AirSim. The RGB image enables the CNN model to understand how visual features of the surroundings should be interpreted, while depth images will provide information in regard to the relative distances of different objects from the PAV. The following section discusses an experimental setup along with a description of configurations and parameters adopted in the investigation, followed by the analysis of results obtained under different simulated conditions.

3.4 Post-Prediction Smoothing for Flight Stability

Aircraft stability is an important requirement of any passenger aerial vehicle design. For a flight to be comfortable and safe, the manoeuvres the aircraft performs must be smooth and free of any jerks or oscillations. Sudden velocity or direction changes not only impair passenger comfort but also threaten structural integrity and navigation accuracy. To mitigate this, we include a post-prediction smoothing method that polishes the raw output of the model, creating smoother and more consistent flight trajectories. It is a critical step in ensuring the reliability of the autonomous piloting system, especially in high-population urban airspace.

To achieve this, we use a *low-pass filtering* method that acts directly on the predicted positional coordinates—namely, the X, Y, and Z values produced by the col-

Algorithm 3.1 PAV Autonomous Navigation Control System

client: AirSim client connection
depth_img: Depth image from camera
uav_size: Size parameters of UAV
pred_pos: Predicted position [x, y, z]
procedure InterventionCounter(client, depth_img, uav_size, pred_pos)
    $current_pos \leftarrow$ client.getPosition()
    $\delta_x \leftarrow (current_pos.x - pred_pos[0])$
    $\delta_y \leftarrow (current_pos.y - pred_pos[1])$
    $\delta_x \leftarrow max(min(\delta_x, 20), -20)$
    $\delta_y \leftarrow max(min(\delta_y, 20), -20)$
    $p_x \leftarrow \delta_x + (current_pos.x)$
    $p_y \leftarrow \delta_y + (current_pos.y)$
    // Depth-based intervention
    $mean_depth \leftarrow mean(depth_img)$
    if $mean_depth > intervention_threshold$ **then**
        // Continue to final destination
        moveToPosition(FINAL_POINT_X,FINAL_POINT_Y,FINAL_POINT_Z)
    else
        // Execute ML guided movement
        moveToPosition($p_x, p_y, pred_pos[2]$)
    end if//Main navigation loop

    while distance_to_goal > threshold **do**
        capture_image()
        process_depth_data()
        predict_next_position()
        InterventionCounter(parameters)
        update_current_position()
    end while
    // Landing sequence
    move_to_landing_position()
    execute_landing()
end procedure

lision avoidance model. Instead of instructing the PAV to move immediately to the newly predicted coordinate, the low-pass filter mixes the predicted position with the previously achieved smoothed position, thus minimizing sudden changes in motion. This filter performs a *weighted average* of the new estimate and the latest smoothed estimate given by the equation:

$$S_t = \alpha \cdot P_t + (1 - \alpha) \cdot S_{t-1} \tag{3.1}$$

Here, S_t is the *smoothed coordinate* at the current time step, P_t is the *model-predicted coordinate*, S_{t-1} is the *previous smoothed coordinate*, and $\alpha \in (0, 1)$ is the *smoothing factor*. A low value of α produces a smoother trajectory by giving more importance to past motion, whereas a high value makes the vehicle respond faster to new predictions but with the penalty of higher jerkiness. By this filtering technique, we add a layer of *temporal continuity* to the movement commands, producing smoother trajectory execution that more closely matches the desired behaviour of a passenger-class aerial vehicle. This is a computationally light method and therefore very well-suited for *real-time deployment* in our AirSim–Unreal Engine simulation. When applied along each axis—X, Y, and Z—the filtered position coordinates (S_t^x, S_t^y, S_t^z) are calculated independently according to the formula:

$$S_t^i = \alpha \cdot P_t^i + (1 - \alpha) \cdot S_{t-1}^i \quad \text{for} \quad i \in \{x, y, z\} \tag{3.2}$$

This filtering not only reduces noise and abrupt spikes in the model's output but also makes the resulting motion closely resemble the *smooth, stable, and predictable trajectories* of human-piloted commercial aircraft. Consequently, it greatly improves both the *functional reliability* and *passenger comfort* of the autonomous navigation system, particularly in complex and dynamic urban airspaces.

