Development of a Disturbance Observer for Wind Estimation by Multirotor Drone using Machine Learning

by

Steven Zimmerman

BASc, The University of British Columbia, 2020

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

The Faculty of Graduate and Postdoctoral Studies

(Mechanical Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

May 2022

© Steven Zimmerman 2022

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Development of a Disturbance Observer for Wind Estimation by Multirotor Drone using Machine Learning

submitted by Steven Zimmerman in partial fulfillment of the requirements for

the degree of Master of Applied Science

in Mechanical Engineering

Examining Committee:

Steven Rogak, Professor, Mechanical Engineering, UBC
Supervisor

Ryozo Nagamune, Professor, Mechanical Engineering, UBC
Supervisory Committee Member

Bhushan Gopaluni, Professor, Chemical and Biological Engineering, UBC
Additional Examiner

Sheldon Green, Professor, Mechanical Engineering, UBC
Additional Examiner

Abstract

Multirotor drones are an ideal platform for a range of environmental studies, but wind measurement by anemometer payload suffers from the downwash of the propellers. This work presents a machine learning (ML) based disturbance observer, which implicitly estimates the wind based on the drone state without requiring a dedicated anemometer. Experimental data is collected by flying an instrumented quadrotor in proximity to two reference anemometers. Four ML models are developed: a long-short-term-memory (LSTM) neural network, an artificial-neural-network, a gated-recurrent-unit (GRU) neural network, and a Gaussian-process-regression. These models are trained with variations in their inputs, considering the addition of drag estimations by rigid-body equations of motion and control inputs, both of which improve performance. Two key processing methods are developed: Data augmentation by global coordinate frame rotation enforces rotational invariance and grid-based data reduction removes data imbalance. The best performing GRU achieves 0.48 m/s root-mean-square-error on unseen complete flight data, although the LSTM performs similarly. This approaches the experimental limits of performance, as reference anemometers cannot be exactly co-located with multirotors in flight. The difference between both spatially offset anemometers accounts for 73% of the GRU estimation error. These results are useful for emissions measurement, gas source localization and other applications.

Lay Summary

Drones are versatile and ideal for measuring wind. However, the measurement quality suffers when doing so directly with a dedicated wind sensor due to the airflow from the propellers. To overcome this, an implicit method is developed that computes what the wind must be based on how the drone is moving in response to the wind. Machine learning is used to train a model that does this, based on data obtained flying a drone close to a reference anemometer wind sensor. Special considerations are made to ensure that these models approximate physics closely, utilizing known physics and rotational properties of the drone. The final model is shown to estimate wind well, which can be used for environmental studies such as emissions measurement and gas source localization.

Preface

This work was conducted as a part of the UBC Aerosol Laboratory, supervised by Dr. Steven Rogak. Two chapters of the thesis are based on one published work, and one manuscript in review.

A version of Chapter 5 has been accepted for publication. This manuscript was produced with the collaboration of the co-authors Miayan Yeremi, Dr. Ryozo Nagamune, and Dr. Steven Rogak. Miayan Yeremi provided guidance on the instrumented drone system, and most significantly helped to design the custom motor speed sensors. Both Dr. Steven Rogak and Dr. Ryozo Nagamune provided input in the form of manuscript editing and supervision. It is important to acknowledge contributions by Dr. Dominic Liao-McPherson in the form of guidance to compare parametric models with non-parametric, and in manuscript editing. The primary ideas and findings were obtained by the author, Steven Zimmerman.

A version of Chapter 6 has been submitted for review. This work largely identifies drawbacks of the previous manuscript and improves upon them. The co-authors, Dr. Steven Rogak and Dr. Ryozo Nagamune provided guidance by supervision and manuscript editing. The ideas and results were obtained by the author, Steven Zimmerman.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to acknowledge the contributions of my supervisor, Dr. Steven Rogak in supporting and guiding me for the past 3 years in both my undergraduate degree and graduate studies. He has provided valuable guidance both personally and professionally, and has helped shape this research. I will look back on this period of my life fondly primarily due to his high quality instruction, collaboration and guidance.

I would also like to acknowledge contributions made by Dr. Ryozo Nagamune, who has provided key guidance and commentary on a regular basis and when producing manuscripts for journal article submission.

I would like to acknowledge fellow student Miayan Yeremi, for long conversations and discussions around this project particularly very early on.

Finally, I would acknowledge my fellow students, friends and family who have supported me continually for the duration of this work.

Chapter 1

Introduction

The need to measure a spatially and temporally varying wind field is present in a number of applications, such as environmental monitoring or wind farming applications. This is commonly achieved by static anemometers or fixed wing aircraft, but multirotor drones are ideal for their high controllability, versatility and positional accuracy. The main issue with using multirotors in this way is that direct measurement of wind by anemometer payloads are corrupted by the downwash of the propellers. The primary method of overcoming this that has been established in literature is an implicit disturbance observer, that can estimate the wind based on the multirotor state without a dedicated sensor. This section explains these ideas in detail, starting with the motivating applications of local wind measurement in Section 1.1, discussing existing methods in Section 1.2, summarizing the previous literature related to wind measurement by multirotor in Section 1.3, and describing the contributions of this work in Section 1.4.

1.1 Motivation

On a local scale, wind prediction is particularly difficult in the lower atmosphere where the Earth's surface, temperature inversions and turbulence contribute to modelling errors [1]. This leads to the need for experimental wind measurement approaches, which are useful for emissions quantification, gas source localization, wind farming, and aeronautical controls engineering. Emissions quantification describes the combination of gas concentration and wind measurement to infer an emissions rate from a known source, such as flares or vents. Gases such as methane have been identified as critical for climate change mitigation due to their high global warming potential, requiring quantification tools on a range of spatial and temporal scales [2]. This can be achieved by a mobile robotics platform moving through a plume, sampling wind speed and gas concentration. Shah [3] presents a near-field Gaussian plume inversion method for flux quantification, where methane concentration is measured by a multirotor drone and the wind measured by a static anemometer with a spatial uniformity assumption. Scheller [4] presented a similar study quantifying the surface methane flux of an arctic landscape. Sun [5] presented a fixed wing drone based eddy-covariance (EC) measurement system, which uses a similar principle to measure carbon dioxide and other surface fluxes. Shaw [6] presents a review of these methods focusing on methane quantification, highlighting the main challenges of measuring the wind at the location of gas concentration sampling, and producing lightweight, sensitive methane sensors suitable for drone deployment. Both Shah and Scheller used gas analyzers on the ground with sampling ports connected to a multirotor, limiting the effective distance of sampling and increasing the response time. A slightly different application of combining gas concentration and wind measurement on a mobile robotics platform is gas source localization, which attempts to locate and quantify unknown gas sources in an environment. Burques [7] presents a nano multirotor equipped with lightweight gas sensors, suitable for indoor environment gas source localization. Neumann [8] demonstrated a combination of implicit wind measurement and gas concentration measurement for outdoor environments. Ercolani [9] extends these results to 3-axis gas source localization with a micro drone measuring volatile organic compounds. Generally these methods

use biology-inspired airflow or concentration gradient-based methods to automatically locate a gas source requiring real-time wind estimation.

Knowledge of the local wind field around buildings or complex terrains can be highly valuable for flow mapping or wind farming applications. In these cases, the complexity of the earths surface or structures interacting with the wind makes modelling by computational fluid dynamics difficult, requiring experimental validation [10, 11]. Marino proposed using a swarm of small drones around the surface of a building to provide multiple point measurements of the flow field in real-time, to guide structural load calculations, validate simulation and inform the design and placement of urban wind turbines. Wildman [10] presented mapping of the wind field by a fixed wing drone for large wind farming applications, assessing the turbulence, magnitude, direction and variability for wind turbine placement and design. In general, the efficiency and power output of a wind turbine depends on wind speed distribution over the area of the blades [12, 13], making knowledge of spatial wind distributions valuable.

In control systems engineering, local wind estimation is critical for disturbance observer based design [14]. Such drones typically have low inertia, and are highly susceptible to atmospheric turbulence and disturbance effects. This can lead to motor or control actuator saturation and depleted battery life/flight time, which is particularly an issue for mission critical applications. Knowledge of the local wind field naturally leads to improved control algorithms, either by optimal control action for minimized control inputs, or by advanced path planning that takes advantage of wind direction and magnitude for mission execution. Autonomous energy harvesting or soaring requires an accurate method of wind estimation in high flight efficiency applications [15].

1.2 Wind Measurement Methods

Despite these valuable applications of wind measurement, there are a number of drawbacks to existing methods. The difficulty lies in the spatial and temporal variability of wind fields, and the ability to locate a wind measurement method at the position and time of interest. Either cup anemometers, sonic anemometers or more expensive light detection and ranging (LIDAR) anemometers are commonly statically located on booms or existing structures to measure the wind field at one site, requiring a spatial uniformity assumption to generalize their results [16]. LIDAR anemometers can measure a wind field over spatial dimensions, however they are limited to a fixed range from their location. Fixed wing drones offer another approach, as wind measurement packages can be mounted without significant downwash effects. This approach is generally suitable for large spatial scales, due to the fast movement and low positional control of the platform. Martin [17] demonstrated using such a system to measure temperature, humidity and wind profiles up to a height of 1500 m, to measure the turbulence structure of the atmospheric boundary layer. Despite these benefits, fixed wing drones are less suitable for local wind measurements due to their minimum stall speed. This minimum stall speed increases the sample rate requirements of sensors, in order to achieve the same spatial resolution. Their inability to hover makes them less suitable for complex flights in and around structures. They require some form of runway for take-off and landing, decreasing their versatility. Multirotors offer an attractive platform for measuring wind in applications where local but spatially varying wind is of interest. They are able to hover with high positional accuracy, and move as desired through a local outdoor or indoor environment. Improved real-time-kinematic (RTK) measurement systems allow a drone to hover at centimeter level accuracy. They allow for highly versatile payloads, with varying capacity based on the size of the drone and number of propellers. They can easily accept lightweight gas concentration sensors to be used for gas source localization or

emissions rate measurement. Despite these advantages, their primary drawback is the complexity in measuring wind speed directly as the propellers create significant downwash over the body of the drone corrupting direct anemometer measurement. In the literature, two primary methods have been proposed to overcome this. The direct method of wind measurement attaches anemometers on vertical or horizontal booms far away from the propellers to reduce interference, generally requiring correction factors to be incorporated and calibrated based on the orientation. The indirect method involves implementation of a disturbance observer, which can implicitly estimate the wind from the drone state alone not requiring any dedicated anemometers in the payload. This method is generally preferred for the benefits to payload and flight time but it is an ongoing research topic.

1.3 Previous Work

A detailed study of the relevant literature related to measuring wind by multirotor drones is provided, to understand the significance of this work related to challenges and improvement areas of existing literature. This section is divided into two parts: the first of which considers the "direct" method of measuring wind involving an anemometers as multirotor payload. The second section details the "indirect" method, using a disturbance observer that estimates wind from the drone state alone.

1.3.1 Direct Method

Attaching an anemometer directly to a drone for wind measurement is much more intuitive, but comes with its own set of challenges. Prudden [18] provided a valuable study using flow mapping around a multirotor in a wind tunnel, showing the disturbed regions close to the drone and developing correction functions for propeller induced flow. Multiple authors [19, 20, 21] present demonstrations of measuring wind with anemometers attached on vertical booms above a multirotor. Typically these systems require large payload capacity (>1kg) drones, which are frequently hexacopters or octacopters. They are generally able to provide wind estimations to an accuracy of around 0.4 m/s root-mean-square-error (RMSE) [19], which requires correction functions developed either by detailed computational fluid dynamics studies or wind tunnel testing. The disadvantage of these works is the requirement for large payload drones, with a penalty in flight time and controllability due to the weight of the boom and anemometer. This is the primary motivation for developing indirect wind measurement methods.

1.3.2 Indirect Method

Neumann [22, 23] presented the pioneering study on this work developing a simplified physics-based wind disturbance observer. This work used a static assumption, to relate the orientation of the drone as measured by the pitch, roll and yaw angles directly to the wind magnitude and direction. To account for the unknown drag coefficients that vary as the drone changes orientation, the bank angle to wind magnitude relationship was calibrated using wind tunnel testing. This showed a polynomial relationship with a minor deadzone. Overall this work achieved a wind magnitude estimation root-mean-square-error (RMSE) of 1.09 m/s[1] when hovering and 0.87 m/s RMSE[1] when flying. The direction performance metrics are 29.12° RMSE[1] and 31.66°

[1]These performance metrics are reported for no moving average filters applied. The full performance metrics with averaging are reported in [23].

RMSE[1] for hovering and flying respectively. This multirotor wind estimation was applied to a gas sensitive microdrone, utilizing "sniffing" algorithms for gas source localization [8].

A number of other papers have provided disturbance observers for estimation of wind using physics based models of varying complexity. Wang [24] and Tomic [25] both present similar studies, developing wind estimations for static hovering drones based on indoor fan testing, or wind tunnel testing. Palomaki [26] used a similar static physics model, directly relating the drone bank angle to the wind but calibrated the average drag coefficient from flight data rather than wind tunnel testing. This work considered hover flight only, achieving wind speed accuracies of 0.33-0.88 m/s RMSE and direction accuracies of 12-21° RMSE depending on the flight trial. Calibrating the model based on real-flight data represents an attractive option for model development by eliminating the need for wind tunnel testing. Gonzalez-Rocha [27] develops 5 linearized models for a multirotor, each corresponding to a different steady state ascension rate of 0 to 2 m/s, which is used for vertical wind profiling studies. Simma [28] presents work that involves more detailed motor modelling provided by static thrust tests and assumes a constant drag coefficient, which is validated with a drone in hover achieving 0.49-0.56 m/s RMSE.

In recent years, machine learning (ML) has been applied to this problem. Allison [29] presented a simulation study of a multirotor that applies a long-short-term-memory neural-network (LSTM NN) to estimate wind. In a controlled simulation environment, they produce wind using a Dryden wind turbulence model and simulated the drones response with known and controlled drag coefficient characteristics. Their model predicted wind directly from the pitch and roll angles, where the ground speed and yaw angles were held constantly at zero. While this is a greatly simplified model of a multirotor drone, this demonstrated the ability of ML models to approximate their dynamics. Wang [30] applied a K-nearest-neighbours (KNN) algorithm to this problem, using an experimental drone that hovers indoors near a controlled wind source. Crowe [31] builds upon both of these works, applying both KNN and LSTM algorithms to an experimental outdoors study. Their drone hovered approximately seven meters away from an ultrasonic anemometer used as reference, and predicted wind based on the pitch, roll and acceleration in three dimensions. Yaw was neglected as an input in this study, and held constant. In general, there is a lack of literature results regarding a model for wind estimation by multirotor drones in generalized flight conditions. Most commonly, literature results neglect the dynamic information of the drone or limit their validation to hover flight only without considering changing yaw angles.

1.4 Contributions of this Work

This work considers the application of ML to the problem of implicit wind estimation using a multirotor drone. It addresses the performance assessment for generalized flight conditions, and improves upon published model accuracy. An experimental system is developed, where a multirotor drone is instrumented to record the full drone state and control inputs. This drone is flown in proximity to an anemometer, in the presence of a turbulent wind field. Performance improvements are pursued through multiple efforts. Full multirotor drone state and control inputs are considered, where different frameworks of varying complexity for applying ML are proposed and compared. Data augmentation by rotation and reduction is applied, which help the models more closely approximate the true physical relationship. Finally, performance is quantified in a generalized and robust way by considering the effects of changing flight conditions.

1.5 Organization of Thesis

This thesis is organized as follows:

Chapter 2 provides background information on multirotor drone modelling and provides a derivation of the theoretical relationship between the drone state and the wind. It also provides a basic review of ML models and practices which will be used in this thesis.

Chapter 3 explains the experimental system developed, the custom motor speed sensors built, the flight tests performed, and the signal processing required before applying ML models.

Chapter 4 considers application of a purely physics-based modelling approach to this problem, showing its benefits and drawbacks. This leads to the primary motivation for applying ML to this problem.

Chapter 5 contains a version of a journal article submitted for publication, which is currently under review. This chapter contains the details of model training and comparison of two out of the four ML model frameworks described in Chapter 2.

Chapter 6 addresses a key drawback of Chapter 5, which is inconsistent performance metrics and develops data rotation and reduction techniques to improve upon this. This chapter is also a version of a journal article submitted for publication and is currently under review.

Chapter 7 unifies these results, completing the analysis required to effectively compare all ML models considered in all four of the frameworks proposed. A discussion of the performance considers changes with varying flight phases, changes for increased averaging, and comparing performance to the expected limits determined by the experimental method.

Chapter 8 concludes this thesis by providing a discussion of future work.

Chapter 2

Background

2.1 Problem Definition

The scope of this work considers development of a disturbance observer for estimating wind by multirotor drone state. A disturbance observer is defined as a model in software that is able to estimate what the disturbance acting on a system must be based on the control inputs, system state and existing knowledge of the system [32]. Intuitively this is understood by considering that the system state is a deterministic function of inputs, disturbances, and their histories. Therefore, an inverse plant model that sufficiently approximates the real system can estimate disturbances. For a multirotor, this concept is shown in Figure 2.1. The notations used in this figure representing the drone state and control inputs are consistent with Figure 2.2.

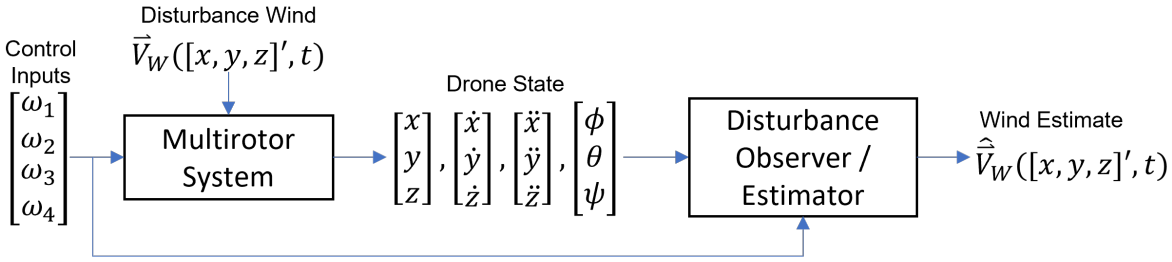


Figure 2.1: Concept of a disturbance observer implemented for a multirotor system.

Here, the disturbance wind is defined as a vector comprising the average free-stream wind magnitude acting on a multirotor over the drone's area. This is expressed as a function of the drone's position and time, as it is a spatially and temporally varying wind field. The spatial scale of the wind estimate will be fundamentally limited by the size of the drone, which can range from centimeters to meters depending on the drone used. To understand what measured drone state variables and inputs are relevant for this problem, we consider studying the equations of motion for a rigid body multirotor.

2.2 Multirotor Physics Modelling

An investigation of the theoretical relationship between the drone state and the wind acting on a multirotor drone is useful for understanding the application of ML to this problem. While this is discussed in each of Chapters 5 and 6 with simplifications or omissions, the relationship is explained here in full detail. The non-linear equations of motion for a multirotor drone [33] are

$$m \begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = \begin{bmatrix}0\\ 0\\ -mg\end{bmatrix} + R \begin{bmatrix}0\\ 0\\ F\end{bmatrix} - \begin{bmatrix}D_x\\ D_y\\ D_z\end{bmatrix}, \tag{2.1}$$

which are expressed in the coordinate frame shown by Figure 2.2. Here m [kg] is the mass of the drone, g [m/s^2] is the acceleration due to gravity, and F [N] is the total thrust force from all the rotors. $\vec{D} = [D_x, D_y, D_z]'$ [N] is the drag acting on the drone due to airspeed. R is the 3-dimensional transformation matrix from the body frame of reference to the global frame of reference, which is defined as

$$R = \begin{bmatrix} C_\psi C_\theta & C_\psi S_\theta S_\phi - S_\psi C_\phi & C_\psi S_\theta C_\phi + S_\psi S_\phi \\ S_\psi C_\theta & S_\psi S_\theta S_\phi + C_\psi C_\phi & S_\psi S_\theta S_\phi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix}, \tag{2.2}$$

where C_x and S_x denote $\cos(x)$ and $\sin(x)$ respectively. The vector $[\phi, \theta, \psi]'$ [rad] represents the pitch, roll and yaw angles of the drone with respect to the global coordinate frame. Appropriately applying R and inverting the equations of motion yields a closed form method of estimating the drag in the global frame of reference acting on the multirotor by the drone state alone:

$$\begin{bmatrix} \hat{D_x} \\ \hat{D_y} \\ \hat{D_z} \end{bmatrix} = \begin{bmatrix} (C_\psi S_\theta C_\phi + S_\psi S_\phi)F - \ddot{x}m \\ (S_\psi S_\theta C_\phi - C_\psi S_\phi)F - \ddot{y}m \\ -gm + C_\theta C_\phi F - \ddot{z}m \end{bmatrix}. \tag{2.3}$$

The drag estimate can then be related to the airspeed moving over the drone by a drag coefficient relationship:

$$\begin{bmatrix} D_x \\ D_y \\ D_z \end{bmatrix} = \frac{1}{2}\rho|\vec{V_a}|S(\cdot)C_D(\cdot) \begin{bmatrix} V_{Ax} \\ V_{Ay} \\ V_{Az} \end{bmatrix}, \tag{2.4}$$

where ρ [kg/m^3] is the air density, $S(\cdot)C_D(\cdot)$ [m^2] is the drag area product, and the airspeed is $\vec{V_A} = [V_{Ax}, V_{Ay}, V_{Az}]'$ [m/s]. This equation assumes that the drag force acts co-linearly with the airspeed, where no net lift is produced. The drag area product is expressed as some unknown function, because this plant parameter is not trivial to identify. It varies as a function of the bank angle of the drone, which is a complex relationship in three dimensions of rotation. This relationship is left as some unknown function f to be approximated by ML:

$$\begin{bmatrix} \hat{V_{Ax}} \\ \hat{V_{Ay}} \\ \hat{V_{Az}} \end{bmatrix} = f\left(\begin{bmatrix} \hat{D_x} \\ \hat{D_y} \\ \hat{D_z} \end{bmatrix}, \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \right). \tag{2.5}$$

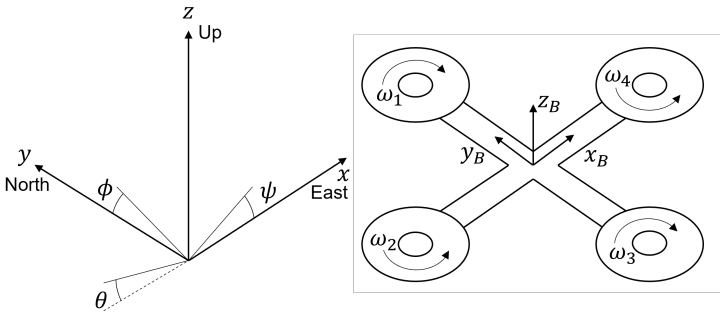


Figure 2.2: Coordinate frame used for notation. Quadrotor shown but derivation is for generic multirotor. Figure reused in Chapter 5.