Chapter 4

Experimentation and Result

4.1 Experimental Design

The experimental setup was implemented on a desktop system equipped with an Intel Core i9-14900F CPU, 32 GB of RAM, and an NVIDIA RTX 4070 Ti Super GPU. The model was developed using Python and the PyTorch [40] deep learning framework. To simulate real-world scenarios, real-time RGB image data was captured from a virtual front-facing camera within the PAV (Personal Aerial Vehicle) simulation environment, developed using AirSim. With the aid of sensor inputs in determining the navigation decisions, the proposed model produces safe velocity commands for the PAV. With Unreal Engine, along with AirSim, made for a strong simulation environment, and the real-world dynamics of the PAV were replicated within urban settings; data collection and testing of models took place.

4.2 Software Stack

Testing PAVs through simulation has multiple advantages against real-world testing, first of all for safety concerns, cost and scalability, among others. Simulations enable PAVs to be tested against complex environments such as varying altitudes and weather conditions with dense obstructions without fear of physical damage to the equipment nor endangering human personnel on site. Customization possibilities al-

low pinpoint control over variables, which is critical in fine-tuning navigation as well as obstacle avoidance systems. This, in turn, makes simulations an attractive, efficient path forward towards UAM advancements.

For the simulation environment, it uses Unreal Engine, which is a powerful 3D development platform that has a great capability to create an immersive virtual environment. AirSim is built on Unreal Engine and is used as a versatile simulator for different types of autonomous vehicles, including drones, cars, and so on. It gives real physics simulations, which include aerodynamics for drones, enabling it to accurately model how such systems operate in real conditions.

AirSim supports a range of sensors that are commonly used in autonomous systems; such sensors include RGB, depth, and segmentation cameras for sensing the environment, plus Lidar for high-accuracy 3D mapping and obstacle detection. Furthermore, it also includes simulated GPS for improved navigation precision and inertial measurement units for accurate motion tracking; this makes the simulation in total more realistic and efficient. Users can interact with the simulator through Python APIs, which enable programmable control of vehicles and the retrieval of real-time data, together with logging capabilities for sensor data, vehicle states, and environmental conditions, which facilitates machine learning model training and analysis of agent behavior.

Datasets: Since the original dataset from a previous study [29] was relatively small, we augmented it using the same method ourselves to improve model performance and generalization. The process of dataset creation is explained in detail in Section 3.1. The dataset that we used is made available in two parts through: URL1 https://bit.ly/4jW0ST2 and URL2 https://bit.ly/4kpX9wY.

4.3 Virtual Environment

The virtual environment is a 1 km x 1 km map with several buildings and foliage, simulating a real urban environment. The heights of the buildings vary from 10 to 100 meters, allowing for a diverse skyline of various navigation challenges. PAV is

expected to fly at an altitude of 80 meters above ground level with a speed of 30 meters per second. The buildings have been arranged in a manner to cover a wide range of complex, realistic scenarios. This would allow testing for the navigation and obstacle avoidance capabilities of the PAV.

4.4 Experimental Results

To assess the effectiveness of the proposed ANCS, we conducted a series of simulated flight experiments in the Unreal Engine–based AirSim environment [41]. A custom dataset of vehicle-centric visual observations was used to train each model variant separately on grayscale and RGB inputs. We compared our method against two state-of-the-art baselines: a purely convolutional neural network (CNN) and a CNN augmented with a LSTM layer. All experiments comprised 4 flights across three urban scenarios: highly dense city, sparse cityscape, and single-obstacle layout.

4.4.1 Training Strategy and Hyperparameters:

The models were trained using the Adam optimizer with an initial learning rate of 1×10^{-4}, which was dynamically reduced using a learning rate scheduler upon plateauing. The batch size was set to 64 sequences.The validation loss with a patience of 10 epochs, and capping training at a maximum of 100 epochs. The empirical depth threshold for obstacle intervention was formally set to 15 meters, providing an optimal balance that allowed sufficient braking time at 30 m/s without triggering premature path deviations. For post-prediction smoothing, the temporal filter parameter α was explicitly set to 0.2, successfully weighting historical trajectory data to filter high-frequency spatial jitter while remaining highly responsive.

4.4.2 Navigation Success Rate

The navigation success rate is defined as the percentage of flights in which the PAV reached its intended destination without collision. Across all 4 trials, our ANCS

yielded a 100% success rate, demonstrating reliable end-to-end route completion under varying urban densities.

4.4.3 Collision Rate

The collision rate R_{col} is defined as the ratio of the number of actual collisions N_{col} to the total number of obstacle contacts N_{contact}:

$$R_{\text{col}} = \frac{N_{\text{col}}}{N_{\text{contact}}} \tag{4.1}$$

We distinguish between (1) *contacts*—instances where the PAV entered the collision threshold zone and (2) *actual collisions*. Out of 75 total contact events across all flights, only 3 resulted in collisions, yielding an *overall collision rate* of 4.0%. This low collision frequency highlights the model's high-precision obstacle detection and avoidance capabilities.

4.4.4 Dynamic Response to Obstacles

To evaluate responsiveness, we recorded the number of in-flight interventions—moments when the controller issued a corrective waypoint due to obstacle proximity. Qualitatively, the PAV consistently adjusted its trajectory in real time, maintaining safe clearance in both static and dynamic obstacle scenarios.

4.4.5 Intervention Success Rate

The intervention success rate R_{int} measures the fraction of in-flight interventions that did *not* result in a collision. Let N_{int} be the total number of interventions and N_{fail} the number of those interventions that still led to a collision. Then

$$R_{\text{int}} = 1 - \frac{N_{\text{fail}}}{N_{\text{int}}} = \frac{N_{\text{int}} - N_{\text{fail}}}{N_{\text{int}}} \tag{4.2}$$

Out of the 75 intervention situations when the PAV found an obstruction and needed a route modification, it collided only three times. This produces an *intervention success*

rate of around 96.0%, therefore illustrating the great dependability of the model in actively preventing accidents during in-flight interventions.

4.4.6 Environment Adaptability

Performance metrics were computed in each urban scenario:

- **Highly Dense City:** Complex intersections of towering structures and narrow corridors. The ANCS avoided all potential collisions with minimal path deviation.

- **Sparse Cityscape:** Open layouts with fewer obstacles. The PAV achieved the shortest path lengths and lowest intervention counts in this setting.

- **Single-Obstacle Layout:** Simplified scenario with an isolated barrier. Path corrections were minimal and highly efficient.

4.4.7 Path Efficiency and Flight Smoothness

We further analyzed trajectory quality using two metrics: path efficiency (ratio between straight-line and actual flown distance) and smoothness. As shown in Table 4.1, the proposed smoothing module has a substantial impact on trajectory quality across multiple evaluation metrics. The trajectory smoothness score improves markedly from 0.42 to 0.89 (on a normalized 0–1 scale), indicating a significantly more fluid and continuous motion path. This enhancement is further reflected in the reduction of harsh turns, dropping from 12 to just 3 per 100 meters, which contributes to more stable and safer navigation. The stability index, another measure of flight consistency, rises from 0.35 to 0.91, showing a dramatic increase in trajectory reliability.