When using this data-driven approach, no assumption about the direction of the drag force is imposed. This means that ML models can appropriately learn the drag and lift characteristics of the drone as it changes orientation. Finally, wind is estimated by vector subtraction, which is also commonly known as the wind triangle:

$$\hat{\vec{V}}_W = \begin{bmatrix} \hat{V_{W}}_x \\ \hat{V_{W}}_y \\ \hat{V_{W}}_z \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} - \begin{bmatrix} \hat{V_{A}}_x \\ \hat{V_{A}}_y \\ \hat{V_{A}}_z \end{bmatrix}. \tag{2.6}$$

In principle, this relationship applies to all three dimensions and has the capacity to estimate vertical components of wind. In practice however we will neglect the vertical component of the wind and apply these equations to the horizontal plane alone as this is primarily of interest. This entire relationship is well summarized by Figure 2.3.

Thus, there is an expected deterministic relationship between the drone state and the wind speed, but the difficulty in determining this relationship is the unknown drag coefficient of the drone at an arbitrary orientation. This is the primary motivation for considering ML approaches, which seek to approximate this relationship by sufficient training examples.

2.2.1 Rotor speed considerations

As will be explained later in Chapter 3, the total thrust of the motors will be estimated by four custom motor speed sensors which were developed for this work. As this is a custom sensor which is not widely accessible, estimations are considered with and without the rotor speeds available as an input which is later discussed in Chapters 5 and 7. When rotor speeds are available, the total thrust force is estimated as

$$F = k(\omega_1^2 + \omega_2^2 + \cdots + \omega_r^2), \tag{2.7}$$

where k [Ns^2] is the thrust constant of all propellers, and $\vec{\omega} = [\omega_1, \omega_2, ...\omega_r]'$ [rad/s] is the set of rotor speeds. This is shown for a generic multirotor with r rotors but in our case this is four. When no rotor information is available, the total thrust is estimated as

$$F = \frac{(g + \ddot{z})m}{C_\theta C_\phi}, \tag{2.8}$$

which comes from the assumption that the airspeed moving over the drone in the vertical direction is negligible. Thus, if this work were to be extended to include the vertical wind components then thrust estimation by control input would be required.

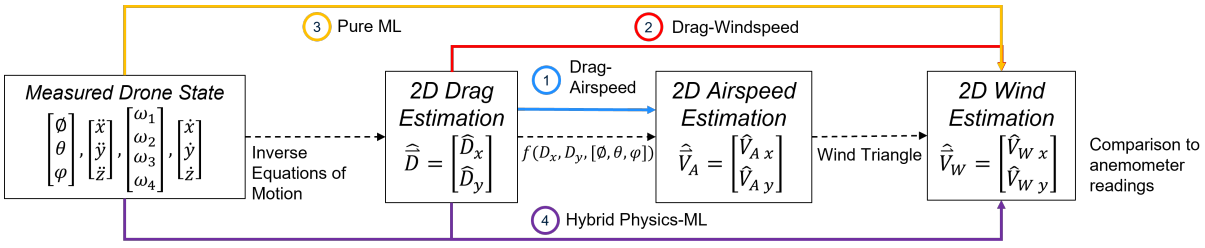


Figure 2.3: Entire relationship from drone state to wind estimation, showing the different ways ML can be applied to this problem.

2.2.2 Application of Machine Learning

ML is useful for this problem, as there is a difficult to identify and unknown relationship between the drag estimates by equation of motion and the airspeed moving over the drone in flight. Figure 2.3 shows the entire derived relationship, where $f(D_x, D_y, [\phi, \theta, \psi]')$ expresses the unknown drag-airspeed relationship. Four different scopes are proposed in which ML could be applied to this problem, ordered in increasing complexity. Each of these cases considers an increasing scope of approximation, which in principle allows the models to correct for any systematic errors in the system or account for physical effects not captured by physical modelling. However, this comes with the increased risk that the models overfit, and are not able to approximate the true physical relationship well. The first of these approaches limits the application to the uncertain drag-airspeed relationship. It uses the known equations of motion and vector math as pre-processing and post-processing steps. In this case the equation approximated by the ML models is

$$\hat{\vec{V}}_A\left(\begin{bmatrix}x\\y\\z\end{bmatrix}, t\right) = \begin{bmatrix}\hat{V_{Ax}}([x,y,z]',t)\\ \hat{V_{Ay}}([x,y,z]',t)\end{bmatrix} = g_{ML1}\left(\begin{bmatrix}\hat{D}_x(t)\\ \hat{D}_y(t)\\ \hat{D}_z(t)\end{bmatrix}, \begin{bmatrix}\phi(t)\\ \theta(t)\\ \psi(t)\end{bmatrix}\right). \tag{2.9}$$

The drag-windspeed approach increases the scope of ML, approximating the wind-triangle portion of the system as well. This opens up the models to approximate or correct for systematic errors in the measurement of the drones global velocity. The equation to be approximated is

$$\hat{\vec{V}}_W\left(\begin{bmatrix}x\\y\\z\end{bmatrix}, t\right) = \begin{bmatrix}\hat{V_{Wx}}([x,y,z]',t)\\ \hat{V_{Wy}}([x,y,z]',t)\end{bmatrix} = g_{ML2}\left(\begin{bmatrix}\hat{D}_x(t)\\ \hat{D}_y(t)\\ \hat{D}_z(t)\end{bmatrix}, \begin{bmatrix}\phi(t)\\ \theta(t)\\ \psi(t)\end{bmatrix}, \begin{bmatrix}\dot{x}(t)\\ \dot{y}(t)\\ \dot{z}(t)\end{bmatrix}\right). \tag{2.10}$$

The pure-ML approach treats the entire process as one function to be approximated, which is expressed as

$$\begin{aligned}\hat{\vec{V}}_W\left(\begin{bmatrix}x\\y\\z\end{bmatrix}, t\right) &= \begin{bmatrix}\hat{V_{Wx}}([x,y,z]',t)\\ \hat{V_{Wy}}([x,y,z]',t)\end{bmatrix} = \\ g_{ML3}&\left(\begin{bmatrix}\phi(t)\\ \theta(t)\\ \psi(t)\end{bmatrix}, \begin{bmatrix}\dot{x}(t)\\ \dot{y}(t)\\ \dot{z}(t)\end{bmatrix}, \begin{bmatrix}\ddot{x}(t)\\ \ddot{y}(t)\\ \ddot{z}(t)\end{bmatrix}, \begin{bmatrix}\omega_1(t)\\ \omega_2(t)\\ \omega_3(t)\\ \omega_4(t)\end{bmatrix}\right).\end{aligned} \tag{2.11}$$

The hybrid physics-ML approach is a variation on the pure-ML approach, where the well-known equations of motion are used to pre-specify the drag estimates which are used as an additional input. The ML models are then allowed to select whether or not to use the drag estimates in the training procedure. The function to be approximated for this fourth and final

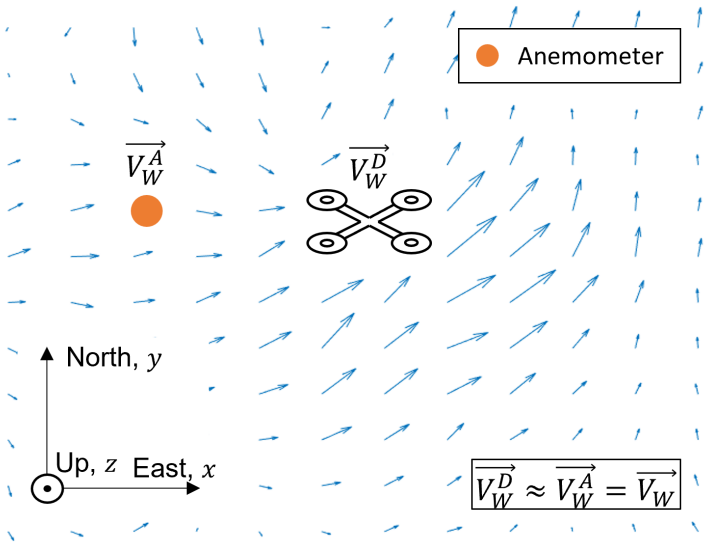


Figure 2.4: Formal definition of spatial uniformity assumption

scope is

$$\hat{\vec{V}_W}\left(\begin{bmatrix}x\\y\\z\end{bmatrix}, t\right) = \begin{bmatrix}\hat{V_{W}}_x([x,y,z]',t)\\ \hat{V_{W}}_y([x,y,z]',t)\end{bmatrix} = g_{ML4}\left(\begin{bmatrix}\hat{D}_x(t)\\ \hat{D}_y(t)\\ \hat{D}_z(t)\end{bmatrix}, \begin{bmatrix}\phi(t)\\ \theta(t)\\ \psi(t)\end{bmatrix}, \begin{bmatrix}\dot{x}(t)\\ \dot{y}(t)\\ \dot{z}(t)\end{bmatrix}, \begin{bmatrix}\ddot{x}(t)\\ \ddot{y}(t)\\ \ddot{z}(t)\end{bmatrix}, \begin{bmatrix}\omega_1(t)\\ \omega_2(t)\\ \omega_3(t)\\ \omega_4(t)\end{bmatrix}\right). \tag{2.12}$$

All of these wind estimations are expressed as a function of the drones position, $[x, y, z]'$ [m], and time t [s]. This means that the drone will be able to spatially and temporally map a wind field, based on its trajectory. To train models using a supervised learning approach, known examples with flight data as input and the wind speed as output are required. It is possible to measure the wind accurately with stationary anemometers, but the difficulty is the inability to co-locate an anemometer with the drone in flight. When flying too close to an anemometer, downwash issues prevail. When flying too far away from an anemometer, spatial variation of the wind prevails. For training purposes, a multirotor drone will be instrumented and flown in proximity to an anemometer with a spatial uniformity assumption to obtain training examples. Formally this is explained by Figure 2.4, where the anemometer readings are assumed to be equal to the wind at the drones location so long as the drone is kept sufficiently close, around five to ten meters away. This figure shows an arbitrary wind field, but in reality this is unknown. The validity of the spatial uniformity assumption for training will be checked by the performance achieved by the models, in Section 7.2.

2.3 Review of Machine Learning

A review of relevant ML models is provided here to properly understand their application. This section explains the the concept of artifical-neural-networks (ANN), Gaussian-process-regressions (GPR), recursive-neural-networks (RNN), long-short-term-memory (LSTM) neural

networks, and gated-recurrent-unit (GRU) neural networks. The RNN section serves as a precursor to the discussion of LSTM and GRU models, as these are both forms of RNN models. A discussion is provided surrounding the theory of learning, selecting training, validation and testing data and cross-validation, concepts that are all applied in this work.

2.3.1 Fundamentals of Supervised Learning

ML is a generalized modelling approach that can be used to approximate unknown non-linear functions. In the case of physical systems, there is some expected relationship between an input variable (X) and output variable (Y) which can be either continuous or categorical. Both (X) and (Y) may be univariate or multivariate. Supervised learning attempts to fit this relationship between using sufficient examples, pre-determined model structures and training procedures. For example, the model structure could be an artificial neural network with a specified number of layers and hidden states. This determines the set of possible internal parameters which are tuned in order to minimize the difference between Y and $\hat{Y}$ using a gradient-based learning method.

The data used to train these models is typically referred to as a "training" data set. Many forms of models are able to quickly achieve very low training errors without the performance generalizing well. This means that when moving from the training data set to unseen data, the performance may decrease significantly. This is commonly referred to as "over-fitting", where the ML model learns the peculiarities of the training data set rather than the underlying true function. "Under-fitting" on the other hand is when a model does not provide good performance on both the training and validation set, meaning that the internal structure is either too simple or has not been trained long enough to learn a meaningful relationship. Figure 2.6 shows a typical training history plot for parametric models trained using gradient-based methods. The validation performance refers to a measure of accuracy on data unseen to the training procedure. Typically, ML practice is to collect or to have access to an entire data set and randomly select validation data to avoid bias in performance evaluation. The ratio between training and validation data is commonly taken to be 70% to 30%. Later on in Chapters 5 and 6, some issues with this practice for this application are discussed and the solutions implemented are explained.

Cross-Validation

The term cross-validation refers to the practice of randomly segmenting a total data set k times, training on all but one segment, validating on the single unseen data segment, and repeating until all data segments have been validated on. This is commonly referred to as k-fold cross-validation, and is primarily used to assess model performance when tuning hyperparameters or when data is limited. Once the k models are obtained, the mean validation performance

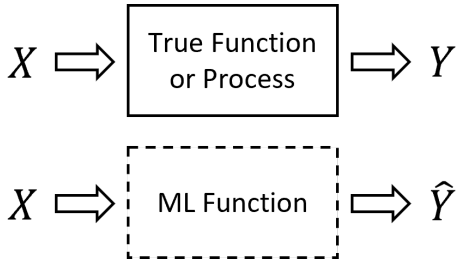


Figure 2.5: High level description of ML approximation process

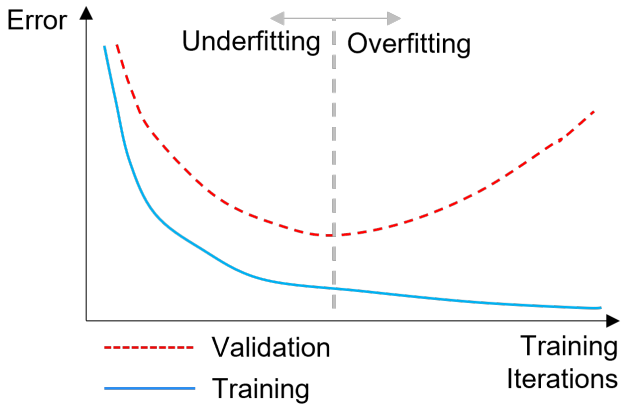


Figure 2.6: Typical training history plot for parametric models

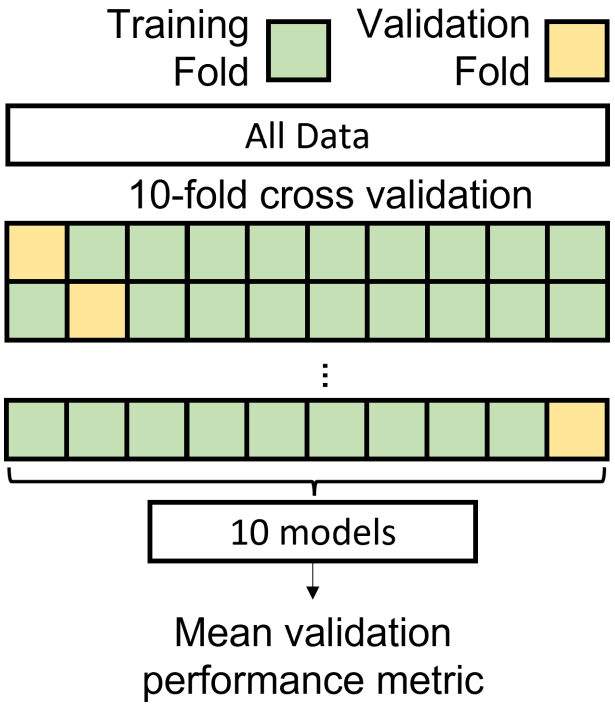


Figure 2.7: Summary of 10-fold cross-validation procedure.

metric is obtained which is a much more robust estimate of model performance. Figure 2.7 explains this practice for $k = 10$ folds, which is a commonly used value. 10-fold cross-validation was used for model training and performance assessment in Chapter 5, but not in Chapter 6. This is because this work generally uses large amounts of data, for which cross-validation is less applicable. Performing 10-fold cross-validation requires that 10 models be trained, which greatly increases the total training time for one model assessment. A new method of selecting training and validation is proposed in Chapter 6 which gives an improved assessment of performance in generalized conditions.

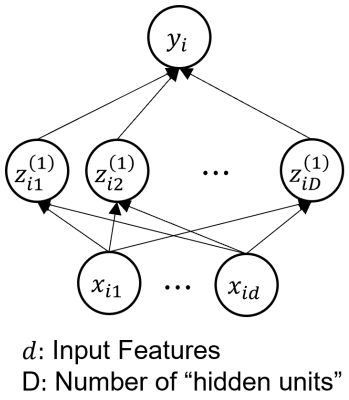


Figure 2.8: Example of a 1-Layer deep ANN

2.3.2 Artificial-Neural-Network

This discussion is primarily based on *Elements of Statistical Learning* [34]. An artificial neural network (ANN) is one such model structure that can be used for function approximation. It is comprised of a number of neurons, each of which computes a linear combination of previous inputs or units and applies a non-linear activation function. Figure 2.8 shows a typical ANN structure, with a single hidden layer consisting of D hidden units. The subscript i refers to the index of the training or testing example x_i, for which y_i is the corresponding output given a fully defined network. An example calculation is shown for the neuron $z_{i1}^{(1)}$:

$$z_{i1}^{(1)} = f\left(w_1 x_{i1} + w_2 x_{i2} + \cdots + w_d x_{id}\right) = f(W^T X_i), \tag{2.13}$$

where $W = [w_1, w_2, ...w_d]'$ is the vector of weights for the linear feature transformation. $X_i = [x_{i1}, x_{i2}, ...x_{id}]'$ is the vector of inputs or previous neurons. f is some non-linear activation function, which could be the sigmoid function, $\sigma(x)$, the rectified linear activation function (ReLU), or some other non-linear function. Given a set of weights that fully define an ANN, estimations of y_i can be obtained by forward computation. To determine a good set of model parameters, gradient based learning approaches are used. Generally all weights are randomly initialized, and then estimations are obtained on a training data set, producing a training error. In regression problems, which is the primary interest of this work, this loss is commonly the mean-squared-error (MSE). The root-mean-square-error (RMSE) is often used for reporting performance as it is in the units of the original signal. These are defined as

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2, \tag{2.14}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2}, \tag{2.15}$$

where N is the total number of examples. Gradient-based approaches tune the network weights and internal paramaters in the negative direction of the gradient, which decreases the loss the most. Most ML packages implement gradient methods using auto differentiation and backpropagation. A discussion of these topics is omitted, but can be found in *Elements of Statistical Learning* [34].

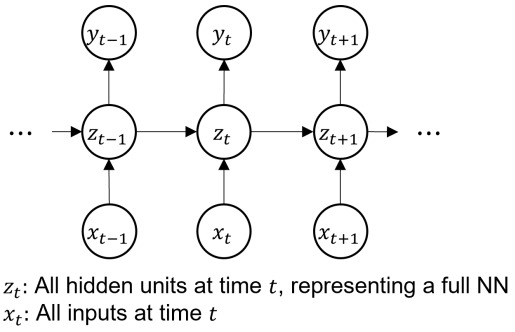


Figure 2.9: Example of an RNN

Gradient-Based Learning Approaches

Fundamentally gradient-based learning approaches compute or approximate the gradient of the loss function with respect to the internal paramters of a NN, in order to update the weights in the direction that minimizes loss. This is known as gradient descent, which is commonly expressed as

$$\vec{\theta_{new}} = \vec{\theta_{old}} - \alpha \frac{\vec{\nabla_\theta} L}{|\vec{\nabla_\theta} L|}, \tag{2.16}$$

where α is the learning rate, $\vec{\theta}$ is the total parameterization of the NN, and L is the selection of loss function computed by the training data. $\vec{\theta}$ is denoted by the subscripts new and old, corresponding to the total parameterization before and after the update. Stochastic gradient descent (SGD) is a modification to this, where rather than computing $\vec{\nabla_\theta} L$ using all training examples, a mini-batch of samples are taken from the training examples to compute the gradient. This generally means that any one update step is less accurate than the full gradient computation, but is more computationally efficient reaching more iterations in less time. The stochastic nature of gradient computation also means that the algorithm is less likely to stay in local minima or saddle points which is undesirable. There are a number of variations to SGD that have been developed, such as AdaGrad, RMSprop or Adam. Adam [35] is a popular optimizer that is generally effective on problems with noisy or sparse gradients. This optimizer is used throughout this work for training parametric models.

2.3.3 Recurrent-Neural-Network

One limitation of an ANN is that it approximates some function relating an example X_i to its output Y_i, without information about other examples. If each of these examples represent a time step at time t, this is equivalent to modelling a static system that has no memory or information about previous time steps. Recursive-neural-networks (RNN) are a generalized modelling approach to modelling dynamic systems or functions that require information from previous time steps. Their overall layout is described by Figure 2.9. The prediction at each time step t is both a function of the input x_t as well as the hidden states at the previous time step z_{t-1}. Mathematically this is expressed as

$$Z_t = f(U^T X_t + W^T Z_{t-1}), \tag{2.17}$$

where f is again some non-linear activation function. U and W are the linear transformations in the parameterization of the RNN that are tuned during training. This way, the RNN is able

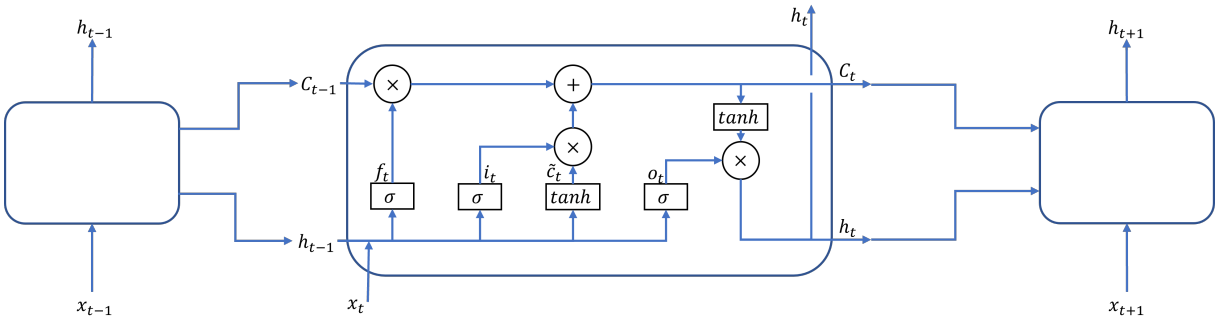


Figure 2.10: Example of the internal structure of an LSTM

to learn time dependent relationships to incorporate information from previous examples into predictions. The network would then be trained through gradient based learning methods, using backpropagation through time. One limitation of this implementation of an RNN is the issue of vanishing gradients. Over multiple time steps, hidden states at previous time steps have less effect on the output at the current time step. Meaning that an RNN can incorporate information from time step $t-1$ effectively into a prediction made at time t, but not from say, $t-10$ or $t-100$ for arbitrarily long sequences of data. To overcome this, RNN variations like LSTM or GRU are used.

2.3.4 Long-Short-Term-Memory Neural-Network

This discussion of LSTM and GRU models is based on *Deep Learning* [36]. LSTM NNs were designed to combat the issue of vanishing gradients by allowing information to persist for arbitrary lengths of time. To do so, they utilize an internal model structure with a cell state that acts as a memory that can be either forgotten, updated, or outputted at any time step. Figure 2.10 shows the internal structure of an LSTM.