While the addition of the smoothing module introduces a modest increase in execution time (from 12 ms to 14 ms per frame) and model complexity (from 5.2M to 7.8M estimated FLOPs), these computational costs are offset by the overall system improvement. The overall performance score, which aggregates multiple quality and

efficiency factors on a scale of 0–100, nearly doubles—from 46 to 87—demonstrating a favorable trade-off between accuracy and resource usage. These results confirm that the proposed smoothing technique significantly enhances trajectory quality while maintaining computational feasibility.

Table 4.1: Impact of Smoothing on Trajectory Quality

Criteria	**Without Smoothing**	**With Smoothing (Proposed)**
Trajectory Smoothness Score (0–1)	0.42	0.89
Number of Harsh Turns (per 100m)	12	3
Stability Index (0–1)	0.35	0.91
Execution Time (ms/frame, simulated)	12	14
Model Complexity (FLOPs, estimated)	5.2M	7.8M
Overall Performance Score (0–100, sim)	46	87

4.4.8 Model Comparison

Table 4.2 presents a comparative analysis of three distinct models employed for PAV Control, evaluating their performance based on seven key metrics: Average Flight Time (in seconds), Average Collision Count, Average Distance (in meters), Average Intervention Counter, Mean Absolute Error (MAE), Training Time (in hours), and Validation Loss. The optimal values for each metric are highlighted in bold. The Mean Absolute Error (MAE), which quantifies the average magnitude of the errors in a set of predictions, is given by

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \tag{4.3}$$

where y_i represents the actual values, $\hat{y}_i$ represents the predicted values, and n is the number of predictions.

The training time refers to the duration taken for the model to train on the dataset and is a critical factor in evaluating the efficiency of the model.

The validation loss is the Mean Squared Error (MSE) loss computed on the validation set of the dataset, and is given by:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{4.4}$$

where n is the total number of samples in the validation set, y_i represents the actual values, and $\hat{y}_i$ represents the predicted values for each sample. Each of these

Table 4.2: Comparison of PAV Control Architectures

Models	Avg. Flight Time (s)	Avg. Collision Count	Avg. Distance (m)	Avg. Intervention Count	Mean Absolute Error	Training Time (hrs)	MSE
PAV Control CNN	113.09	0.79	3485.90	9.50	**0.056**	**2.95**	**0.0177**
PAV Control (CNN + LSTM)	117.34	0.33	3680.12	13.25	0.401	58.72	0.4445
Proposed Method	**109.11**	**0.04**	**3324.87**	**18.75**	0.400	85.31	0.4427

metrics provides significant insights into the efficiency and performance of the models identified as CNN), LSTM, and GRU. Table 4.2 presents a comparative evaluation of three control architectures for Personal Aerial Vehicle (PAV) navigation: PAV Control CNN, PAV Control (CNN + LSTM), and the Proposed Method. The PAV Control CNN model shows strong overall performance with a high average flight time of 113.09 seconds and a substantial average distance of 3485.90 meters. It also achieves the lowest mean absolute error (0.056), indicating superior predictive accuracy, along with the shortest training time (2.95 hours) and lowest validation loss (0.0177), making it efficient and generalizable. The CNN + LSTM model, while achieving the longest flight time (117.34 s) and maximum distance traveled (3680.12 m), suffers from a higher intervention count (13.25) and poor prediction accuracy (MAE of 0.401), coupled with long training time (58.72 hrs) and high validation loss (0.4445). On the other hand, the Proposed Method excels in safety, recording the lowest collision count (0.04), which is crucial for real-world deployment. However, it has the highest intervention count (18.75) and longest training time (85.31 hrs), with predictive performance similar to the LSTM model. Overall, while the CNN-only model offers the best balance of accuracy, efficiency, and generalization, the proposed approach stands out for its remarkable safety, making it highly suitable for critical navigation tasks where collision

avoidance is paramount. Although our proposed CNN + GRU smoothing approach does not yield the lowest mean absolute error, falling slightly behind the pure CNN baseline—the choice of the average collision count as the primary evaluation metric reflects the safety-critical nature of PAV operations. In urban air mobility, even minimal deviations from the nominal trajectory can lead to passenger risk and potential harm. Our method achieves the lowest average collision count (0.04 per flight), significantly outperforming both the CNN + LSTM variant (0.32) and the pure CNN (0.79). By prioritizing collision avoidance over marginal improvements in prediction accuracy, we ensure that passenger safety remains the foremost design criterion for autonomous navigation in congested urban environments.

These findings confirm that the system can run effective and safe urban navigation throughout a spectrum of simulated cityscapes. Based on the performance data, our PAV system seems fit for UAM uses, therefore laying a basis for dependable autonomous flying in actual urban settings.

Chapter 5

Conclusion and Future Works

This study presented an Autonomous Navigation Control System (ANCS) for personal aerial vehicles (PAVs), addressing key challenges in the emerging domain of urban air mobility (UAM). Our proposed hybrid deep learning architecture achieved a 96% intervention success rate for obstacle avoidance while maintaining an exceptionally low collision rate of 0.04 per flight, establishing a benchmark for real-time, collision-free autonomous navigation.

The system demonstrated real-time processing capability at 14 ms per frame, incorporating a trajectory smoothing mechanism via an innovative low-pass filtering technique. This approach contributed to safer, smoother, and more comfortable flight experiences within densely populated and obstacle-rich urban environments. Our methodology was validated through extensive simulations using both collision and non-collision datasets in the AirSim/Unreal Engine simulation platform, confirming the robustness and practical feasibility of our system. Overall, the work represents a significant advancement toward achieving scalable and safe autonomous urban air transportation.

The proposed ANCS contributes meaningfully to the development of autonomous PAVs by integrating real-time decision-making for collision avoidance, thereby minimizing or potentially eliminating the need for human piloting. This system enhances flight reliability and operational safety, which are critical for future smart city applications.

At the core of our solution is a novel CNN-GRU-based model that processes two consecutive front-facing camera images along with the corresponding spatial coordinates (X, Y, Z) captured at each timestep. This fusion of spatial and temporal data allows the system to understand the immediate environment and dynamic scene changes effectively. By analyzing both the sequential visual frames and positional context, our dual-stream architecture delivers enhanced perception, making the system resilient to the complex and rapidly changing nature of urban airspaces.

While the short-term reactive decision-making module demonstrated promising performance in real-time navigation and obstacle avoidance, future work will focus on extending the ANCS to support long-term route planning. We aim to integrate a GPS-based path planning module that calculates optimal source-to-destination routes, thereby enabling a hybrid navigation system that combines both short-term reactivity and long-term strategic planning for efficient and safe PAV journeys.

Another critical future direction is the creation of an extensive dataset representing diverse and adverse environmental conditions, including rain, fog, snow, and sandstorms. Training and evaluating the ANCS under these conditions will test the system's robustness and adaptability in real-world scenarios where atmospheric visibility and sensor reliability are compromised. The inclusion of such scenarios will enable the system to generalize better and perform reliably under unpredictable and challenging operational environments. To further improve robustness and scalability, multi-sensor fusion techniques—combining inputs from LiDAR, radar, IMU, and vision—can be explored to enhance situational awareness, particularly under low-visibility or GPS-denied conditions. Additionally, hierarchical planning systems that incorporate dynamic obstacle prediction and real-time re-planning will be investigated to increase the flexibility and intelligence of navigation strategies in dynamic environments. Moreover, energy efficiency optimization through intelligent path planning, adaptive speed control, and potential integration of hybrid propulsion systems could significantly extend operational range and reduce energy consumption. Regulatory compliance, safety assurance, and standardized testing protocols will also be essential focus areas as we move toward real-world deployment.

Bibliography

[1] A. Koźlak, "The role of the transport system in stimulating economic and social development," vol. 72, pp. 19–33, 2017.