Here, c_t represents the cell memory state at time t, and h_t the hidden state. f_t represents the forget gate output, which controls whether information in the previous cell state is kept or forgotten. $\tilde{c}_t$ represents the candidate cell state which is a transformation of the input x_t. The input gate i_t controls how much of the candidate state is saved to the cell state of the current time step. o_t represents the output gate, which controls what information of the cell state moves towards the hidden state which is then passed to the next time step and can be used for prediction output at the current time step. Mathematically, this is expressed as

$$\begin{aligned} f_t &= \sigma(W_f[h_{t-1}, x_t]), \\ i_t &= \sigma(W_i[h_{t-1}, x_t]), \\ o_t &= \sigma(W_o[h_{t-1}, x_t]), \\ \tilde{c}_t &= \tanh\left(W_c[h_{t-1}, x_t]\right), \\ c_t &= f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t, \\ h_t &= o_t \cdot tanh(c_t). \end{aligned} \tag{2.18}$$

where all W represent the parameterization to be trained. The term "gate" is used here for each non-linear sigmoid activation function, as it outputs a value between 0 and 1 acting effectively as a control of how much information is passed. The cell memory state is usually represented by a value between -1 and 1 which is determined by use of the tanh function in making the candidate

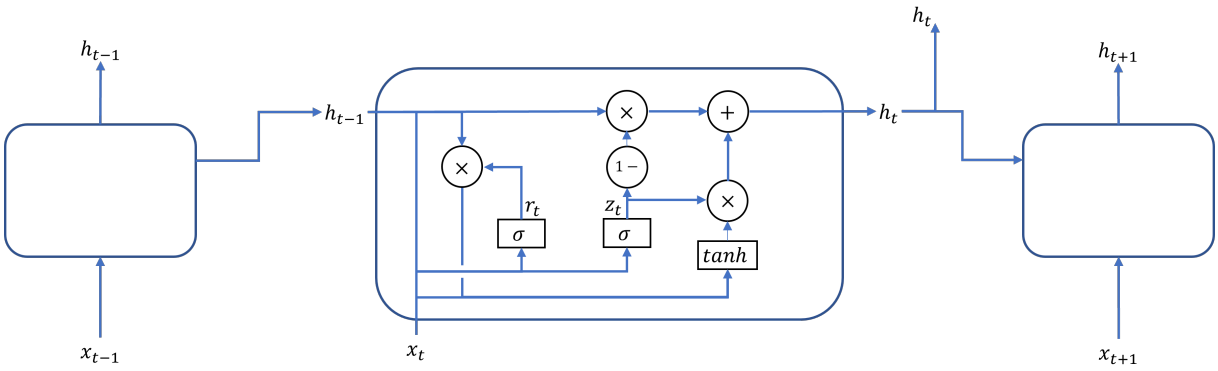


Figure 2.11: Example of the internal structure of a GRU.

cell state. By structuring an RNN this way, the LSTM is able to remember information for arbitrary lengths of time which is a powerful generalized modelling approach for time series problems.

2.3.5 Gated-Recurrent-Unit Neural-Network

A GRU is a simplification of the LSTM structure, which combines the function of the cell memory and hidden state into just the hidden state. The forget and input gates are combined into a single update gate, where there is also a reset gate. Figure 2.11 shows an example of the internal structure of a GRU. r_t is the reset gate vector, and z_t the update gate vector. $\tilde{h_t}$ represents the candidate hidden state vector. Mathematically this is expressed as

$$\begin{aligned} z_t &= \sigma(W_z[h_{t-1}, x_t]), \\ r_t &= \sigma(W_r[h_{t-1}, x_t]), \\ \tilde{h_t} &= tanh(W[r_t \cdot h_{t-1}, x_t]), \\ h_t &= (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h_t}, \end{aligned} \tag{2.19}$$

where all W represent the parameterization to be trained. GRUs generally achieve similar performance to LSTMs, but have simplified internal mechanisms that allow them to train faster on fewer training examples. However, their drawback is that they cannot learn as complex relationships as LSTMs in general.

2.3.6 Gaussian-Process-Regression

This discussion of GPR models is primarily based on *Gaussian Processes for Machine Learning* [37] and *An Intuitive Tutorial to Gaussian Processes Regression* [38]. Before defining what a GPR model is, it is useful to review basic definitions regarding univariate and multivariate Gaussian or normally distributed random variables. A univariate Gaussian is a random variable with probability distribution function (PDF)

$$P_X(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \tag{2.20}$$

which is fully defined by the mean, μ and variance σ^2. Commonly this expressed as $P_X(x) \sim \mathcal{N}(\mu, \sigma)$ as a shorthand notation. When considering multiple random variables that are corre-

lated with each other, it can be described as a multivariate normal distribution

$$P_X(x|\mu,\Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right), \tag{2.21}$$

which is expressed for the collection of random variables $x = [x_1, x_2, ...x_D]'$, where D is the number of dimensions, Σ is the $D \times D$ covariance matrix and μ is the D dimensional mean vector. The covariance matrix contains all information about how each variable is individually distributed, and how they are correlated with each other. In the specific case of a bivariate normal distribution where $D = 2$, the covariance matrix is

$$\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix}, \tag{2.22}$$

where σ_{11} and σ_{22} describe the independent variances of x_1 and x_2 respectively. The off-diagonal terms σ_{12} and σ_{21} represent the correlation between each of the variables. In the generic case of a multivariate normal distribution, there are a large number of entries in this covariance matrix describing the correlation between all variables. This can be constrained by a kernel function or covariance function, which specifies prior knowledge of the correlation between variables. The most common selection of kernel function for GPR models is the squared-exponential kernel function which is defined as

$$k_{SE}(x_i, x_j) = cov(x_i, x_j) = \sigma^2 \exp\left(-\frac{(x_i - x_j)^2}{2l^2}\right), \tag{2.23}$$

which defines the $(i,j)th$ entry of the covariance matrix. This function is parameterized by the length scale, l, and scale factor, σ^2.

A GPR is a non-parametric model, which acts as a "look-up-table" for making predictions using statistical inference on an active set of the training data. The core assumption is that the function being modelled, $f(x)$, can be treated as an infinite dimensional multivariate normally distributed random variable, with a covariance matrix specified by a predetermined kernel function, any samples of which are jointly Gaussian distributed. A GPR defines a probability distribution over all possible functions that fit the training data, from which the posterior is computed by the conditional distribution given the active set and new inputs. The mean of this posterior distribution is taken to be the estimate of the model, with the covariance a measure of uncertainty. Formally this is described as

$$P(\vec{f}|X) \sim \mathcal{N}(\vec{f}|\mu, K) \tag{2.24}$$

where $X = [x_1, x_2, ...x_n]'$ are the observed data points, $\vec{f} = [f(x_1), f(x_2), ...f(x_n)]'$ is the true function, $\mu = [\mu(x_1), \mu(x_2), ...\mu(x_n)]'$ and K is the covariance matrix which is defined by a kernel function $k(x_i, x_j)$. Generally, we don't have access to the true function $\vec{f}$, but only a set of noisy observations from a training data set. When this is specified instead, the probability distribution of all data points is defined as

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu \\ \mu_* \end{bmatrix}, \begin{bmatrix} K(X,X) + \sigma_n^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right), \tag{2.25}$$

which comes from the fact that both the training set and test set are jointly Gaussian distributed. μ and μ_* are the mean functions which are assumed to be constant and specified, and σ_n the specified variance of independent, identically distributed noise on the training data. Commonly

the mean functions are set to be zero, which is suitable for data normalized to have zero mean. In the context of a GPR, the kernel function encodes the "smoothness" of the output function. When two variables x_i and x_j are considered similar by the kernel function (as determined by the length scale parameter, l), then the outputs $f(x_i)$ and $f(x_j)$ have a higher correlation strength and are expected to be similar. Predictions are then made on new data by computing the posterior distribution

$$\begin{aligned} f_*|X, y, X_* &\sim \mathcal{N}(\bar{f}_*, \Sigma_*), \\ \bar{f}_* &= \mu_* + K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}(y - \mu), \\ \Sigma_* &= K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-}1K(X, X_*), \end{aligned} \tag{2.26}$$

which comes from computing the conditional distribution of f_* given X, y and X_*. A GPR is able to make predictions directly from the data, without requiring a gradient-based training method. As there are only a few hyperparameters within the kernel function, which are usually selected using pre-determined methods, they are much more robust to hyperparameter selection. Their primary drawback however is their high computational cost, which comes from (2.26) and increases as $O(N^3)$. To get around this, most common packages that implement GPR models use sparse matrix versions to decrease computation time.

Chapter 3

Experimental Methods

For this study, a multirotor drone was instrumented to measure the full set of state and control inputs, acting as a valuable platform for obtaining training, validation and testing data. This chapter explains the work developing this system, and the data collection methods used.

3.1 Hardware Development

To collect flight data, a Indro Robotics Scout MKIIIB which is based on a DJI Matrice 100 airframe was instrumented with appropriate sensors. The drone uses a Pixhawk flight control system, which measures orientation, acceleration, and GPS position and velocity internally in flight. These measurements are available as a dataflash log, which can be manually downloaded from the drone after each flight. GPS measurements are internally recorded at a rate of 5Hz, orientation measurements (which come from internal inertial-measurement-unit (IMU) data fusion) are recorded at 10Hz, and acceleration recorded at 25Hz. Each of these signals are later downsampled to match the anemometer measurement rate. The drone with modifications is shown in Figure 3.1.

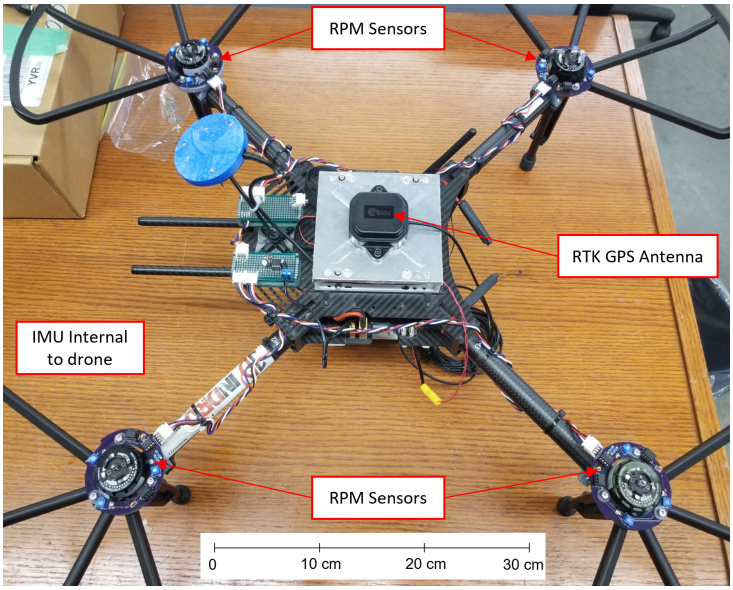
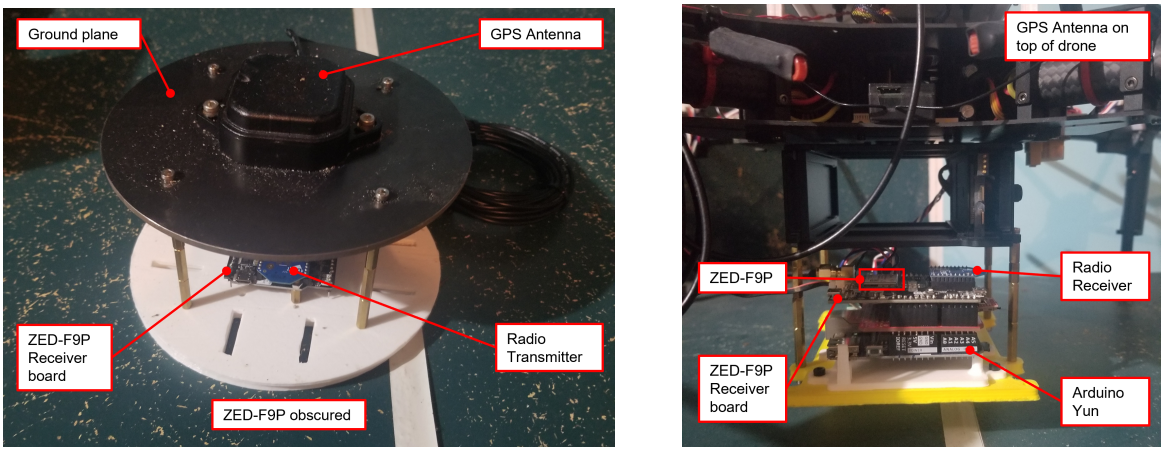


Figure 3.1: Overview of multirotor drone system and instrumentation.

A real-time-kinematic (RTK) GPS measurement system was added to improve the ground speed and position measurement accuracy. On paper, this system is capable of achieving centimeter level positioning accuracy. To do so, a ground station GPS receiver is required that is located at either a known or constant location. This ground station then sends radio correction messages to the moving GPS receiver, which computes an improved accuracy position



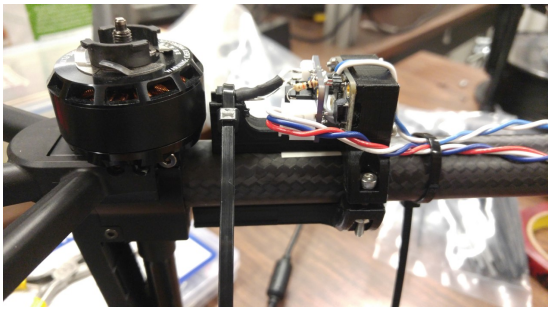
(a) Ground station (b) Rover

Figure 3.2: Picture of RTK GPS ground station and rover based on ZED-F9P board.

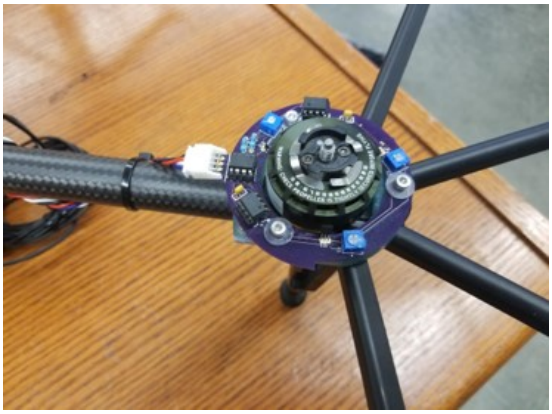
based on the corrections and its own GPS measurements. This system was based on two U-blox ZED-F9P receivers, with one configured for base station operation and the other configured as a moving rover. Pictures of each of these are shown in Figure 3.2. A conductive ground plane is recommended for maximizing the antenna's view of the sky. This is implemented with a steel plate on the ground station and aluminum tape on the drone to save weight. When the precise location of the ground station is not specified, it is determined by the receiver's observations over time meaning that more precise measurements are obtained the longer the ground station has been stationary. Before any flights, the ground station was left stationary for 10 minutes which is recommended to achieve nominal accuracy. One source of bias in the data is that the GPS accuracy depends on the time the ground station has been stationary for, the number of satellites in view, and partially on the movement of the rover. The self-reported accuracy achieved was typically 0.3 m in position and 0.1 m/s in velocity. To achieve centimeter level position accuracy, the ground station is required to be stationary for up to 24 hours previously. For reference, plots of GPS self-reported accuracy are included in Appendix A.

3.1.1 Motor speed sensor development

To develop the motor speed sensors, multiple design iterations were considered with the help of fellow student Miayan Yeremi in the UBC Aerosol Lab. The first design utilized an optical sensor to measure the passage of a segment of reflective tape on the surface of the motor. This system worked well in the lab, but poorly in the field when flying in ambient light conditions as the stray daylight created false readings corrupting the measurement. This motivated the final design, which is based on a hall effect sensing principle. On a custom printed-circuit-board (PCB) design (which was provided by Miayan Yeremi), three hall effect sensors are equally spaced around each motor. These sensors measure the changing magnetic field of the spinning rotor, which produces a changing analog voltage. A comparator op-amp circuit turns this into a rising and falling edge signal, which an ATtiny85 microcontroller embedded on the PCB computes motor speed from using the pulse counting method. This is then communicated to an Arduino Yún logging system which saves the measurements to an SD card. The pulse counting method functions by specifying a fixed interval of time, and then using an interrupt service routine on the microcontroller to count the number of rising and falling edge signals per timing



(a) Original



(b) Final

Figure 3.3: Comparison of first and last design iteration of custom motor speed sensors.

interval which can then be converted to motor speed.

Appendix B describes in detail the experiments and analysis that went into developing the motor speed sensors. In summary, there is an expected maximum thrust estimation resolution of 0.05 N at the maximum operating speed which is considered negligible in this application. Typical motor thrust measurements here are on the order of 30 N.

3.1.2 Anemometers

To measure the true wind speed, two ultrasonic anemometers are used. The first is a Gill Instruments WindSonic 60, and is a 2-axis sonic anemometer. The second was introduced only for tests conducted in 2022 is a Gill Instruments WindMaster, which can measure 3-axis wind fields. Each of them measure the wind speed and direction by time of flight theory from ultrasonic pulses. Between two probes, the transit time of ultrasonic pulses are measured moving in both directions which eliminates effects of temperature. The relevant performance parameters of each anemometer are summarized in Table 3.1. The 2-axis anemometer samples at the maximum rate of 4 Hz. The 3-axis anemometer is set to sample at 10 Hz, but can sample as fast as 20 Hz. The accuracy of the anemometers as the wind speed approaches zero is considered in Appendix C. To do so, measurements are collected for approximately 30 minutes with plastic bags surrounding each anemometer corresponding to zero wind speed. The measured errors are on the order of 0.005 m/s, which is considered negligible for this application.

Parameter	Gill WindSonic 60	Gill WindMaster
Axis	2-axis	3-axis
Max Sample Rate	4Hz	20Hz
Threshold	0.01 m/s	0.01 m/s
Resolution	0.01 m/s	0.01 m/s
Range	0-60 m/s	0-50 m/s
Accuracy	±2% @ 12 m/s	±2% @ 12 m/s
Output Format	RS232	RS232

Table 3.1: Anemometer performance specifications.

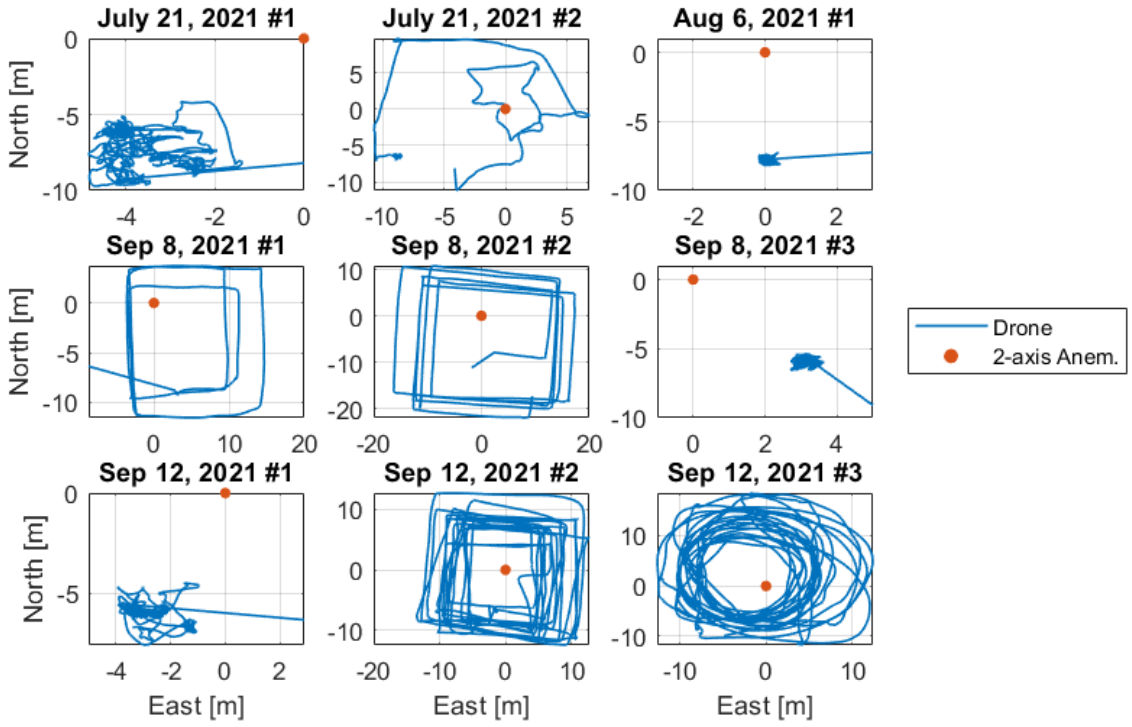


Figure 3.4: First nine test flights overview.

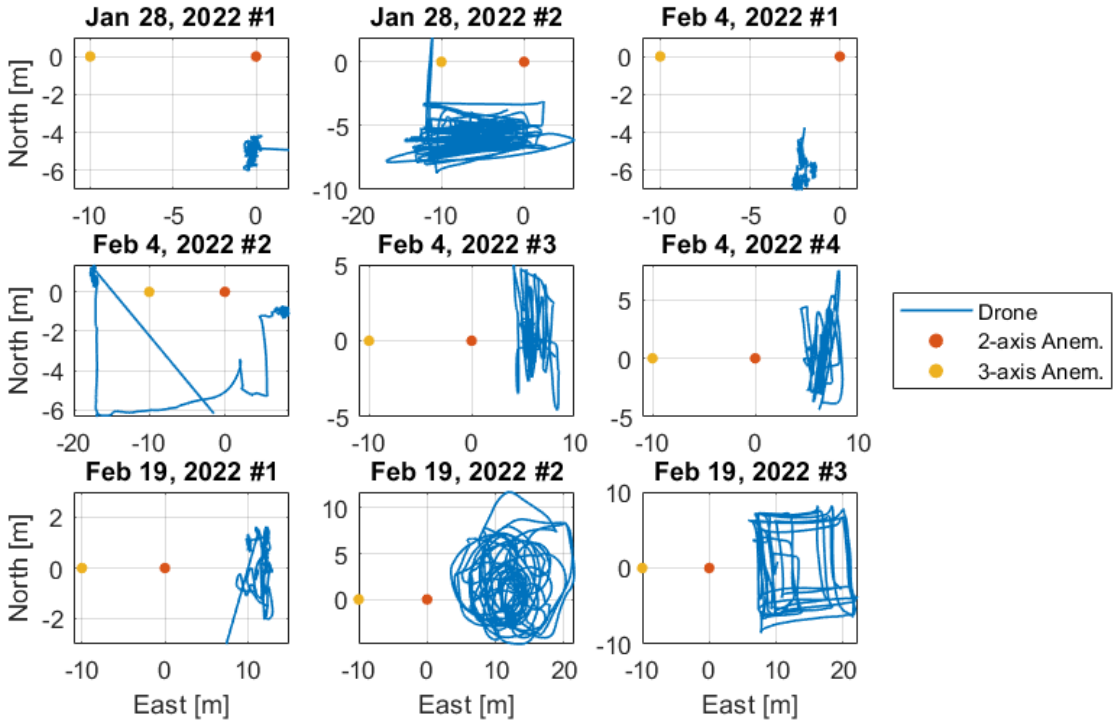


Figure 3.5: Second nine test flights overview.

Figure 3.6: Example picture of the drone in flight hovering near the 2-axis sonic anemometer.