[2] A. Choudhary, S. Gokhale, P. Kumar, C. Pradhan, and S. Sahu, "Urban traffic congestion: its causes-consequences-mitigation," *Research Journal of Chemistry and Environment*, 2022.

[3] L. Jie, "An analysis of the socio-economic impact of traffic congestion," *Journal of Beijing Jiaotong University*, 2012.

[4] A. Garcia-Castro and A. Monzón, "Measuring the effects of traffic congestion on fuel consumption," 2015.

[5] D. Gillis, M. Petri, A. Pratelli, I. Šemanjski, and S. Semanjski, "Urban air mobility: A state of art analysis," pp. 411–425, 2021.

[6] A. Straubinger, R. Rothfeld, M. Shamiyeh, K.-D. Büchter, J. Kaiser, and K. Plötner, "An overview of current research and developments in urban air mobility – setting the scene for uam introduction," *Journal of Air Transport Management*, vol. 87, p. 101852, 2020.

[7] A. P. Cohen, S. Shaheen, and E. Farrar, "Urban air mobility: History, ecosystem, market potential, and challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, pp. 6074–6087, 2021.

[8] M. Moore, "Nasa personal air transportation technologies," 2006.

[9] R. Rothfeld, M. Fu, M. Balać, and C. Antoniou, "Potential urban air mobility travel time savings: An exploratory analysis of munich, paris, and san francisco," *Sustainability*, 2021.

[10] Y. Liu, M. Kreimeier, E. Stumpf, Y. Zhou, and H. Liu, "Overview of recent endeavors on personal aerial vehicles: A focus on the us and europe led research activities," *Progress in Aerospace Sciences*, vol. 91, pp. 53–66, 2017.

[11] R. Clothier and R. Walker, "Determination and evaluation of uav safety objectives," 2006.

[12] C. Stöcker, R. Bennett, F. Nex, M. Gerke, and J. Zevenbergen, "Review of the current state of uav regulations," *Remote. Sens.*, vol. 9, p. 459, 2017.

[13] A. Cracknell, "Uavs: regulations and law enforcement," *International Journal of Remote Sensing*, vol. 38, pp. 3054 – 3067, 2017.

[14] P. Narayan and S. Dogramadzi, "Enabling autonomous flight capabilities onboard commercial aircraft to improve safety," *2013 IEEE/AIAA 32nd Digital Avionics Systems Conference (DASC)*, pp. 3E4–1–3E4–10, 2013.

[15] A. Dietrich, "Unsettled topics in the general aviation autonomy landscape," 2022.

[16] S. Rezwan and W. Choi, "Artificial intelligence approaches for uav navigation: Recent advances and future challenges," *IEEE Access*, vol. PP, pp. 1–1, 2022.

[17] V. J. Hodge, R. Hawkins, and R. Alexander, "Deep reinforcement learning for drone navigation using sensor data," *Neural Computing and Applications*, vol. 33, pp. 2015 – 2033, 2020.

[18] A. Sanders, "An introduction to unreal engine 4," 2016.

[19] C. Ma, Y. Zhou, and Z. Li, "A new simulation environment based on airsim, ros, and px4 for quadcopter aircrafts," *2020 6th International Conference on Control, Automation and Robotics (ICCAR)*, pp. 486–490, 2020.

[20] B. G. Maciel-Pearson, S. Akçay, A. Atapour-Abarghouei, C. Holder, and T. P. Breckon, "Multi-task regression-based learning for autonomous unmanned aerial vehicle flight control within unstructured outdoor environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4116–4123, 2019.

[21] Y. S. Mandloi and Y. Inada, "Machine learning approach for drone perception and control," in *Engineering Applications of Neural Networks: 20th International Conference, EANN 2019, Xersonisos, Crete, Greece, May 24-26, 2019, Proceedings 20*. Springer, 2019, pp. 424–431.