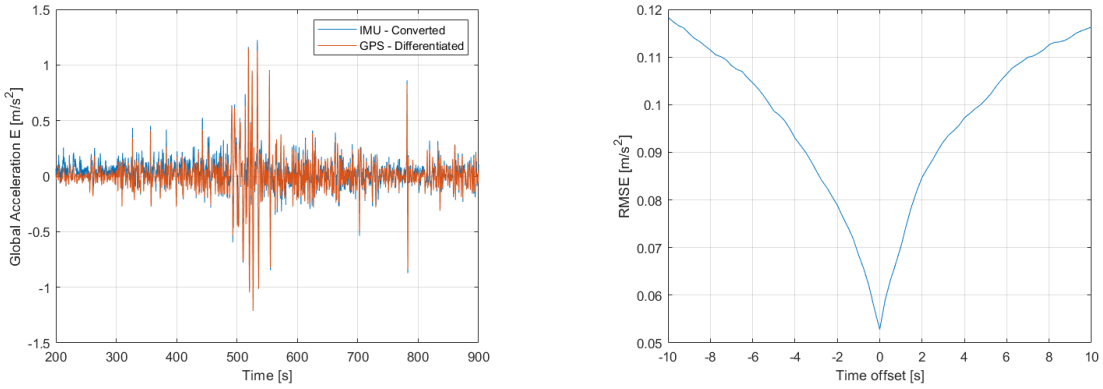
3.2 Flight Tests

Data was collected over seven different days of test flights, specifically on July 21st 2021, August 6th 2021, September 8th 2021, September 12th 2021, January 28th 2022, February 4th 2022 and February 19th 2022. The test flight days in 2021 were flown with respect to a single anemometer, whilst in 2022 a second anemometer was introduced to assess the spatial variation of the wind. These flight paths were selected to reduce the spatial variability between the anemometer and drone locations, while not allowing the anemometers to be influenced by the downwash of the propellers. Dynamic flight phases are included to try to generalize the results to arbitrary flight paths. In principle, it is desirable for the data set to encompass all airspeed ranges in all drone orientations, to learn the drag-airspeed relationship in all orientations. Circular flights achieve this somewhat effectively. Figures 3.4 and 3.5 show each complete flight trajectory. Figure 3.6 shows an example picture of a drone flight in hover.

3.3 Signal Processing

To process the data before applying wind estimation models, it is desirable to convert the measurements into the units and reference frames used by the approximations. That is, to convert the accelerations which are measured in the body frame of reference to the global frame of reference. To do so we apply the transformation matrix R given by (2.2).

To align data from multiple sources temporally, we use the absolute timestamps of measurement given by each sensor. For example, each sensor provides a time measurement in coordinated universal time (UTC) which can be converted to a local time. One concern when doing this is the possibility of time measurements being systematically offset due to time errors. To pre-process the data, we consider comparing signals that are known to be the same from sensor sources and vary the time offset by discrete amounts. Then, the RMSE is compared between these signals, where it is expected the global minimum in error will be reached when the time offset is correct. This is well explained by Figure 3.7 where the difference between the acceleration as measured by the IMU is converted to the global coordinate frame and compared to the differentiated GPS velocity signal. The global minimum in error at a time offset of 0 indicates that we have correctly offset the signals.



(a) Time series, for true offset

(b) RMSE

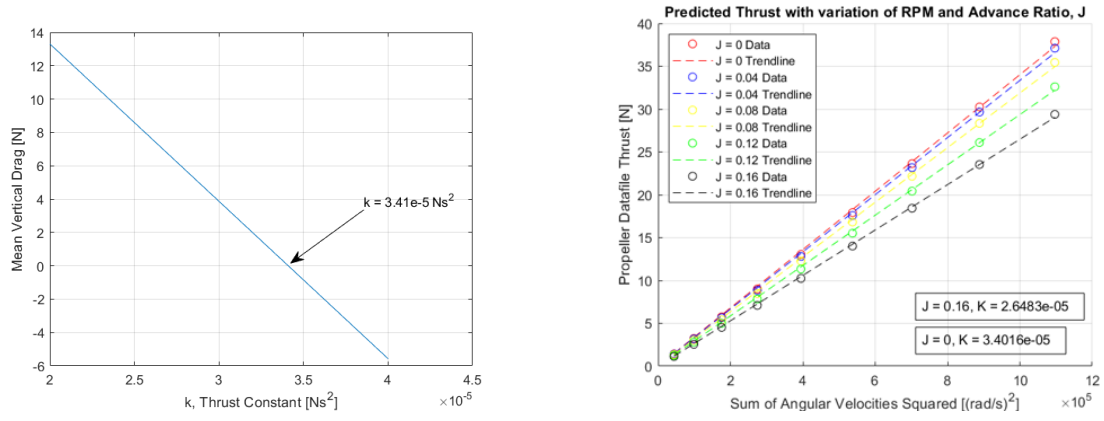
Figure 3.7: Comparison of how RMSE changes for varying time offset.

3.4 System Identification

When utilizing the hybrid, drag-to-airspeed, drag-to-windspeed or physics-based approaches, the inverse equations of motion are used to produce estimates of the drag. These equations require knowledge of the plant parameters k and m which represent the propeller thrust constant and mass of the drone respectively. For the mass of the drone, we take this to be 3.2 kg which comes from directly weighing the drone with all modifications added. To identify the propeller thrust constant, we consider a combination of theoretical modelling and experimental data. Consider the vertical component only of (2.3), which describes the relationship between the drone state and the vertical component of drag,

$$D_z = -gm + C_\theta C_\phi F - \ddot{z}m. \tag{3.1}$$

The first test flight from August 6th, 2021 is used which is a hover at a constant low-altitude. In this flight with negligible vertical movement close to the ground, we expect the vertical component of drag to be zero on average although this can fluctuate due to vertical



(a) Experimental Approach

(b) Theoretical Approach

Figure 3.8: Identification of thrust constant, k

wind eddies. The thrust constant is varied over a range to determine at which value this is true, as shown in Figure 3.8.

The theoretical approach comes from taking the propeller data files for the propeller geometry [39], and plotting the thrust as a function of RPM. This is varied for changing advance ratios to see how the thrust constant may vary when the drone isn't flying at a constant altitude. From theoretical modelling with zero advance ratio, we expect a propeller thrust constant of $3.4 \times 10^{-5}\text{Ns}^2$. From experimental modelling, we achieve a very similar result of $3.41 \times 10^{-5}\text{Ns}^2$. The similarity between these values is taken to indicate that we likely have properly identified the propeller thrust constant for drag estimation.

Chapter 4

Physics Based Modelling

Before applying ML models to this problem, it is useful to consider the application of purely physics based modelling as it has a number of simplifications and advantages. The main advantages of using physics modelling rather than ML are a high degree of interpretability, improved domain limitations in validation, and inherent rotational invariance. While ML modelling is generally treated as a black box, physics-based models can be examined at every step. Changes to the plant parameters are easily implemented. When validating a ML based model, one must consider the domain or operating range that is being validated over. This means that the validation performance metrics achieved in ML only apply over the range of operating conditions covered by the data set. When applying a model on a range of data outside the trained or validated data set, nothing can be said about the expected performance due to the highly non-linear nature of these methods. As will be later discussed in Chapter 6, rotational invariance is an important property of real systems that is inherent to the equations of motion but not in ML models without special considerations. Despite all these challenges, the main feature of approximating a real system by ML models is the ability to fit highly complicated, unknown or non-linear relationships that are not well captured by simplified physics. This section goes over applying a purely physics-based modelling approach with simplifying assumptions to show the strengths and weaknesses when applied to this problem.

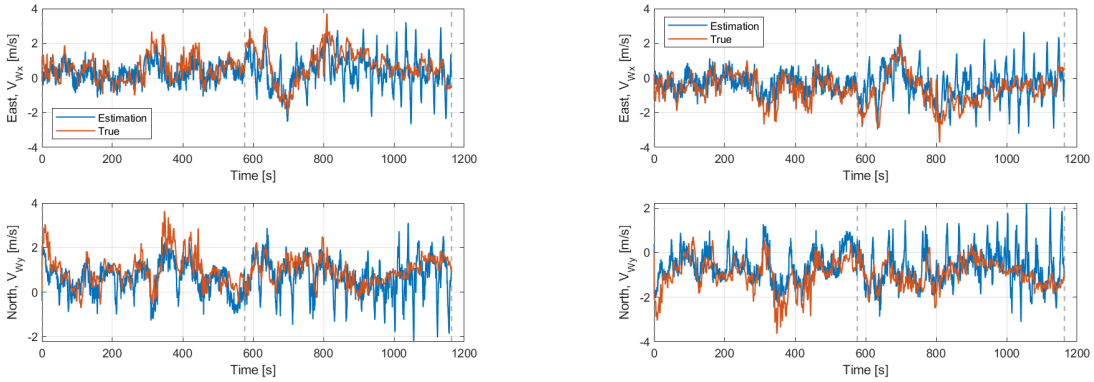
4.1 Constant Drag Coefficient

As introduced in Chapter 2 there is a deterministic relationship between the drone state and the wind magnitude, but the specific relationship between the drag and airspeed is unknown. This is because the drag and lift coefficients vary with the changing orientation of the drone in three dimensions, which are complex and unknown plant parameters. This relationship is derived in (2.4), where the zero-lift assumption is applied. A further simplifying assumption is to treat the drag coefficient as constant, and to determine the accuracy of wind estimation given this to check its validity. Mathematically, this means to go from the drone state to the wind estimation, the drag estimate is first computed by (2.3), and then the airspeed estimate is computed by

$$\begin{aligned} |\hat{\vec{V}}_a| &= \sqrt{\frac{2|\hat{\vec{D}}|}{\rho S C_D}}, \\ \begin{bmatrix} \hat{V_{Ax}} \\ \hat{V_{Ay}} \\ \hat{V_{Az}} \end{bmatrix} &= \frac{|\hat{\vec{V}}_A|}{|\hat{\vec{D}}|} \begin{bmatrix} \hat{D}_x \\ \hat{D}_y \\ \hat{D}_z \end{bmatrix}, \end{aligned} \tag{4.1}$$

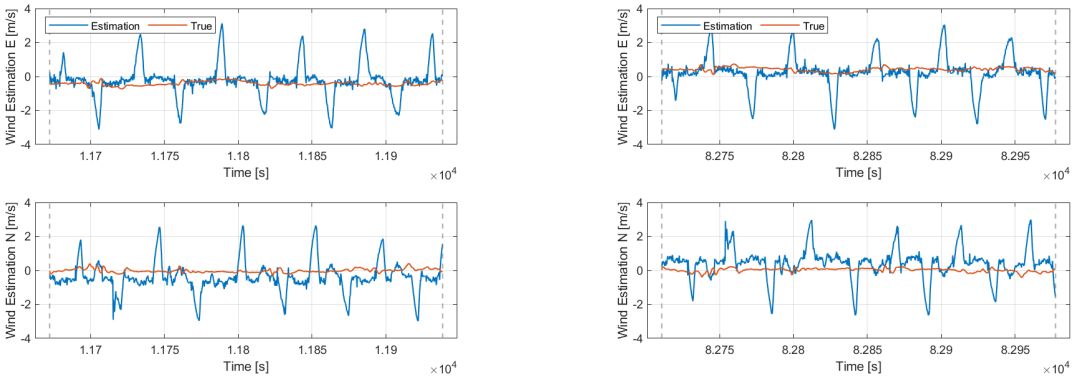
where the drag area product SC_D is now assumed to be a constant irregardless of drone orientation. It should be noted that these and all other equations in Chapter 2 are derived for three

dimensions, but we will neglect the vertical component. Finally, the windspeed is estimated by applying the wind triangle, previous described by (2.6).



(a) Original Data (b) Rotated

Figure 4.1: Applying constant drag coefficient wind estimation to Sep 12 2021 Test flights #1 and #2, representing high wind with low and high ground speed. $SC_D = 1$ m^2.



(a) Original Data (b) Rotated

Figure 4.2: Applying constant drag coefficient wind estimation to Sep 8 2021 Test flights #2, representing near-zero wind with high ground speed. Here $SC_D = 1$ m^2.

Applying these equations yields the following wind predictions for selected complete test flights. Figure 4.1 shows the wind estimation for $SC_D = 1$ [m^2] on the first two test flights conducted on September 12 2021. These flight conditions represent high wind, with the first

Test Flight	Typ. Ground speed	Typ. Wind speed	Wind RMSE East	Wind RMSE North
Sep 12 #1, 2021	1 m/s	3 m/s	0.64 m/s (75.4%)	0.73 m/s (58.9%)
Sep 12 #2, 2021	5 m/s	3 m/s	1.02 m/s (84.6%)	1.09 m/s (78.7%)
Sep 8 #2, 2021	6 m/s	0.5 m/s	0.96 m/s (220%)	0.95 m/s (847%)

Table 4.1: Summary of performance for constant drag area assumption. Performance is consistent with rotation.

flight in hover and the second in a dynamic square flight pattern. The wind estimations with this simplifying assumption in the first flight achieve good accuracy, reaching 0.64 m/s RMSE and 0.73 m/s RMSE in east and north wind estimations respectively. These performance accuracy metrics do not change when rotating the global coordinate frame used, which is shown in the same figure for the data rotated by 2.8 rad or 160°. When adding movement, which is represented by moving to the second flight of that day the performance decreases to 1.02 m/s RMSE and 1.09 m/s RMSE for east and north wind respectively. This is because the constant drag coefficient assumption holds for small deviations away from equilibrium only, and begins to fail when the drone banks aggressively. Since wind speed is estimated as the difference between the airspeed and the ground speed, if the airspeed is underestimated for high ground speed then the wind speed will be over estimated. This effect is strongly seen in Figure 4.2, where there is dynamic movement but very low wind. All of these performance metrics are summarized in Table 4.1.

4.2 Polynomial Drag-Airspeed Modelling

Another way of modelling this relationship is to fit a polynomial for each independent axis of the drag-airspeed relationship. This is similar to fitting a generic drag coefficient, that is allowed to change arbitrarily in each axis as the drone banks. This idea is introduced here, but is later thoroughly discussed in Chapter 5 where it is shown to provide less accurate wind estimations than developed ML models.

4.3 Machine Learning Motivation

Thus, it is possible to apply simplifying assumptions to the drag-airspeed relationship in order to develop a physics-based disturbance observer but this generally results in a lack of performance in generalized flight. A constant drag coefficient does not sufficiently capture the change from hover to dynamic flight, and fitting a generic polynomial to this relationship yields less accurate estimations overall. Despite the previously mentioned drawbacks of approximating real systems using ML models, the ability to fit this unknown relationship using a purely data-driven approach is highly attractive and is the primary motivation for considering ML. This leads to Chapter 5, which is the first study conducted applying ML to this problem.

Chapter 5

Wind Estimation by Multirotor Drone Dynamic State Measurement and Machine Learning Models

This chapter contains a version of a journal article that has been accepted for publication as [40]. The primary author is Steven Zimmerman, with co-authors Dr. Ryozo Nagamune, Miayan Yeremi and Dr. Steven Rogak.

5.1 Introduction

Wind estimation is a difficult but important task, particularly in the lower atmosphere where the wind is variable. The earth's surface, temperature inversions and turbulence all contribute to wind modelling inaccuracies in this region [1], leading to the need for well-developed experimental methods. Being able to measure the spatially and temporally varying wind field has a number of applications and advantages in gas source localization, flux emissions measurement, disturbance rejection in control engineering, wind farm site surveying and others. In environmental studies, measuring a gas or pollutant concentration coupled with knowledge of the wind field leads to gas distribution mapping, gas source localization and flux emissions measurement. For gas source localization, typically a mobile robotics platform moves through an environment detecting concentration and/or airflow to find the source. Many researchers [7, 8, 9] present gas source localization methods, typically using biology-inspired airflow or concentration gradient ascent algorithms for indoor or outdoor environments. For flux emissions measurement, typical methods include Eddy-covariance (EC) [5], mass balance box models, tracer dispersion methods, or near-field Gaussian plume inversion (NGI) [3]. In these approaches, the general method is to use a mobile or stationary sensor platform to measure pollutant concentration and wind speed downwind of an emissions source to infer the total flux. In aeronautical controls engineering, local wind estimation is necessary for disturbance observer based controller design, favoured in systems with low inertia and high susceptibility to wind [14]. In mission critical or high efficiency applications, knowledge of the wind field allows for advanced path planning that that avoids actuator saturation or utilizes the wind field for soaring [15]. Another key application of wind field measurement is flow mapping around buildings or complex terrains to inform the design and placement of wind turbines and other structures. Generally the efficiency and power output of a wind turbine depends on the wind speed distribution over the entire area of the blades [12] making it important to study the spatially and temporally varying wind field. This is typically done by mobile platform [10].

Despite the wide and far-reaching applications of wind measurement, this problem is not fully solved. While the ideal technique measures a spatially and temporally varying wind field, practically this is infeasible and typical measurement involves assuming some level of spatial uniformity, at varying scales. This is particularly true when wind is measured by fixed-location

anemometers on tall masts [16], which is usually done when long timescale, detailed information is required about one location typically in wind or weather measurement stations. Expensive sensors such as light-detection-and-ranging (LIDAR) can be used here as the cost is justified by the long measurement time. Sensors such as these can measure a wind field over spatial dimensions, although this is limited to some fixed range. When information is required over multiple sites of interest, or when there are changing location requirements, one may favour the use of a fixed-wing drone. The relatively high efficiency of such drones enables high altitude flight up to 1500m and durations of up to an hour [17]. They lend themselves naturally to wind measurement as free-stream airspeed measurement sensors such as pitot-tube anemometers are common. Such systems can survey a large number of sites easily, but suffer in the amount of information per individual site. As fixed-wing drones require a minimum airspeed to fly, they require very high sample rate hardware in order to achieve the same spatial measurement resolution leading to high hardware costs. Multirotors offer an intermediate trade-off in this spectrum of wind measurement devices. They are able to hover and move as desired, allowing for detailed information about specific site locations and allowing for much lower sample rate and lower cost hardware. Improved drone flight technology such as differential or real-time-kinematic (RTK) global positioning system (GPS) systems allow the drone to hover at centimeter level accuracy. Improved battery technology leads to extended flight times that allow for entire sites to be surveyed in detail at once, at distances horizontally and vertically up to 1 kilometer or more. Multirotor drones allow for highly versatile payloads, and can easily take packages specific to the application such as lightweight gas concentration sensors. The primary issue when using multirotors is the errors introduced in anemometer measurements when including such a sensor in the payload, as the downwash of the propellers changes the airflow over such a sensor relative to the free-stream air. This could be overcome by attaching a lightweight anemometer on a thin boom far away from the propellers [1], but this comes at the trade-off of decreased flight time and worsened flight handling due to the asymmetrical structure and weight added. The primary method used to overcome this limitation in wind estimation by a multirotor is through the development of a disturbance observer or estimator that is able to infer the wind acting on the drone by knowledge of the state and control inputs of the drone, that uses a minimized number of additional sensors to common drone packages.

Neumann [22] presented the first significant experimental results that used a simplified physics model to estimate the wind from the pitch and roll angles of a multirotor, assuming static equilibrium. This work required identification of the non-linear relationship between the bank angle of the drone and the airspeed, which was found using wind tunnel testing. This pioneering approach was not accurate during dynamic flight phases, and the approach does not generalize well to different multirotors as each new drone requires wind tunnel testing to find its particular drag-airspeed characteristics. Allison [29] presented a simulation study that improves upon these results, utilizing a machine learning (ML) approach using a long-short-term-memory neural network (LSTM NN) that relates the measured drone state variables of position and orientation over time to the wind speed. In their simulation environment, they produce wind using a dryden wind turbulence model and simulate a drone's response with controlled drag coefficient characteristics. Crowe [31] builds upon this work, applying the same LSTM NN ML technique to experimental data. They include dynamic effects by training their network on drone orientation and acceleration as regression features to predict the wind speed.

In this paper, we improve upon these published works through multiple efforts. The effects of an added sensor input giving control inputs are considered, increasing the complexity of the models. New models are trained that use a hybrid physics-ML approach, utilizing well understood equations of motion to pre-process the drone state and limit the machine learning to the

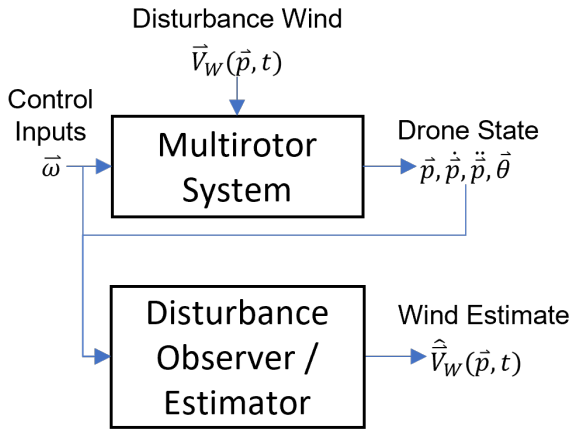


Figure 5.1: Overview of the problem formulation expressed as a disturbance observer to estimate the wind acting on a drone.

ill-understood portions of the model. This work can be applied to any general multirotor, given sufficient training information to create a model specific to a given drones drag characteristics. This paper is organized as follows: Section 5.2 describes the initial approach and framework for applying ML, followed by Section 5.3 describing the experimental approach. Then the process for training the models and assessing their performance is described in 5.4 and 5.5. The significance and context for these results is discussed in Sections 5.6 and 5.7, the discussion and conclusion.

5.2 Problem Formulation

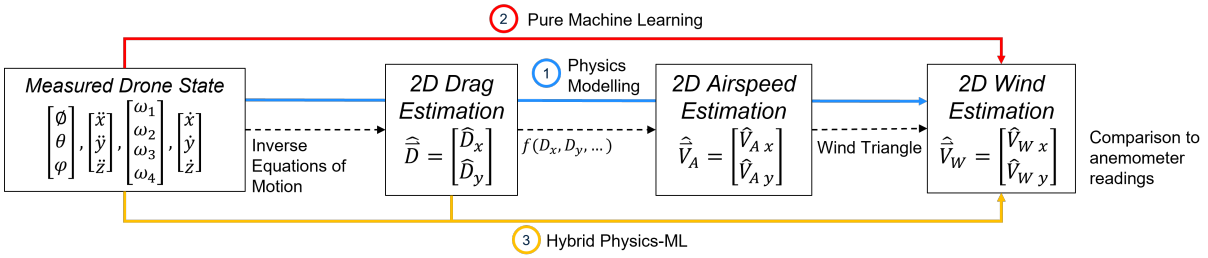


Figure 5.2: Overview of wind estimation calculations, and how approximations by machine learning methods contribute to the wind estimation.

The problem this paper addresses is the design and implementation of a disturbance observer that is able to estimate the disturbance wind acting on the drone from the state and input measurements, described by Figure 5.1. Here, $\vec{p} = [x, y, z]'$ [m], $\dot{\vec{p}} = [\dot{x}, \dot{y}, \dot{z}]'$ [m/s], $\ddot{\vec{p}} = [\ddot{x}, \ddot{y}, \ddot{z}]'$ [m/s^2] represent the position, velocity, and acceleration of the drone relative to the ground in a global reference frame. $\vec{\theta} = [\phi, \theta, \psi]'$ [rad] are the pitch, roll and yaw angles of the drone. The control inputs are expressed $\vec{\omega} = [\omega_1, \omega_2, \ldots \omega_R]$ [rad/s] as all the rotor speeds of the multirotor, with R representing the number of rotors. These fully define the control forces and moments acting on the drone. The disturbance wind varies in time and space and the drag force due to wind depends on the drones velocity and state at any given time. This section

describes the theory of modelling behind the design of such a disturbance observer starting with a physics-based approach, moving to a ML solution and the development of a hybrid physics-ML method. These three approaches are summarized by Figure 5.2.

5.2.1 Physics-Based Modelling

Starting with the non-linear equations of motion for the rigid body dynamics of a multirotor [33], we can arrive at multiple wind estimations with varying levels of complexity. This is a standard model based in the global (X-Y-Z) coordinate frame which is aligned with the earths east-north-up (E-N-U) coordinate frame shown in Figure 5.3:

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} (C_\psi S_\theta C_\phi + S_\psi S_\phi)F - D_x \\ (S_\psi S_\theta C_\phi - C_\psi S_\phi)F - D_y \\ -mg + C_\theta C_\phi F - D_z \end{bmatrix}, \tag{5.1}$$

where C_x and S_x denote $\cos(x)$ and $\sin(x)$ respectively. The total thrust force from all the rotors applied to the center of gravity is represented by F [N], and m [kg] the mass of the drone. g [m/s^2] is the acceleration due to gravity. $[D_x, D_y, D_z]'$ [N] is the drag force acting on the drone due to net airspeed either by wind or movement of the drone. Based on the assumption that the wind produces a net force alone and no moment on the drone, we can ignore the rotational equations of motion for this wind estimation problem. These equations can be inverted to estimate the drag force acting on the drone based on the measured drone state from drone instrumentation.

$$\begin{bmatrix} D_x \\ D_y \\ D_z \end{bmatrix} = \begin{bmatrix} (C_\psi S_\theta C_\phi + S_\psi S_\phi)F - \ddot{x}m \\ (S_\psi S_\theta C_\phi - C_\psi S_\phi)F - \ddot{y}m \\ -gm + C_\theta C_\phi F - \ddot{z}m \end{bmatrix}, \tag{5.2}$$

$$\begin{bmatrix} D_x \\ D_y \\ D_z \end{bmatrix} = \frac{1}{2}\rho V_a S(\cdot) C_D(\cdot) \begin{bmatrix} V_{Ax} \\ V_{Ay} \\ V_{Az} \end{bmatrix} = f(\vec{V_A}), \tag{5.3}$$

$$F = k(\omega_1^2 + \omega_2^2 + \cdots + \omega_R^2). \tag{5.4}$$

The airspeed is represented by the vector $\vec{V_A} = [V_{Ax}, V_{Ay}, V_{Az}]'$ [m/s]. If a standard turbulent drag equation is used, then the airspeed is determined by a drag coefficient relationship (5.3). Otherwise, this can be treated as a generic non-linear function relationship. The force from the rotors can be estimated using the quasi-static assumption (that the vertical component of thrust equals the drone weight) or from control inputs (5.4). Here, k [Ns^2] is the total average propeller

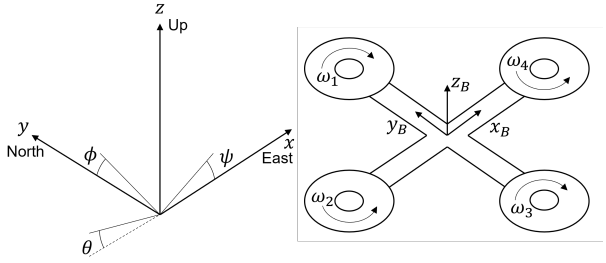


Figure 5.3: Coordinate frame used for notation. Quadrotor shown but derivation is for generic multirotor.

thrust constant and $\omega_{1,2,...R}$ [rad/s] are the rotor speeds of each individual propeller. R is the total number of motors and propellers on the multirotor. As later described in Section 5.3, we will use a standard quadrotor and therefore $R = 4$. Here, ρ [kg/m^3] is the density of air, $S(\cdot)C_D(\cdot)$ [m^2] is the drag area expressed as some unknown, nonlinear function of the drone state, which may include the rotor speeds, orientation and relative airspeed direction to the drone. In general, estimating the drag coefficient of the drone at an arbitrary orientation and state is a difficult task so this is left up to experimental correlation or machine learning approximation. As the drone banks, the cross-sectional area increases while the momentum drag due to changing streamlines decreases. For hover flight and small deviations away from equilibrium, the drag coefficient can be assumed to be approximately constant, but this assumption does not hold well for all flight phases. As a first approximation, we can consider a generic non-linear function, that is assumed to be independent in each principal axis (5.5). Each relationship is approximated by a polynomial for comparison with the machine learning techniques. Finally, the wind is estimated from the determined airspeed and measured ground speed using vector subtraction (otherwise known as the "wind triangle").

$$\begin{aligned} \hat{\vec{V}}_W(\vec{p}, t) &= \dot{\vec{p}} - \hat{\vec{V}}_A \\ &= \dot{\vec{p}} - g_{poly}(\hat{\vec{D}}). \end{aligned} \tag{5.5}$$

5.2.2 Machine Learning Implementation

Machine learning has been shown to be effective for flight dynamics applications, particularly recurrent Neural Networks (RNN's) [41]. Long-Short-Term Memory Neural Networks (LSTM NN's) have been presented for this wind estimation problem [29, 31] as they are able to incorporate data from previous time steps into their estimations effectively. Viewing the wind estimation problem in this way, there is a generalized non-linear equation from the drone state directly to the wind estimation. Combining equations (5.2), (5.3) and (5.4), identifies the relevant state variables to this generic relationship,

$$\hat{\vec{V}}_W(\vec{p}, t) = g_{ML}\left(\dot{\vec{p}}, \ddot{\vec{p}}, \vec{\theta}, \vec{\omega}\right). \tag{5.6}$$

This equation represents the entire unknown relationship between the drone state and the wind at any time step, which can be approximated by ML techniques. Specifically considering an LSTM NN, they typically consider time-series problems where each example consists of a window of previous observations with targets that can be one value or a sequence. Here, we consider a regression problem predicting the wind at time step $T-1$ based on an observation window from time T to $T-n$,

$$\begin{aligned} \hat{\vec{V}}_W(\vec{p}, T-1) = & g_{ML-LSTM}\big(\dot{\vec{p}}(T), \dots \dot{\vec{p}}(T-n), \\ & \ddot{\vec{p}}(T), \dots \ddot{\vec{p}}(T-n), \\ & \vec{\theta}(T), \dots \vec{\theta}(T-n), \\ & \vec{\omega}(T), \dots \vec{\omega}(T-n)\big). \end{aligned} \tag{5.7}$$

This is the formulation expressed by Allison [29] and Crowe [31], with added control inputs and accelerations as regression features. For comparison however, multiple models are developed with different scopes within the wind estimation framework. While the position or velocity of

the drone in the global frame could be used to predict its derivative over time using multiple time steps, both velocity and acceleration are used as regression features as they come from different sensor sources and have different sources of error, avoiding co-linearity issues.

5.2.3 Hybrid Physics-Machine Learning Implementation

While the previously described pure ML formulation approximates the entire process from state to wind estimation, we can develop a formulation that uses the well-known equations of motion as a pre-processing step isolating the ML to the ill-understood portions of the model. Specifically, this refers to the complex drag area relationship expressed by (5.3). This way, the non-linear function being approximated is as in (5.8), where the relationship between drag and airspeed depends on some non-linear function of the drone state:

$$\hat{\vec{V}}_W(\vec{p},t) = g_{Hybrid}\left(\hat{\vec{D}},\dot{\vec{p}},\ddot{\vec{p}},\vec{\theta},\vec{\omega}\right). \tag{5.8}$$

The hybrid physics-ML approach simply adds the regression feature of the drag estimation by drone equations of motion, reducing the scope of approximation required by the ML models. If such models handle approximation of the entire process well, we expect no significant change in performance between the pure ML and hybrid approaches. These three modelling approaches, namely the pure physics, pure ML, and hybrid physics-ML, are well summarized by Figure 5.2. Three ML models are considered altogether, namely an LSTM NN, an artificial neural network (ANN), and a Gaussian process regression model (GPR). A version of each is developed with both sets of regression features. The comparison between the GPR and shallow-NN provides a good comparison between the effectiveness of parametric and non-parametric models. Similar to equation (5.7), we can express the hybrid approach for an LSTM network with a moving window of features, although this is omitted for conciseness.

5.2.4 Rotor Speed Considerations

As indicated by equation (5.2), for full wind estimation with no state assumptions the rotor speeds are required to estimate the rotor force. In all of the ML functions that have been expressed (equations (5.6), (5.7), and (5.8)) and are to be approximated, the rotor speeds have been included as a regression feature. However, such sensors are not commonly included in off-the-shelf drone sensor packages. While custom motor speed sensors were implemented (as described in Section 5.3), it is important to consider the effectiveness of ML models that neglect this sensor source. Two variations of each model described were developed: the first including motor speed as a regression feature and the second without. This means that in total, 12 ML models were developed, 4 variations for each of the LSTM NN, ANN, and GPR models. In the hybrid approaches, the drag estimations are computed neglecting the rotor speeds in the equations of motion, assuming that there is no-vertical drag component acting on the drone,

$$F = \frac{(g+\ddot{z})m}{C_\theta C_\phi}. \tag{5.9}$$

5.3 Experimental Platform

The experimental platform consists of a modified Indro Robotics Scout Mk IIIB, based on a DJI Matrice 100 airframe, with added sensors for measuring the drone state (pictured in Figure 5.4). An RTK GPS based on a u-blox ZED-F9P receiver has been added with ground

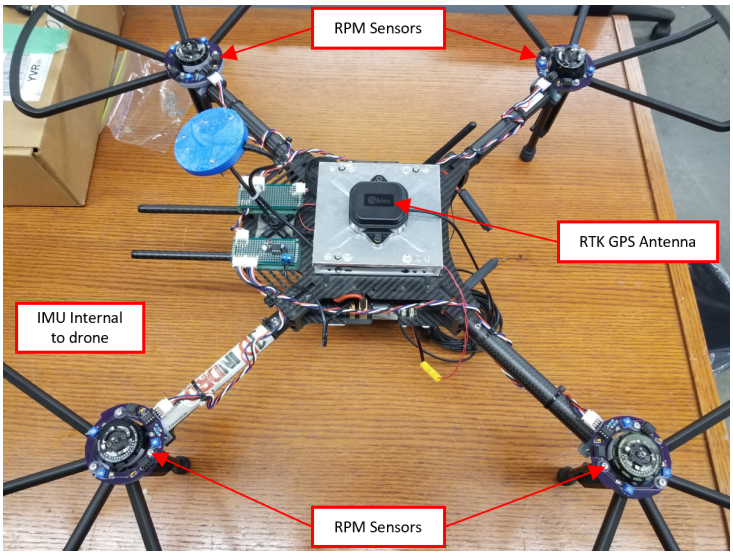


Figure 5.4: Picture of multirotor used on ground with added sensor modifications.

station for accurate reporting of ground position and velocity, achieving a reported 0.4 m and 0.1 m/s horizontal position and velocity accuracy respectively. On paper, this device is capable of achieving decimeter accuracy but this requires the ground station to be stationary for up to 24 hours previously. Custom motor speed sensors used for control input measurement are installed on all motors of the drone. Each sensor comprises of three hall effect sensors, that turn the rotating magnetic field of each rotor into a rising and falling edge signal for revolution-per-minute (RPM) calculation. Using the pulse counting method, these sensors achieve a motor speed resolution of 0.285 RPM which corresponds to approximately 0.02 N of thrust force resolution per motor at the expected flight conditions. A hall effect RPM measurement system is favoured here as opposed to optical systems, as an optical system would require good light rejection from ambient daylight in order to avoid false readings.

A 2-axis sonic anemometer located on the ground with a wind speed resolution of 0.01 m/s was used for true wind measurement. Typically, the horizontal wind components are of more interest close to the ground, but use of a 2-axis anemometer limits the scope of the training and validation to horizontal wind only. In principle however a 3-axis anemometer could be used to validate models for estimating 3-axis wind fields.

5.3.1 Signal Processing

The drone state variables are recorded by the developed flight system at rates up to 25Hz. and then downsampled to the rate of anemometer measurement of 4Hz. A major difficulty encountered however when dealing with multiple sensor sources was aligning all data sources temporally, ensuring that there was no unknown offsets in the timestamps of different sensor readings. This issue was overcome by comparing sensor signals that were expected to give the same readings, and then manually adjusting the time offset by one sample until a global minimum in error is reached between the two signals. For example, we can convert the internal-measurement-unit (IMU) measured acceleration into the global frame of reference by the orientation measurements, and then compare this signal to the global acceleration as measured by the differentiated GPS position measurements. The IMU acceleration data converted from the body frame of reference to the global frame was used as the acceleration input to the ML models.

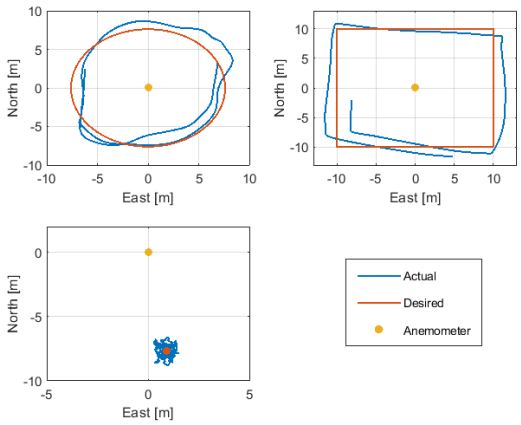


Figure 5.5: Overview of flight modes used to collect flight data for model training and validation.

5.3.2 Test Flights

Data was collected over seven days of test flights, allowing for varying wind conditions and directions. A total of 18 test flights were conducted across all days, with varying flight lengths of up to 10 minutes. Different flight paths were collected and used for training and model validation, such as hover, circular flight, and linear flight in a square pattern centered around the anemometer (Figure 5.5). The goal with these flight patterns is to reduce the error introduced by the spatial variation of the wind field, whilst avoiding the anemometer measuring the downwash of the drone itself. Generally, the drone was flown 5-10 m away from the anemometer. While this strategy hopes to reduce this error, this effect is still present and is discussed in Section 5.5. As a 2-axis sonic anemometer was used for validation purposes, each model is developed with respect to X and Y wind only and neglects validation in the vertical direction. In total, 151 minutes of test flight data was collected and used for model training, validation and testing.

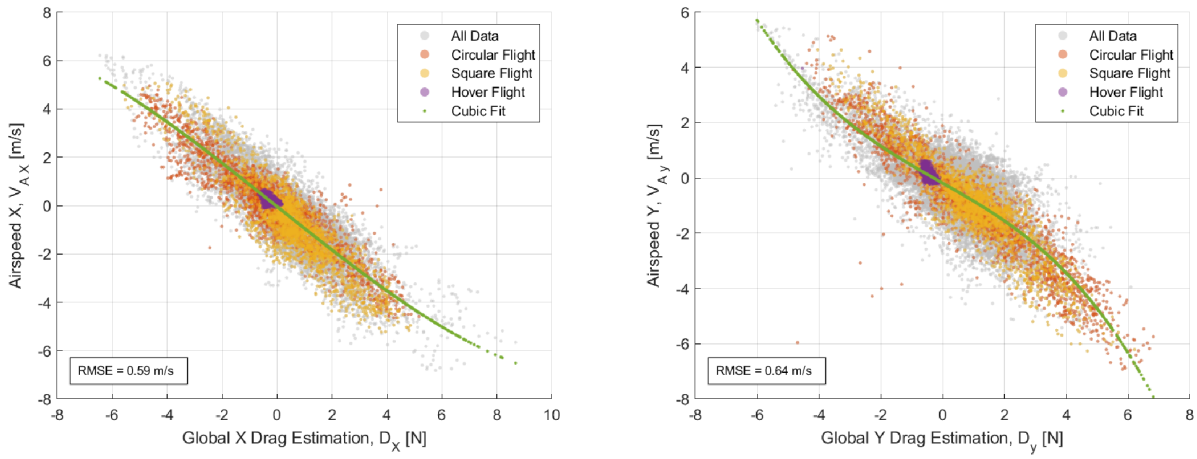


Figure 5.6: Order-3 polynomial fitting of direct drag-airspeed relationship in each of the principal directions.

5.4 Results and Analysis

This section describes the results of the previously described test flights, and is composed of four main sections. The first describes the development of the simplest wind estimation model using only independent polynomial fits intended for benchmarking performance. Each subsequent section describes the development of a different ML model and their variations starting with the LSTM NN, then the ANN and the GPR models. Each model has four variations with and without rotor speeds used as a regression feature, and whether the model is applied to the pure ML or hybrid approach. All models were trained using MATLAB.

5.4.1 Drag-Airspeed Polynomial Fitting

A third-order polynomial was used to fit the direct relationship between $D_x \rightarrow V_{ax}$ and $D_y \rightarrow V_{ay}$ using the least-squares solution. In each case the data from subsequent flight days is overlayed to indicate that the fit is consistent across trials and over-fitting is low. Over-fitting is not expected to be a significant issue here due to the simplicity of the model. The fit is seen in Figure 5.6.

5.4.2 LSTM Model Training

The LSTM network was developed in reference to the work done by Crowe [31], who applied a network to the same problem but with simplified inputs A similar structure and hyperparameters are used here, as they were found to provide good results. That is, a LSTM network was trained with two hidden layers with 90 and 48 units each, followed by a 10% dropout layer and a fully connected regression layer which is presented in Figure 5.7. The achieved performance of a NN is typically highly dependent upon the selection of optimizer and loss function. Here, we use the mean-squared-error (MSE) as the loss function, and the Adam optimizer algorithm as this was found to give good results. Each network was trained over 50 epochs and the network taken from the last iteration. In all cases this was more than enough to ensure that steady-state performance was reached. The initial learning rate was 0.001, which dropped by 80% every 10

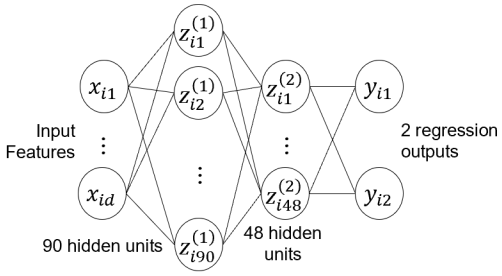


Figure 5.7: Presentation of basic ANN and LSTM structure used.

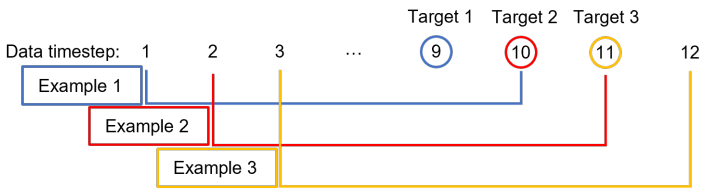


Figure 5.8: Explanation of data pre-processing required for formatting timeseries input to LSTM NN.

epochs. A mini-batch size of 10 and $n = 10$ were used and tuned by hand to give good results. n here refers to the number of previous time steps to include within each window, as shown in equation (5.7). This means that a set of $n = 10$ ordered vectors were organized as one input, for every one training example. That is, the drone state from time steps 1 to 10 is used to predict the wind at time step 9, the drone state from time steps 2 to 11 are used to predict the wind at time step 10, and so on. This is described by Figure 5.8. Special attention is payed to the transitions between discrete flights in the total dataset, as there is no meaningful relationship between the end of one flight and the beginning of the other. All regression features were normalized to account for varying magnitudes of scales. This is particularly important when incorporating motor speeds which are on the order of 4000 to 5000 RPM, compared to orientations which may be 0 - 0.2 rad. To train the network, three discrete test flights were segmented as a testing data set, which were selected by taking different but representative flight conditions (which are later summarized in Table 5.2). Then, the remaining data was used for 10-fold cross validation to determine the performance distribution of 10 differently trained models. The mean-performing model out of these was put forward as the final trained model, and is tested with the previously obscured testing dataset. Figure 5.9 shows the training and validation loss as a function of the number of iterations (epochs) for this selected model, specifically only showing the model applied to the pure ML approach with the motor speeds. The good agreement between the two lines indicates that there is little overfitting, which is supported by the consistent training, and validation errors (see Figure 5.11). In these scatterplots, the insets show an approximate distribution of the performance achieved by the 10 cross validated models.

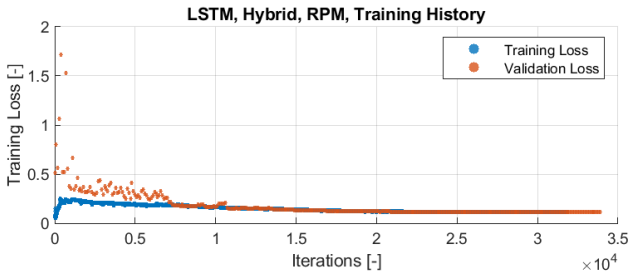


Figure 5.9: Training history of the LSTM, hybrid with RPM showing little overfitting and good agreement of training and validation loss.

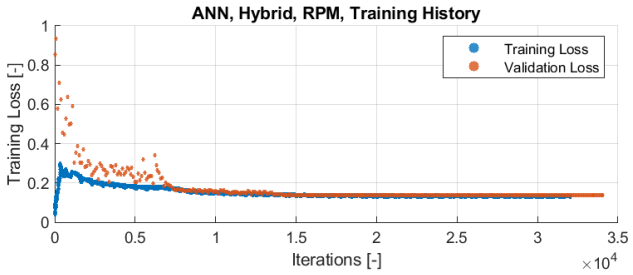


Figure 5.10: Training history of the ANN, hybrid with RPM showing little overfitting and good agreement of training and validation loss.

5.4.3 ANN Model Training

An ANN was used for their well-documented ability to fit generic non-linear functions [42]. To be consistent with the LSTM, the ANN uses the same structure presented in Figure 5.7. The same training algorithm was used: namely, the Adam optimizer with a learning rate of 0.001 dropping by 80% every 10 epochs for a total of 50 epochs. The same 10-fold cross validation technique as in the LSTM model development was used, where the mean performing model was selected as the final model and applied to the testing data. Figure 5.10 shows the training history, which is trained until the validation loss stops improving. This is shown specifically for the version using rotor speeds and the pure ML approach. Again, the scatterplots in Figure 5.12 show the distribution of the cross validated models, and the final performance of the selected model on training, validation and test datasets.

5.4.4 GPR Model

A GPR model was trained using each of the four variations, in order to provide a comparison between typical performance of parametric and non-parametric models as a GPR is a non-parametric kernel-based probabilistic model. Here, we used an rational quadratic kernel function as it was found to minimize the validation error and overfitting as compared to other functions. The quasinewton optimizer was used for parameter estimation, with a constant basis function. Since the dataset was large for each model created ($\approx$ 36326 observations), 2000 randomly selected data points were used as the active set to reduce computation time of the final model. Initial values of the kernel parameters were varied by hand over multiple orders of magnitude to ensure that for all cases the models were converging to similar performance levels, indicating that any one model wasn't converging to a local optimum. When using the same models to predict airspeed and compute wind speed using the wind triangle, they were consistently less accurate than directly estimating wind. Again, 10-fold cross validation was used in order to determine the distribution of performance achieved with varying samples of the dataset. Figure 5.13 shows the distribution of performances, as well as the training, validation and test performances of the selected model.