[22] A. Kouris and C.-S. Bouganis, "Learning to fly by myself: A self-supervised cnn-based approach for autonomous navigation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[23] J. Zhang, H. Zhang, J. Zhou, M. Hua, G. Zhong, and H. Liu, "Adaptive collision avoidance for multiple uavs in urban environments," *Drones*, vol. 7, no. 8, p. 491, 2023.

[24] K. Amer, M. Samy, M. Shaker, and M. ElHelw, "Deep convolutional neural network based autonomous drone navigation," in *Thirteenth International Conference on Machine Vision*, vol. 11605. SPIE, 2021, pp. 16–24.

[25] H. Bülthoff, "Flying air taxis: A solution to the traffic problem in malta?" 2018.

[26] R. J. Lee and I. Sener, "Transportation planning and quality of life: Where do they intersect?" *Transport policy*, vol. 48, pp. 146–155, 2016.

[27] F. Liu, Z. Lu, and X. Lin, "Vision-based environmental perception for autonomous driving," *ArXiv*, vol. abs/2212.11453, 2022.

[28] D. McLean, "Automatic flight control systems," *Measurement and Control*, vol. 36, pp. 172 – 175, 1990.

[29] D. Prasad, S. T. Adhithiyan, K. Thanmai, and G. Banda, "Paveds: A synthetic dataset for developing autonomous personal aerial vehicles," *IEEE Access*, vol. 11, pp. 101 556–101 566, 2023.

[30] S. Aggarwal and N. Kumar, "Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges," *Comput. Commun.*, vol. 149, pp. 270–299, 2020.

[31] G. Gugan and A. Haque, "Path planning for autonomous drones: Challenges and future directions," *Drones*, 2023.

[32] J. Luo, Y. Tian, and Z. Wang, "Research on unmanned aerial vehicle path planning," *Drones*, 2024.

[33] H. Gao, B. Cheng, J. Wang, K. Li, J. Zhao, and D. Li, "Object classification using cnn-based fusion of vision and lidar in autonomous vehicle environment," *IEEE Transactions on Industrial Informatics*, vol. 14, pp. 4224–4231, 2018.

[34] V. Stojnić, V. Risojevic, M. Mustra, V. Jovanović, J. Filipi, N. Kezic, and Z. Babic, "A method for detection of small moving objects in uav videos," *Remote. Sens.*, vol. 13, p. 653, 2021.

[35] A. A. dos Santos, J. M. Junior, M. S. Araújo, D. R. D. Martini, E. Tetila, H. L. Siqueira, C. Aoki, A. Eltner, E. Matsubara, H. Pistori, R. Feitosa, V. Liesenberg, and W. Gonçalves, "Assessment of cnn-based methods for individual tree detection on images captured by rgb cameras attached to uavs," *Sensors (Basel, Switzerland)*, vol. 19, 2019.

[36] H. Wang, B. Lu, J. Li, T. Liu, Y. Xing, C. Lv, D. Cao, J. Li, J. Zhang, and E. Hashemi, "Risk assessment and mitigation in local path planning for autonomous vehicles with lstm based predictive model," *IEEE Transactions on Automation Science and Engineering*, vol. 19, pp. 2738–2749, 2022.

[37] A. Benterki, V. Judalet, C. Maaoui, and M. Boukhnifer, "Long-term prediction of vehicle trajectory using recurrent neural networks," *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1, pp. 3817–3822, 2019.

[38] PX4, "Px4 - open source autopilot," 2025, accessed: 2025-05-14. [Online]. Available: https://px4.io/

[39] QGroundControl, "Qgroundcontrol user guide," 2025, accessed: 2025-05-14. [Online]. Available: https://docs.qgroundcontrol.com/master/en/qgc-user-guide/index.html

[40] PyTorch, "Pytorch - deep learning framework," 2025, accessed: 2025-05-14. [Online]. Available: https://pytorch.org/

[41] Epic Games, "Unreal engine," https://www.unrealengine.com/en-US, 2025, accessed: 2025-05-08.