5.5 Performance assessment of machine-learning approaches

This section discusses the achieved performance of each of the trained models and compares them considering how the error varies in different flight phases given the data collected, limiting factors given the experimental setup, signal averaging effects and finally a wind speed distribution comparison. The root-mean-squared-error (RMSE) is used as the primary metric for comparing the accuracy of each model, as thoroughly discussed and shown in the scatter plots of Figures 5.11 to 5.13. For clarity, only the scatterplots are shown for the models trained with RPM as a regression feature and trained on the hybrid, although all model's performance is summarized in table 5.1. Each models performance is reported in the training data set, a randomly selected validation data set, and three independent testing data sets each from a test flight. These three testing flights were selected before final model training, and represent a hover flight with low wind, a hover flight with high wind, and a dynamic square-pattern (see Figure 5.5) flight with high wind. From the cross-validation steps, we are able to see the performance distribution of the 10 differently trained models, giving an initial estimation in the uncertainty of each performance metric. Without performing hyperparameter optimization, these compared machine learning models achieve around 0.31 - 0.44 m/s RMSE on randomly sampled validation data, already

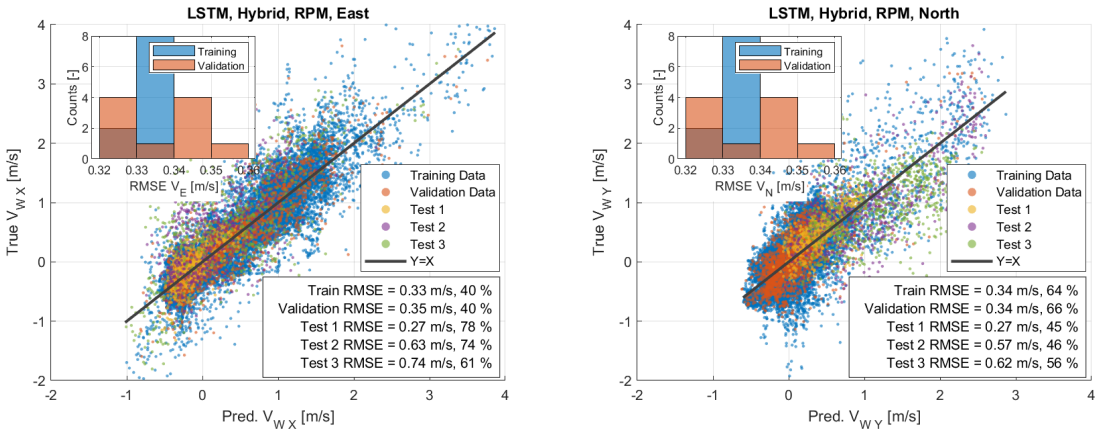


Figure 5.11: Scatterplot comparison of estimated wind by LSTM model using hybrid approach with RPM as a feature and true anemometer readings.

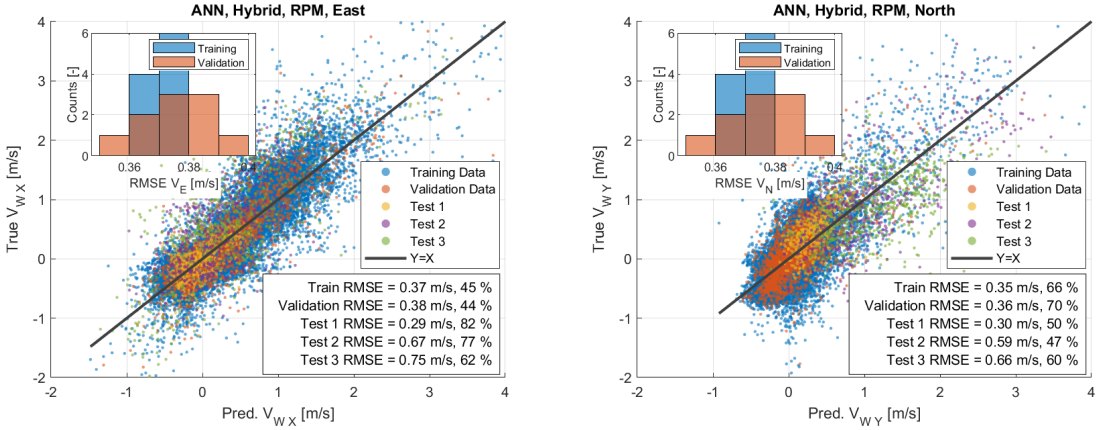


Figure 5.12: Scatterplot comparison of estimated wind by ANN model using hybrid approach with RPM as a feature and true anemometer readings.

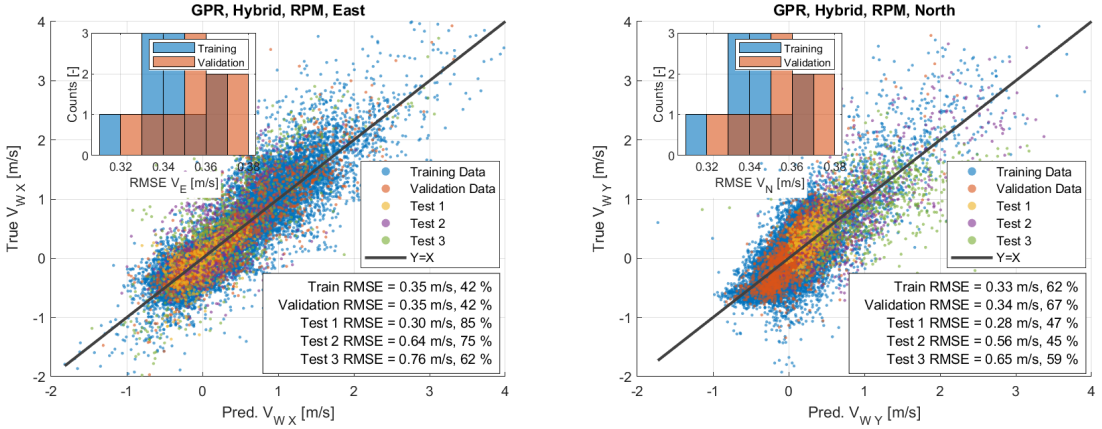


Figure 5.13: Scatterplot comparison of estimated wind by GPR model using hybrid approach with RPM as a feature and true anemometer readings.

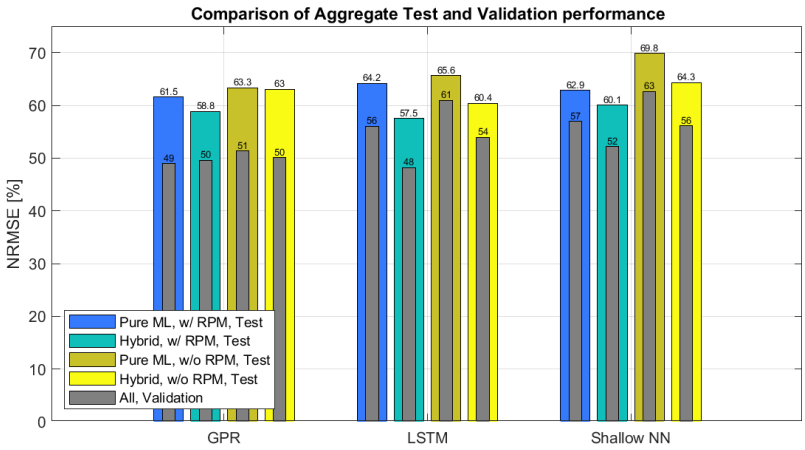


Figure 5.14: Bar plot comparison of aggregated test and validation performance metrics.

Model	East Validation RMSE	East Test 1 / 2 / 3	North Validation RMSE	North Test 1 / 2 / 3
LSTM NN, Pure ML with RPM	0.37 m/s (46%)	0.29 / 0.66 / 0.80 m/s (83% / 78% / 66%)	0.39 m/s (75%)	0.26 / 0.66 / 0.74 m/s (44% / 54% / 68%)
LSTM NN, Pure ML, RPM Omitted	0.42 m/s (49%)	0.28 / 0.58 / 0.78 m/s (81% / 68% / 64%)	0.44 m/s (85%)	0.40 / 0.75 / 0.77 m/s (67% / 60% / 70%)
LSTM NN, Hybrid with RPM	0.35 m/s (40%)	0.27 / 0.63 / 0.74 m/s (78% / 74% / 61%)	0.34 m/s (66%)	0.27 / 0.57 / 0.62 m/s (45% / 46% / 56%)
LSTM NN, Hybrid, RPM Omitted	0.34 m/s (43%)	0.27 / 0.59 / 0.73 m/s (78% / 69% / 60%)	0.39 m/s (73%)	0.38 / 0.67 / 0.66 m/s (63% / 54% / 60%)
ANN, Pure ML with RPM	0.40 m/s (51%)	0.30 / 0.70 / 0.83 m/s (85% / 82% / 68%)	0.37 m/s (69%)	0.30 / 0.58 / 0.67 m/s (49% / 47% / 61%)
ANN, Pure ML, RPM Omitted	0.43 m/s (52%)	0.30 / 0.62 / 0.79 m/s (87% / 62% / 79%)	0.44 m/s (86%)	0.45 / 0.90 / 0.73 m/s (76% / 73% / 67%)
ANN, Hybrid with RPM	0.38 m/s (44%)	0.29 / 0.67 / 0.75 m/s (82% / 77% / 62%)	0.36 m/s (70%)	0.30 / 0.59 / 0.66 m/s (50% / 47% / 60%)
ANN, Hybrid, RPM Omitted	0.40 m/s (47%)	0.29 / 0.62 / 0.76 m/s (83% / 72% / 63%)	0.39 m/s (74%)	0.41 / 0.77 / 0.65 m/s (69% / 62% / 60%)
GPR, Pure ML with RPM	0.38 m/s (44%)	0.30 / 0.66 / 0.77 m/s (86% / 77% / 63%)	0.31 m/s (60%)	0.30 / 0.62 / 0.68 m/s (51% / 50% / 62%)
GPR, Pure ML, RPM Omitted	0.40 m/s (46%)	0.31 / 0.61 / 0.75 m/s (87% / 71% / 62%)	0.32 m/s (63%)	0.37 / 0.74 / 0.69 m/s (61% / 59% / 62%)
GPR, Hybrid with RPM	0.35 m/s (42%)	0.30 / 0.64 / 0.76 m/s (85% / 75% / 62%)	0.34 m/s (67%)	0.28 / 0.56 / 0.65 m/s (47% / 45% / 59%)
GPR, Hybrid, RPM Omitted	0.36 m/s (43%)	0.30 / 0.62 / 0.80 m/s (86% / 72% / 65%)	0.34 m/s (66%)	0.35 / 0.73 / 0.63 m/s (59% / 59% / 57%)

Table 5.1: Comparison of expected and lower/upper bounds on accuracies of each trained machine learning model, including effects of adding/removing RPM sensor source.

improving upon the published state-of-the-art of 0.6 m/s RMSE by Crowe [31]. Since the RMSE is an ineffective performance metric for comparing between two different datasets, the normalized-root-mean-square-error (NRMSE) is computed by normalizing with the root-mean-square (RMS) of the true wind signal. Interestingly, the NRMSE consistently increases when going from east wind estimation validation to test performance metrics, but always decreases when going from north wind estimation validation to test. This could possibly indicate that the total data set collected is biased towards one wind direction, due to the particular wind conditions recorded at the flight testing location. To provide a concise summary of this table, Figure 5.14 provides a comparison of aggregate testing and validation performance which is

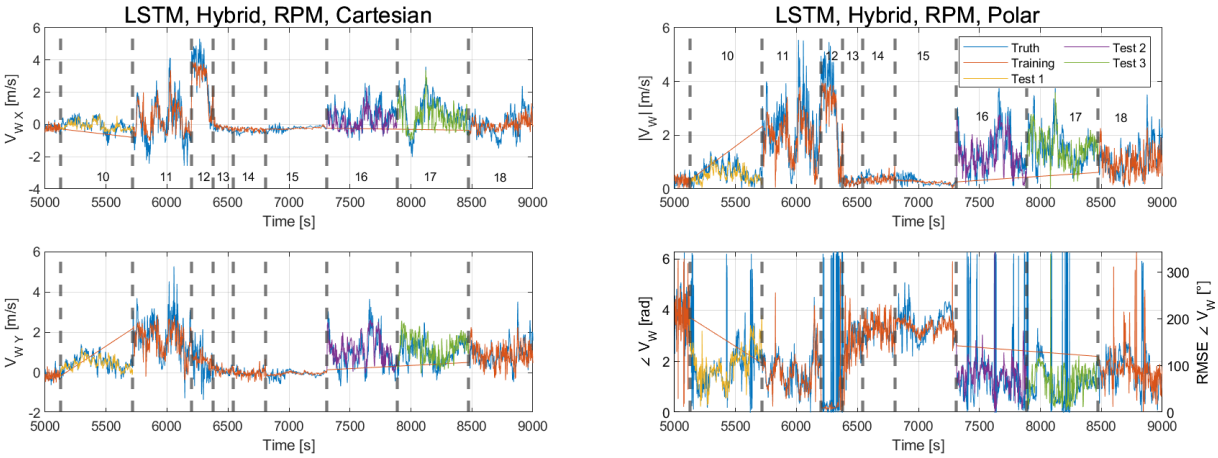


Figure 5.15: Time series comparison of estimated wind by LSTM model for hybrid approach with RPM and anemometer data in cartesian and polar coordinates.

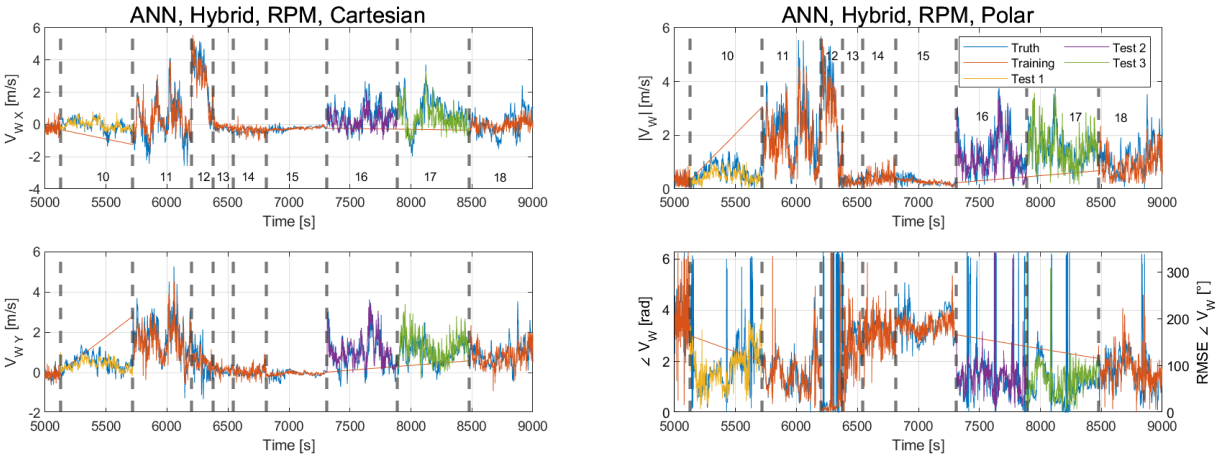


Figure 5.16: Time series comparison of estimated wind by ANN model for hybrid approach with RPM and anemometer data in cartesian and polar coordinates.

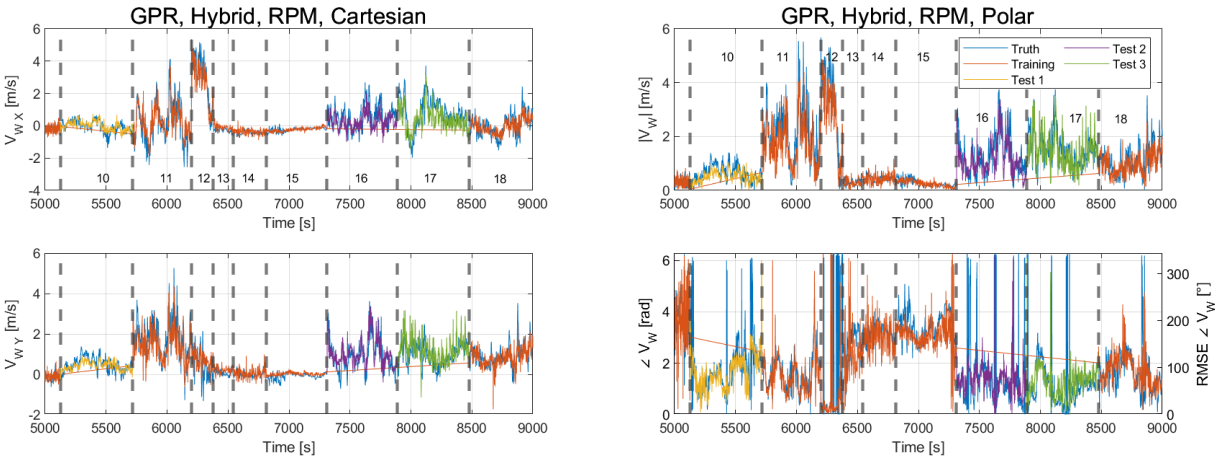


Figure 5.17: Time series comparison of estimated wind by GPR model for hybrid approach with RPM and anemometer data in cartesian and polar coordinates.

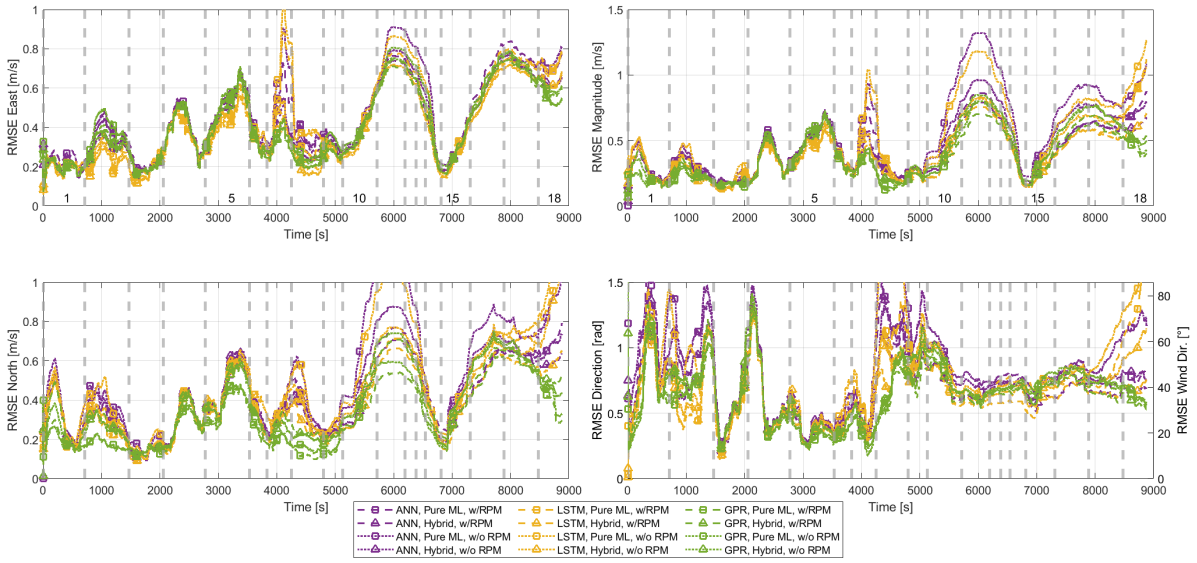


Figure 5.18: Comparison of RMSE over different flight phases for all ML models with hybrid and motor speed variations.

computed by the NRMSE of each data set appended together. In all cases, adding the drag estimations in the hybrid approach improves the model performance by up to 8% NRMSE. Adding the motor speed signal as an input still improves the estimation, but generally by not as much. The best performing model based on the dataset presented and models compared is the LSTM using the hybrid approach, however since significant hyper parameter optimization was not attempted it is difficult to generalize this result. This is further discussed in Section 5.6, considering the primary sources of error in data collection. Generally, all models achieve a similar level of performance however, and the trends observed when using the hybrid approach or using the motor speed signal hold across models.

5.5.1 Changing Flight Phases

The errors reported from the training, validation and test data sets only consider random samples from the total dataset and do not consider how the error typically varies in different flight conditions. To do so, we consider predictions on the full, unshuffled time-series data which is shown in Figures 5.15 to 5.17 for each of the three developed ML models and in the hybrid approach. These plots have been zoomed in on the portion of the data encompassing all three testing data flights. While it is generally bad practice to conduct analysis using the training data, this analysis would otherwise not be possible without re-collection of time-series testing data. The similarity between training, and validation indicates that this will still give a reasonable assessment of performance. The X and Y wind estimations are all converted into magnitude and direction, as this is a common way of expressing wind in atmospheric measurements. Each plot is segmented into portions of each discrete test flight, with labels for the flight conditions shown in Table 5.2 categorizing each flight as hover, circular or square and also showing the wind speed and ground speed ranges. Generally we can see that the overall accuracy for wind magnitude estimations generally decreases as the wind strength increases, but the direction error suffers when the wind strength is low. This is particularly evident when looking at the direction estimations between 6300 and 7300 seconds, where the true wind signal is less than 0.5 m/s.

Test Flight	Ground speed Range	Wind speed Range	V_{WE} RMSE	V_{WN} RMSE	$\lvert V_W \rvert$ RMSE	$\angle V_W$ RMSE
1,Hover	0.6 m/s	1.2 m/s	0.20 m/s	0.31 m/s	0.29 m/s	0.77 rad (44.5°)
2, Square	5.0 m/s	1.1 m/s	0.24 m/s	0.26 m/s	0.25 m/s	0.69 rad (40.0°)
3, Hover	0.9 m/s	0.8 m/s	0.17 m/s	0.13 m/s	0.17 m/s	0.42 rad (24.2°)
4, Hover	1.5 m/s	2.4 m/s	0.36 m/s	0.30 m/s	0.33 m/s	0.69 rad (39.8°)
5, Square	2.2 m/s	4.2 m/s	0.44 m/s	0.48 m/s	0.45 m/s	0.39 rad (22.7°)
6, Square	2.9 m/s	2.0 m/s	0.30 m/s	0.30 m/s	0.33 m/s	0.39 rad (22.8°)
7, Hover	1.3 m/s	3.2 m/s	0.37 m/s	0.30 m/s	0.37 m/s	0.44 rad (25.3°)
8, Circular	3.4 m/s	0.9 m/s	0.20 m/s	0.30 m/s	0.23 m/s	0.85 rad (48.7°)
9, Square	4.3 m/s	1.1 m/s	0.27 m/s	0.25 m/s	0.22 m/s	0.89 rad (51.5°)
10, Hover (Test 1)	0.6 m/s	1.4 m/s	0.27 m/s	0.26 m/s	0.25 m/s	0.75 rad (43.2°)
11, Hover	1.7 m/s	5.4 m/s	0.77 m/s	0.75 m/s	0.83 m/s	0.42 rad (24.3°)
12, Square	2.5 m/s	5.6 m/s	0.78 m/s	0.65 m/s	0.80 m/s	0.45 rad (25.9°)
13, Square	3.6 m/s	1.5 m/s	0.20 m/s	0.20 m/s	0.16 m/s	0.92 rad (52.8°)
14, Square	6.0 m/s	0.6 m/s	0.18 m/s	0.18 m/s	0.18 m/s	0.46 rad (26.4°)
15, Hover	3.9 m/s	0.7 m/s	0.10 m/s	0.12 m/s	0.10 m/s	0.48 rad (27.4°)
16, Hover (Test 2)	1.3 m/s	4.1 m/s	0.63 m/s	0.57 m/s	0.54 m/s	0.77 rad (44.4°)
17, Square (Test 3)	5.6 m/s	4.0 m/s	0.74 m/s	0.61 m/s	0.54 m/s	0.60 rad (34.5°)
18, Circular	5.8 m/s	3.5 m/s	0.61 m/s	0.60 m/s	0.64 m/s	0.70 rad (40.5°)

Table 5.2: Summary of LSTM model performance with RPM signals and the hybrid approach, over all test flights and operating conditions.

Here, typical errors in cartesian coordinates become very large errors in direction estimation. These effects are more clearly presented in Figure 5.18, which was produced by computing the RMSE as it varied using a 200 second moving average filter for each of the 12 models. This plot should not be used to assess the relative performance of each model as it includes training data, but can be used to effectively show in what wind ranges and flight conditions the models are more accurate. The error in the wind magnitude estimation is highest around 6000 seconds when the wind signal is strong, while the opposite is true of the wind direction estimation. These effects for each discrete test flight are summarized in Table 5.2 for the best performing model. We take this to be the LSTM model using the motor speed signal and the hybrid approach, as established by the testing datasets and comparison in Figure 5.14. In our test flights with high wind, up to 5 m/s, the RMSE for wind magnitude estimation may get as high as 0.6 m/s. In low wind cases, less than 1 m/s, the RMSE can reach 0.2 m/s.

5.5.2 Limiting Factors

Rather than continued optimization of these models through hyperparameter tuning, we consider possible limits in performance due to experimental errors. One possible limiting factor in the experimental setup is the spatial variability of the wind field. This would cause a difference between the wind as measured by the anemometer, and that experienced by the drone at its location. To estimate the order of magnitude of this effect, we consider a convection model that uses the position of the drone relative to the anemometer. When the drone is upwind of the anemometer, the wind gusts at the anemometer have travelled through the drone location before reaching the anemometer and thus the readings must be shifted backwards in time by that difference. Similarly, when the drone is downwind wind gusts reach the anemometer before the drone and the readings must be shifted forward in time by that difference. This convection model for estimating the wind at the drone location is summarized by equation (5.10). Here, p_D and p_A represent the position of the drone and anemometer respectively at time step t. Due to the effects of turbulence and changing wind directions, this model is not used to directly estimate the true wind at the drone location or for training the ML models. However, it does provide a reasonable estimate of the typical variation in wind between these two locations and

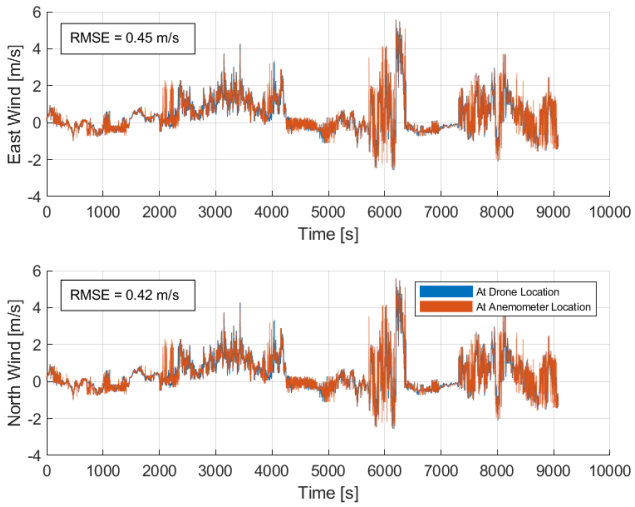


Figure 5.19: Comparison of anemometer readings and estimated wind at drones location using convection model.

can be used to state whether this significantly contributes to the limit in performance achieved by the models. Comparing the anemometer wind readings to the estimated wind at the drones location yields an RMSE of 0.45 m/s in east and 0.42 m/s in north (see Figure 5.19), both of which are similar in magnitude to the achieved model performance on the entire data set. This indicates that this effect is likely the limiting factor in the accuracy achievable for wind estimation using this experimental approach with only one sonic anemometer.

$$\begin{aligned} \Delta t_E &= (p_{DE} - p_{AE})/V_{AwE}, \\ \Delta t_N &= (p_{DN} - p_{AN})/V_{AwN}, \\ V_{DWE}(t) &= V_{AWE}(t + \Delta t_E), \\ V_{DWE}(t) &= V_{AWN}(t + \Delta t_E). \end{aligned} \tag{5.10}$$

5.5.3 Averaging Effects

Generally, for signals with Gaussian sensor or process noise applying a moving average filter will increase the overall accuracy but decrease the sensor bandwidth and response time. Figure 5.20 considers this, and was generated by applying moving average filters of varying lengths to both the anemometer readings and model estimations. Neumann's published results [23] and Crowe's [31] are included for comparison to show the improvements made relative previously developed works. The most significant improvements are made to wind magnitude estimation, where the developed LSTM model reaches approximately 0.35 m/s RMSE relative to 0.6 and 1 m/s achieved by Crowe and Neumann respectively at a 1 second moving average filter. At a glace, no improvements are made by these models in wind direction estimation and perform approximately to the same level of accuracy as Neumann's work. However, a limiting factor in our wind direction estimations are including the direction estimations for low (less than 0.5 m/s) wind speeds. Neumanns results were based on wind estimations exclusively between 1 and 5 m/s, neglecting this low-wind inaccuracy. The convection model is included in this plot representing an initial estimate of the variation in the spatial wind field, indicating that this

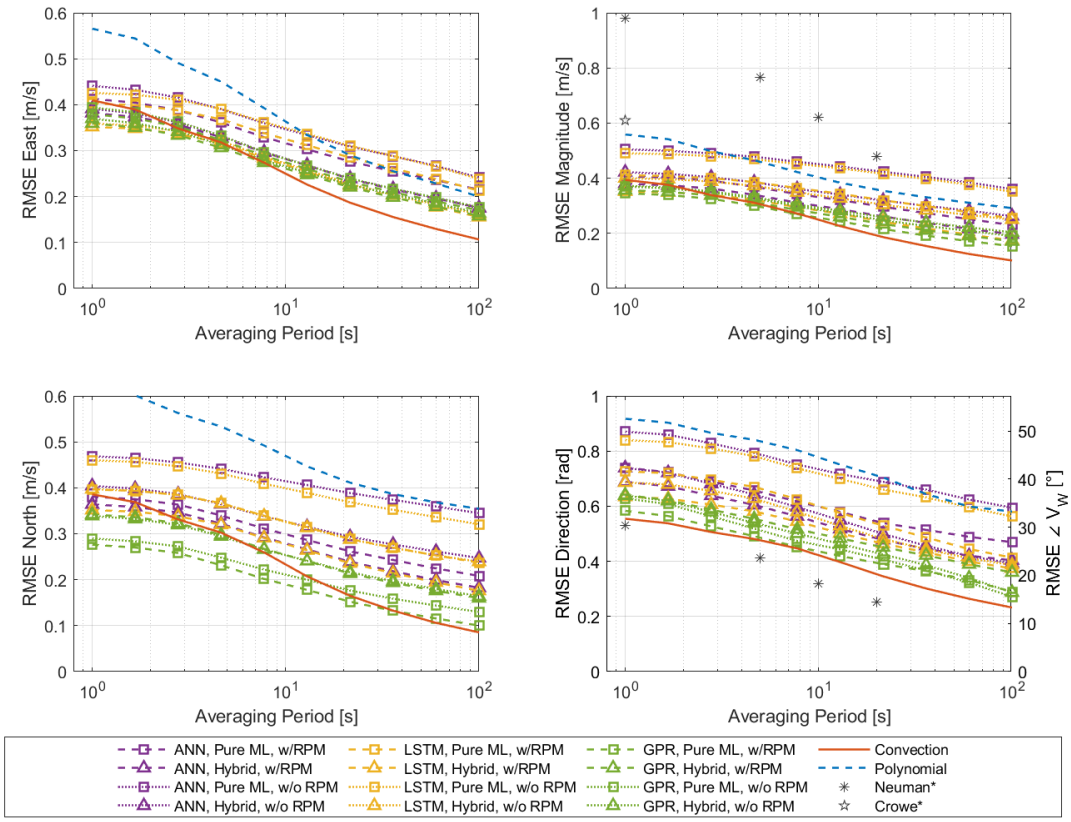


Figure 5.20: Summary of multiple model RMSE comparison with changing moving average period on testing data only, including published results (Neumann, [22], Crowe, [31]).

is the leading cause of limited performance in the wind estimation models as it is comparable to the best performing models. We can see that for long averaging times, the effects of adding drag estimations and motor speeds stay as the difference between all models is consistent. This analysis can be used to assess wind estimation by quadrotor dynamics when applied to use-cases with varying time scales of interest. When using this method for applications only interested in long-time scales (on the order of multiple minutes), the LSTM model achieves around 0.15 m/s and 0.27 rad (15.5°) RMSE in wind magnitude and direction respectively.

5.5.4 Wind speed Distribution Comparison

For wind surveying applications, an important factor to estimate is the overall wind speed distribution as well as the turbulence at a given flight location. For these results, we compare the computed turbulence for each individual test flight and the distribution of all flight data for the hybrid LSTM model and the anemometer readings. This effectively gives an average reading for the test flight location and within the flight limits, however this procedure could be applied to a finite spatial resolution for detailed wind survey studies with enough data. To compare turbulence, the variance in each principal direction was computed for each signal. For the comparison, a Weibull distribution was fit to each and compared [12]. Figure 5.21 shows good agreement between the wind speed distributions as well as the turbulence characterization.

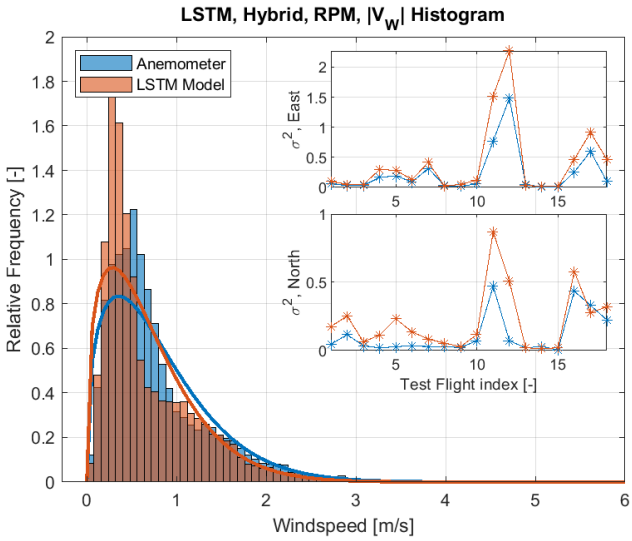


Figure 5.21: Comparison of LSTM and anemometer wind speed distributions and turbulence in east and north principal directions.

While the wind does not seem to follow a Weibull distribution particularly well for wind speeds less than about 1 m/s, the distribution created by the LSTM model is a good approximation of the anemometer readings. The turbulence is also well-characterized by this model.

5.5.5 Execution Time Performance

When considering application of these methods to real-time systems, execution time is a strong requirement where it is highly desirable to have fast executing predictions. Thus, we briefly compare the execution time of the final trained models on the entire data set as this gives insights into which methods are favourable for real-time systems as opposed to post-processing. Table 6.2 shows this comparison, where the ANN by far computes the fastest and the GPR the slowest. This is expected as the GPR uses a table of training data and uses inference to make predictions, making this a much slower and memory-intensive procedure. Additionally, a GPR typically only predicts a scalar rather than multi-dimensional regression meaning that the execution time is twice that it would usually take.

Model	LSTM	ANN	GPR
Total Execution Time	3.14 s	0.43 s	19.99 s
Time per example	1.29e-4 s	1.75e-5 s	8.17e-4 s
Ratio to Fastest	7.4	1.0	46.8

Table 5.3: Comparison of model execution time. Each model presented for the hybrid approach with motor speeds.

5.6 Discussion

While this work has been completed in reference to 2-axis wind estimation in the horizontal plane alone, it can in principle be extended to 3-axis wind estimation. Referencing equation (5.7), the vertical wind component is included in these approximations but neglected due to our use of a

2-axis anemometer in the experimental setup. While this is feasible, it in general has a different set of experimental implications. When measuring the drone state, rotor speed measurement becomes more important. Considering a multirotor hovering at constant altitude with changing vertical updrafts and downdrafts, the rotors must increase and decrease their speed to change the thrust while keeping all other state variables constant. It is expected however that the motor speed measurement resolution achieved by the previously presented custom RPM sensors should be sufficient for such an application. When comparing the measured drone state with 3-axis anemometer readings, the concern of spatially varying wind becomes more important as turbulent eddies that have high spatial variability dominate the vertical wind gusts close to the ground.

We have shown that including the drag estimations in the hybrid approach improves the accuracy of all models by the most, typically improving the NRMSE by $\sim$ 7% in the case of the LSTM (as normalized by the RMS of the true signal). From this we can suggest that by studying the equations of motion of the drone and utilizing the well-known physics, we expect the models performance to increase. Overall, the LSTM model with the hybrid approach and motor speed signals performed the best reaching 48% NRMSE aggregate validation on randomly selected data and 57.5% NRMSE aggregate testing on full test flights. This is fairly close to the performance of other models such a the GPR with the same inputs, reaching 50% and 58.8% validation and test NRMSE respectively. Although it is common practice in ML to randomly select validation and testing data, this generally gives a measure of how well the model interpolates between the data points it has previously seen. In our case however these models all perform much better on the randomly selected validation data than the new test flights, which is how the system would be used in practice. This possibly indicates that the test flights are different in nature to the training data, and that the training data should be specifically chosen to give a representation of the underlying physics. In principle, the equations of motion described by (5.6) is a closed form instantaneous relationship, requiring no previous time steps for input with the spatial derivatives such as ground velocity and acceleration pre-specified. In which case we expect the LSTM to achieve the same performance as the GPR and ANN, which is not true. We suspect that the LSTM is able to form improved state estimations, through a state filtering effect with multiple time steps of the drone state.

When considering applying this work to other multirotor drones or application areas, it is important to consider the reported accuracy and how that may change with the selection of wind speed operating range and the addition of motor speed sensors. We have shown the added motor speed sensor to increase the accuracy of the wind estimations, with the trade-off of the added complexity to the sensor suite. The motor speed sensors presented are low-cost devices however, utilizing off-the-shelf electronics components such as hall effect sensors, and microcontrollers on a custom printed circuit board (PCB). In this work, the largest wind experienced by the drone was 5-6 m/s which was well within the datasheet maximum allowable wind speed of 10 m/s for the drone used. In large scale wind farming applications, wind speeds may reach as high as 10-20 m/s, requiring a drone that is able to withstand such winds. There may as well be an increase with the error in wind estimation with the strength of the wind signal as indicated by these results. More experiments with a more wind resistant drone would be required to fully investigate implications in this increased wind measurement range.

As previously mentioned in Section 5.3, considerable effort was spent temporally aligning multiple sensor sources, as it was unknown if the timestamps from multiple devices had small time inaccuracies or offsets. In future work, implementing a singular data acquisition system that is able to record all flight variables of interest is recommended, as this will overcome the uncertainty of unknown time offsets between different sensor sources.

5.7 Conclusions and Future Work

In this paper, we have presented physics and machine learning models leading to an improvement upon the published performance of wind estimation by quadrotor dynamics. Three different structures of ML models were developed, which were trained on the drone state both with and without motor speeds as an input and using two separate sets of regression features. An LSTM was developed that used a time-series prediction moving window of the 10 previous data points to predict the wind. An ANN and GPR model predicted the wind speed by assuming an instantaneous relationship between the drone state and the wind. The first set of regression features predicted the wind directly from the drone state measurements, without any preprocessing or added features. The second set included the drag estimations from the drone state as computed by the developed non-linear equations of motion of a rigid quadrotor drone. The models performance was computed by a number of metrics: the RMSE of a validation set which was randomly selected, and the RMSE on three separate test flights. In general, adding the drag input features in the hybrid approach improved the accuracy of all models. Adding the motor speed signals additionally helped, but by not as much as the drag estimations. Generally selecting the training data and input features to represent the well-understood physics is expected to improve performance, but this left for future work.

The LSTM model was found to perform the best, but with all models reaching a similar level of performance. For all models however, adding the drag estimations improved performance indicating that applying machine learning to the entire physical process in a black box fashion may not be optimal for this application. The performance did change when omitting the rotor speeds, indicating that there is a trade-off between hardware complexity and wind estimation accuracy when applying this work to other drones. In the case of the LSTM, adding motor speeds typically improved the 5-6% for validation data and 2-3% for testing data. When applying this approach to other drones, one must consider the added complexity and cost of including the additional motor speed sensors relative to the performance improvements and the requirements of a given application.

The primary source of error in this work is likely the use of a single sonic anemometer, that is assumed to measure a locally spatially uniform wind field. The convection model developed indicates that possible errors in this assumption are larger than the achieved accuracy by ML models. This, as well as the relatively close distribution of performance metrics across multiple types of machine learning models indicates that these models are likely approaching the limits of performance given this setup. For future work, it is recommended that an improved validation method be considered such as using multiple sonic anemometers on the ground that would be able to provide a weighted estimate of the wind at the drone location. It is expected that improvements here will prove more effective than continued hyperparameter optimization or model tuning.

This work is applicable to a number of fields, as it allows for wind estimation at the spatial drone location and at all times during a given flight. The models that omit the motor speeds as a regression input are the most versatile as they use sensors common in drone hardware packages, not requiring any custom hardware implementations. While this work utilized a RTK GPS solution which is an improvement to the native GPS included in the base drone used, such a sensor becomes more and more common as drone technology improves and the price of the drone increases as well. Generally this work has great advantages in applications such as energy-efficient path planning, flux estimation, wind farm surveys and others. In flux estimation from a given source, no spatial averaging based on anemometer readings are required. The airspeed can be measured and coupled with concentration readings in order to accurately integrate the

flux over a given area.

Chapter 6

Wind Estimation by Multirotor Drone State using Machine Learning with Data Rotation and Reduction

This chapter contains a version of a journal article submitted for review. The primary author is Steven Zimmerman, with co-authors Dr. Ryozo Nagamune, and Dr. Steven Rogak. This manuscript references the previous manuscript and Chapter *Wind Estimation by Multirotor Drone Dynamic State Measurement and Machine Learning Models* as [40].

6.1 Introduction

Measuring a locally varying wind field is important in a number of fields and applications, such as environmental monitoring, wind farming, control systems engineering, and others. In environmental studies, wind measurement can be coupled with gas concentration to produce flux emissions measurements. This can be used to quantify the emissions rate of known sources or to find unknown sources. Many papers [5, 43, 3] have presented results on this, typically relying on a mobile robotics platform to sample the concentration and measuring the wind by a static anemometer with a spatial wind uniformity assumption. A review is provided by Shaw [6]. Knowledge of the changing wind field at the location of gas measurement provides significant benefits to these fields. In wind farming applications, wind measurement can be used to inform wind turbine placement, design, and control [11, 10]. This is particularly important as the wind field around complex terrain or buildings is not easily modelled. In control systems engineering, flying robotics platforms that are susceptible to wind can benefit from real-time estimation of the disturbance wind field [14, 15]. Control algorithms can be created to modify a desired trajectory in order to take advantage of the wind field for higher flight efficiency, or to reject this disturbance. Each of these applications requires a method of experimentally measuring the wind field on a local scale, with high positional accuracy.

Some common methods of measuring the wind field include small, fixed wing aircraft or static anemometers on large booms, but both of these have their own trade-offs. These methods represent two ends of a spectrum, where the static anemometer measures detailed information about one location and a fixed wing aircraft measures coarse information about a larger number of locations due to their fast movement speed. Multirotor drones offer an intermediate trade-off in this spectrum, as they have high positional versatility and control. They can be deployed at a variety of sites to conduct surveys, and typically have the payload capacity for a variety of sensors. Thus, they offer a very attractive option for wind measurement, but this suffers when attaching an anemometer as a payload. The downwash from the propellers creates local airflow over an anemometer, disturbing the measurement of the free-stream wind. It is possible to attach an anemometer on a boom away from the propellers [44], but this adds weight and flight control penalties. A method to overcome this established in literature is by implementing a

disturbance observer that can estimate the wind implicitly from the multirotor state and control inputs, eliminating the need for a dedicated anemometer.

Neumann [22] presented the pioneering work on this problem, demonstrating a disturbance observer based on a simplified physics representation of a multirotor. This work used a static assumption, directly relating the orientation to the wind magnitude and direction through wind tunnel testing. However this work does not generalize well to dynamic flight phases. Allison [29] improved upon these results by applying a machine learning (ML) long-short-term-memory (LSTM) neural-network (NN) model to a simulated multirotor predicting Dryden wind based on orientation alone. Wang [30] conducted an experimental study with indoor fan wind and an ML K-nearest-neighbours algorithm to estimate airspeed. Crowe [31] demonstrated both of these models experimentally in the field, applying a similar LSTM model to a real drone with orientation and acceleration as inputs in hovering conditions. We studied this problem experimentally in [40], expanding the number of inputs to include the full drone state and control inputs by measuring the rotor speeds in flight. It was shown that accuracy improved when utilizing a hybrid approach, that utilized drag estimations from known equations of motion as an additional model input. However there was a decrease in performance between randomly selected and complete flight performance metrics, indicating a lack of generalizability.

In this paper, we improve upon the drawbacks of previous of previous work by considering the effects of data rotation and reduction to improve generalizability. Data rotation enforces rotational invariance in the trained ML models, while data reduction eliminates data imbalance issues and decreases the number of training examples needed to sufficiently train a model.

This paper is organized as follows: first the previous work is reviewed in Section 6.2, explaining the theory and experiment which are consistent with this work and explaining the drawbacks of previous methods. Section 6.3 explains how rotation and reduction are applied, and how the training, validation and testing data sets are selected. The models are trained, and performance compared in Section 6.5. Finally in Sections 6.6 and 6.7, the paper concludes with a discussion of significance and future work.

6.2 Review of Previous Work

In [40], we provided a derivation of the theoretical relationship from the drone state to the wind. As this work is a direct improvement to that work, it is concisely summarized here. The main drawbacks of this previous work are addressed, which is the primary motivation for developing data rotation and reduction procedures.

6.2.1 Wind Estimation Problem

An understanding of the physical relationships between drone state and wind is useful in developing data augmentation and reduction procedures. The following relationship is derived in the coordinate frames established by Figure 6.1, showing the difference between the global and body reference frames. The relationship between drag and drone state is

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ F \end{bmatrix} - \begin{bmatrix} D_x \\ D_y \\ D_z \end{bmatrix}, \tag{6.1}$$

where m [kg] is the mass of the drone, g [m/s^2] is the acceleration due to gravity, and F [N] is the total thrust from the propellers which is a quadratic function of the rotor speeds [33], denoted by $[\omega_1, \omega_2, \omega_3, \omega_4]'$ [rad/s]. $\vec{D} = [D_x, D_y, D_z]'$ [N] is the drag force acting on the drone due to

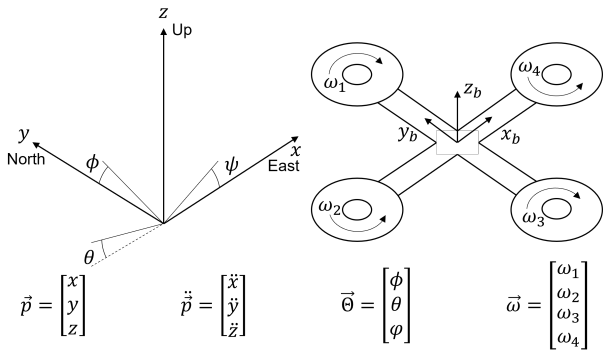


Figure 6.1: Coordinate system used for all wind estimation. The global reference frame is shown on the left, with the body reference frame of the drone which moves and rotates with respect to the global reference frame on the right. Modified from [40].

airspeed. R is the transformation matrix from the body frame to the global frame of reference and is a function of the orientation of the drone, which is omitted for brevity but described in [33]. The drones position in global coordinates is represented by $\vec{p} = [x, y, z]'$ [m] as shown in Figure 6.1, meaning that $\ddot{\vec{p}} = [\ddot{x}, \ddot{y}, \ddot{z}]'$ [m/s^2] is the acceleration in global coordinates. It is important to note that this equation is derived in the global frame of reference, which is why the rotor force, F, which acts in the body frame is translated to the global frame by R. By rearranging (6.1), the disturbance drag can be estimated by full knowledge of the drone state, namely by measuring the total thrust (or rotor speeds), the acceleration, and the orientation of the drone. The drag is then related to the airspeed by a drag coefficient relationship. However, this requires knowledge of how the drag coefficient area product varies with drone orientation in three dimensions. This plant parameter is not trivial to obtain, which is the primary motivation for applying ML to this problem. Finally, wind is estimated by vector subtraction, which is known as the wind triangle.

Therefore, it is possible to relate the drone state directly to wind estimation with appropriate measurement of the drone's orientation ($\vec{\Theta} = [\phi, \theta, \psi]'$ [rad]), acceleration ($\ddot{\vec{p}} = [\ddot{x}, \ddot{y}, \ddot{z}]'$), rotor speeds ($\vec{\omega} = [\omega_1, \omega_2, \omega_3, \omega_4]'$) and ground speed ($\dot{\vec{p}} = [\dot{x}, \dot{y}, \dot{z}]'$ [m/s]). Summarizing this entire relationship as a function, g_{ML}, to be approximated by ML models yields

$$\hat{\vec{V}}_W(\vec{p}, t) = g_{ML}\left(\dot{\vec{p}}(t), \ddot{\vec{p}}(t), \vec{\Theta}(t), \vec{\omega}(t)\right), \tag{6.2}$$

which is an instantaneous relationship at time t. It was shown in [40] that model accuracy improved when utilizing a hybrid physics-ML approach, which included the additional inputs of drag estimations which are computed by (6.1) directly. Thus the modified relationship to be approximated is

$$\hat{\vec{V}}_W(\vec{p}, t) = g_{Hybrid}\left(\vec{D}(t), \dot{\vec{p}}(t), \ddot{\vec{p}}(t), \vec{\Theta}(t), \vec{\omega}(t)\right). \tag{6.3}$$

All of [29, 31, 40] demonstrated the effectiveness of time series based models, applying an LSTM for wind estimation which requires the inputs be a set of ordered vectors for one wind estimation. A GRU model also requires this input. Expressing this instantaneous relationship

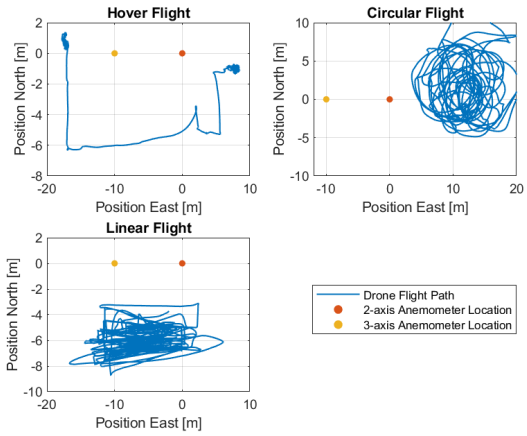


Figure 6.2: Typical flight paths used for data collection, relative to the 2-axis and 3-axis anemometers.

in a form incorporating previous time steps gives

$$\begin{aligned}\hat{\vec{V}}_W(\vec{p}, T-1) = & g_{LSTM} = g_{GRU}\big(\\ & \vec{D}(T), \ldots \vec{D}(T-n), \\ & \dot{\vec{p}}(T), \ldots \dot{\vec{p}}(T-n), \\ & \ddot{\vec{p}}(T), \ldots \ddot{\vec{p}}(T-n), \\ & \vec{\Theta}(T), \ldots \vec{\theta}(T-n), \\ & \vec{\omega}(T), \ldots \vec{\omega}(T-n)\big),\end{aligned} \tag{6.4}$$

where $T-1$ is the time step at which wind is being estimated, and n is the number of time steps used for each prediction which is taken to be 10 as per Allison [29]. That is, the data is processed in a way that 10 ordered vectors of inputs are used for one training example. Since we are only interested in estimating wind in the horizontal plane, we take $\hat{\vec{V}}_W = [\hat{V}_{WX}\hat{V}_{WY}]'$, which estimates wind in Cartesian coordinates. Cartesian coordinates are preferred to polar, to avoid the discontinuity in direction.

6.2.2 Experiment

The experimental platform is largely the same as used in [40], but is summarized here. The primary difference is the use of a second anemometer. To collect flight data, a modified quadrotor IndroRobotics Scout Mk IIIB was used with added sensors. The internal IMU was used to sample acceleration ($\ddot{\vec{p}}$) and orientation ($\vec{\Theta}$), utilizing the onboard dataflash logs to sample at rates of 25 Hz and 10 Hz respectively. An real-time-kinematic (RTK) ground-positioning-system (GPS) was added as an additional sensor, which provided more accurate ground speed ($\dot{\vec{p}}$) and position ($\vec{p}$) measurements. This utilized a GPS receiver ground station, which sent correction messages GPS receiver on the drone over radio. Custom motor speed sensors based on hall effect sensors on a ring printed-circuit-board (PCB) were used for each motor, to measure the rotor speeds ($\vec{\omega}$) and to compute the thrust (F).

Two ultrasonic anemometers, namely a Gill Instruments WindSonic 60 and WindMaster, are used to measure the true wind speed ($\vec{V_W}$); the first being a 2-axis anemometer and the second

Figure 6.3: Picture of flying drone in hover near the 2-axis anemometer used.

a 3-axis. Both instruments have a measurement resolution of 0.01 m/s, and an accuracy of ±2% @ 12 m/s. They use a time-of-flight sensing principle that eliminates temperature effects on the speed of sound. The 3-axis anemometer was placed approximately 10 m due west of the 2-axis anemometer. Two anemometers are used here, as they give an estimate of the spatial variation of the wind, which in turn acts as the limiting factor of the wind estimation accuracy. This is further discussed in Section 6.5.3. The 2-axis anemometer was set to sample at its maximum rate of 4 Hz, and the 3-axis 10 Hz but was down sampled in post processing to match the 2-axis measurements.

Data was collected over seven days of test flights, allowing for varying wind conditions and directions. A total of 18 test flights were conducted across all days, with varying flight lengths of up to 10 minutes. In total, 151 minutes of test flight data was collected and used for model training, validation and testing. Different flight paths were collected and used for training and model validation, with common flight paths shown in Figure 6.2. An example picture of the drone in flight is shown in Figure 6.3. In each of these flights, the distance between the drone position and the 2-axis anemometer was kept comparable to the distance between the two anemometers. The goal with these flight paths is to keep the differences in wind from the anemometers location to the drones location low, whilst avoiding the anemometer measuring the downwash of the drone itself. This hopes to reduce the error introduced by the spatial variation of the wind field.

6.2.3 Previous Results

In [40], we trained a variety of ML models, the best of which was the LSTM using the hybrid physics-ML approach which achieved 0.34 m/s root-mean-square-error (RMSE) or 48% normalized-root-mean-square-error (NRMSE) on randomly selected training data and 0.55 m/s RMSE or 57.5% NRMSE on complete, unseen test flights. The NRMSE was normalized by dividing by the root-mean-square (RMS) of the true wind signal, which accounts for varying magnitude of signals. The primary drawback of this work is this decrease in performance when moving from randomly selected data to new complete flights, which is how the models would be used in reality. Two issues are identified that likely cause this drawback. One issue is the lack of rotational invariance considerations in approximating a rotationally symmetric set of equations,

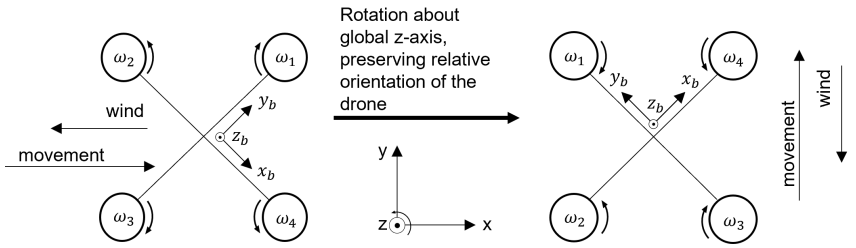


Figure 6.4: Illustration of rotational invariance property, which applies to the global reference frame but not the body reference frame.

and the other is data imbalance issues in the original training data set. These issues are detailed in the following subsections.

Rotational Invariance

The equations of motion are inherently invariant to rotations of the global reference frame, but the ML models are not. That is, the equations of motion work in any global coordinate frame. Since we are concerned with horizontal wind estimation only, we are only interested in implementing rotational invariance about the global Z-axis, which is aligned with gravity. Intuitively this feature is understood by thinking about a drone moving at a given direction with respect to the wind direction, which is shown in Figure 6.4. Physically, there is nothing different about this system when rotating by a given angle about the global Z-axis, which preserves the relative angle of the wind to the drone orientation. This effect does not however apply to the body frame of reference, which is fixed to the drone frame. We cannot expect to rotate the wind but not the drone and expect the system to behave the same, due to the different drag coefficient characteristics in different orientations. Considerations of rotational invariance are not included in previous work.

Data Imbalance

As demonstrated by (6.1), a compact way of representing the drone state which accounts for acceleration, orientation and rotor speeds is the drag estimation which is computed using the inverse equations of motion. We convert this global drag estimation to the body frame of reference using

$$\begin{bmatrix} D_{bx} \\ D_{by} \\ D_{bz} \end{bmatrix} = R^{-1} \begin{bmatrix} D_x \\ D_y \\ D_z \end{bmatrix}, \tag{6.5}$$

where $\vec{D_b} = [D_{bx}, D_{by}, D_{bz}]$ represents the drag estimate in the body frame. This is used to plot a two-dimensional histogram of the drone state (without true wind labels) in Figure 6.5. This shows an imbalance in the data set favouring low drag conditions, improperly weighting the training examples. The reason for this imbalance comes from practical flight planning. Naturally, the pilot will spend more time loitering and hovering for safety reasons than in dynamic flight phases.

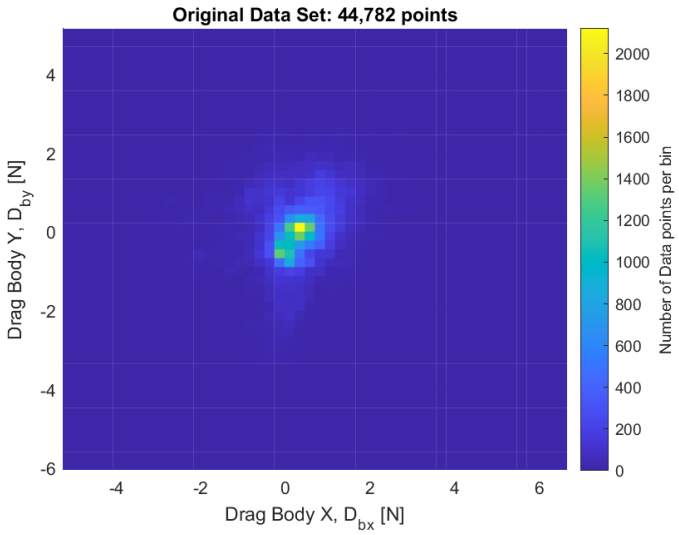


Figure 6.5: Histogram of representation of the drone state, from original data set which was used for training in [40].

6.3 Data Processing

To combat these two identified issues, we consider two data processing approaches (namely data augmentation by rotation and reduction) in order to select appropriate training, validation and testing data sets for the ML models. Subsection 6.3.1 describes the procedures used to augment the data, by considering invariance about rotations in the global Z-axis. Subsection 6.3.2 describes how this rotated data set is reduced to apply appropriate weighting and to reduce train time. Finally, subsection 6.3.3 summarizes these steps and explicitly states the selection of training, validation and testing data sets for each model.

6.3.1 Data Rotation

The previously described rotational invariance property is not inherent to ML models, and must either be learned through sufficient training examples or enforced in model implementation [45].

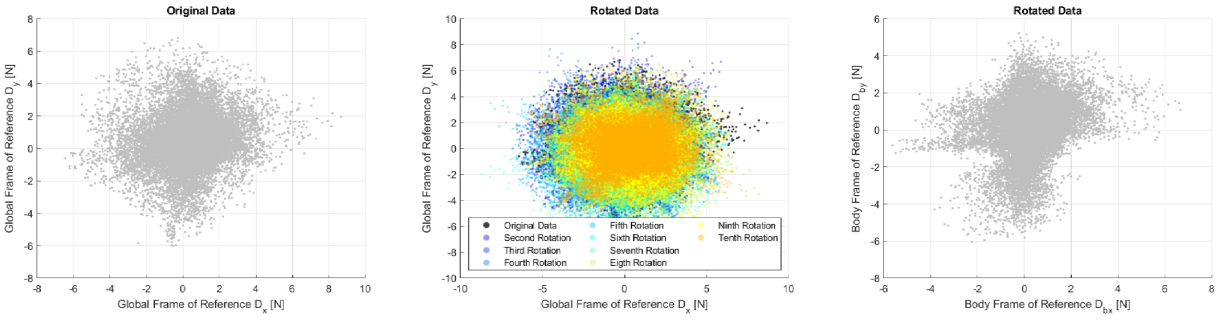


Figure 6.6: Left: Representation of drone state in global drag coordinates for all original data. Middle: Rotations of the drone state in global drag coordinates for 10 rotations over the full circle. Right: Representation of the drone state in body drag coordinates, which does not change with rotation of the global reference frame.

Here, we consider data rotation in order to teach the trained ML models rotational invariance about the global Z-axis. By doing so, we expect to increase the generalizability of these results as the estimations are no longer tied to the coordinate frame used in the particular training set. We rotate the data set a total of $k = 10$ times (including the original data set) over 2π, resulting in 0.62 rad or approximately 36° per rotation. Formally, this is implemented by the following 2-dimensional rotations:

$$\begin{aligned} i &= 2, 3, ...k, \\ \alpha^i &= \frac{2\pi}{n}(i-1), \end{aligned} \tag{6.6}$$

$$\begin{bmatrix} V_{W}{}^{i}_{X} \\ V_{W}{}^{i}_{Y} \end{bmatrix} = \begin{bmatrix} \cos(\alpha^i) & -\sin(\alpha^i) \\ \sin(\alpha^i) & \cos(\alpha^i) \end{bmatrix} \begin{bmatrix} V_{WX} \\ V_{WY} \end{bmatrix}, \tag{6.7}$$

$$\psi^i = \psi + \alpha_i, \tag{6.8}$$

$$\begin{bmatrix} \dot{x}^i \\ \dot{y}^i \end{bmatrix} = \begin{bmatrix} \cos(\alpha^i) & -\sin(\alpha^i) \\ \sin(\alpha^i) & \cos(\alpha^i) \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}, \tag{6.9}$$

$$\begin{bmatrix} \ddot{x}^i \\ \ddot{y}^i \end{bmatrix} = \begin{bmatrix} \cos(\alpha^i) & -\sin(\alpha^i) \\ \sin(\alpha^i) & \cos(\alpha^i) \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix}, \tag{6.10}$$

where the superscripts i represent the ith rotation for each variable. All other variables are not modified as they will be the same for each rotated coordinate frame, namely the vertical components of velocity and acceleration, the pitch, the roll, and all rotor speeds. This is well described by Figure 6.6, which shows how the drag estimate varies when the global reference frame is rotated. In body coordinates, this representation does not change. The downside of this operation however is that the number of training examples has increased k fold, going from 44,782 to 447,820 which greatly increases the training time.

6.3.2 Data Reduction

The purpose of data reduction is to remove data imbalance issues highlighted by Section 6.2.3 and to reduce the total training time by reducing the number of training examples without

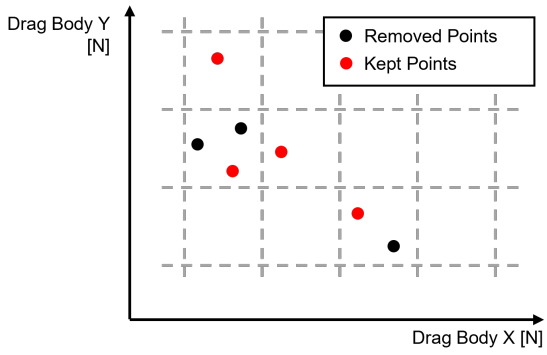


Figure 6.7: Description of grid-based data reduction technique addressing data imbalance. Similar to Figure in [46].

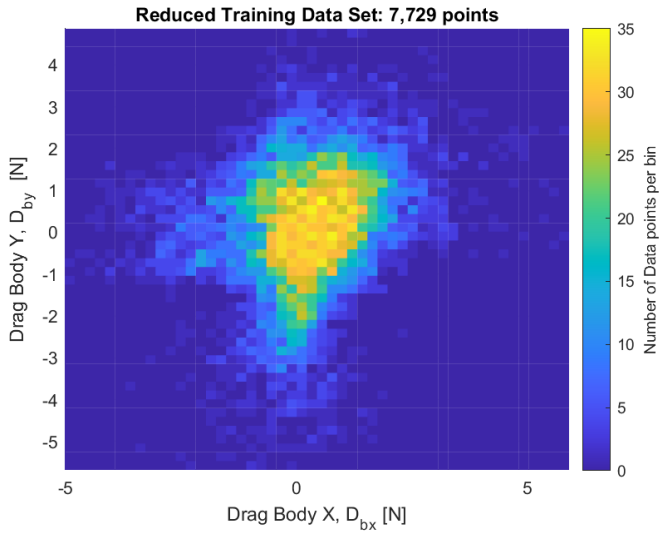


Figure 6.8: Histogram of data set after applying grid based data selection.

significantly affecting performance. Data is reduced by enforcing an approximately even data point density in the body frame drag coordinates, by applying the grid based data selection method described by Jules [46]. This space was divided up into an evenly spaced grid, where only one data point was randomly selected to be kept for the reduced data set per cell. The rest of the points in each cell were discarded. Random selection preserves the non-linear relationship obeyed by each data point, which is not true of an ensemble method like averaging. This process is well described by Figure 6.7. The number of cells was selected to be 300, which gave approximately even data point density between the training and test sets, the selection of which is described in the next subsection. The reduced training data set is shown in Figure 6.8. While this method is effective for reducing the training time and applying more favourable weighting, the downside is that the training procedure is more sensitive to the particular noise and errors of this reduced set. Jules describes multiple methods for dealing with this imbalance problem, the most complex of which is by adding a weight vector computed by euclidean distance metrics. This was not pursued as it would not reduce the rotated data set, requiring extensive training time. Additionally MATLAB was used for model training, which does not have built-in functionality for specifying a weighted loss function in NN training.

6.3.3 Training, Validation and Testing Data

Data rotation and reduction are both applied in order to produce appropriate training and validation data sets. Starting from the original data set, which comes from all test flights performed, this data is rotated and appended corresponding to a tenfold increase in length. The testing data set is selected as three complete test flights from the original unrotated data, each of which correspond to a representative flight condition. Namely, one test flight with hover in low wind speed conditions, another flight in hover with high wind speed, and another flight with high ground speed and high wind speed were used as the total testing set. Then from the remainder of the total data, data reduction is performed to produce a training data set. From the remainder of this operation, data reduction is performed again to produce a validation data set. By doing so, we produce a training data set with short enough length to provide fast training time, and without heavily imbalanced training examples. The validation data set

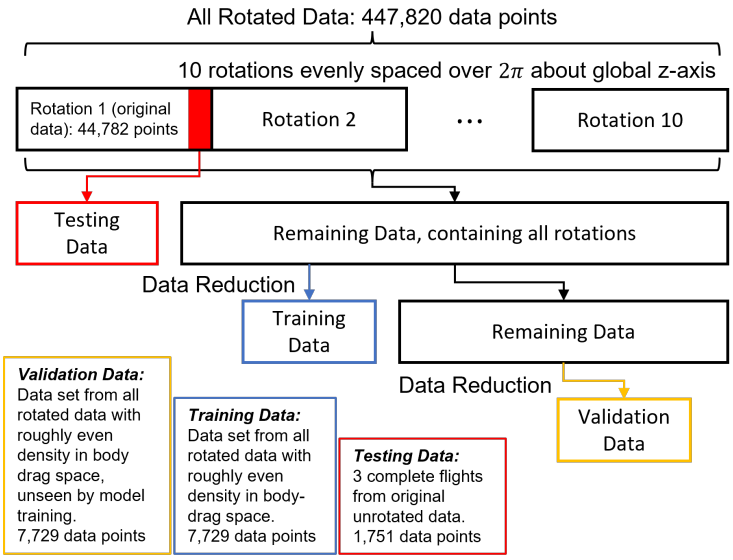


Figure 6.9: Summary of how training, validation and testing data sets were selected and reduced from the original rotated data set.

represents a similar data set to the training set in terms of state distribution and size, but is still unseen by the training procedure. It also comes from the total rotated data, meaning that it is a measure of how the model will perform on generalized data in any coordinate frame. Random selection is typically used when developing ML models, to avoid bias in performance assessment. However, in reality one would apply such a model to new, complete flight data which is what the testing data set represents. By using complete, non-random data sets for test we measure how well the model generalizes to new flights. Random selection is used in selection of the validation set, allowing for comparison between these metrics. This entire process is well summarized by Figure 6.9.

In order to demonstrate the effectiveness of these rotation and reduction techniques, multiple variations of these models are developed without these features and compared. In the case of models trained using reduction but not rotation, the training set is selected by performing data reduction on the remainder of the original unrotated data set alone. The validation data set is still selected from the rotated data, to show that this method does not generalize well to new coordinate frames. For models trained without rotation or reduction, the training set is taken as the total remainder of the original unrotated data set, with the same validation set. In all cases the testing and validation data sets are the same, to provide an effective comparison.

6.4 Machine Learning Model Development

Given the determined training, validation and testing data sets, this section describes the process used to train all ML models and their variations. Two main models are considered here: an LSTM model and a GRU model. LSTM's have been shown to be effective in this application by multiple sources [31, 29, 40], as they are optimized for time series analysis. A GRU model is considered as they are a simplification from the typical LSTM structure but still perform well on time series problems. While an LSTM unit is composed of three gates, namely an input, output and forget gate, the GRU combines the forget and input gates into a single update gate. GRU's are typically more computationally efficient than LSTMs, but with less complexity [36].

For both models, three variations are developed: one with data rotation and reduction applied, one without reduction but not rotation, and one with neither rotation nor reduction.

6.4.1 LSTM Model

LSTM models are an improvement upon general recurrent-neural-networks, as they are able to hold information from previous time steps in an internal cell state for an arbitrary number of time steps. They are a generalized approach to time series machine learning problems. To develop the LSTM NN, the same structure and training procedure was followed as in previous work [40], but is summarized here. Two hidden layers are used with 90 and 48 units each. Each layer uses L_2 regularization with a strength of 1×10^{-4}. No dropout layers were used as it was found that overfitting typically did not occur, which is seen by the training history plot in Figure 6.10. The Adam optimizer was used with an initial learning rate of 1×10^{-3} which dropped by 80% every 10 epochs until a maximum of 50 epochs. After this training, the training history was checked to see that overfitting did not occur. A mini-batch size of 30 was used. All model training was done using MATLAB, and was executed on remote clusters using Compute Canada resources.

6.4.2 GRU Model

A GRU NN is a simplification of an LSTM model, generally reducing the complexity of the model with faster training times and improved generalizability on less data. For consistency, the same structure and training procedure was used as the LSTM, that is two hidden layers with 90 and 48 units each with 1×10^{-4} L_2 regularization. The same training optimizer, learning rate schedule and batch size are used. The training history for the GRU is shown in Figure 6.11

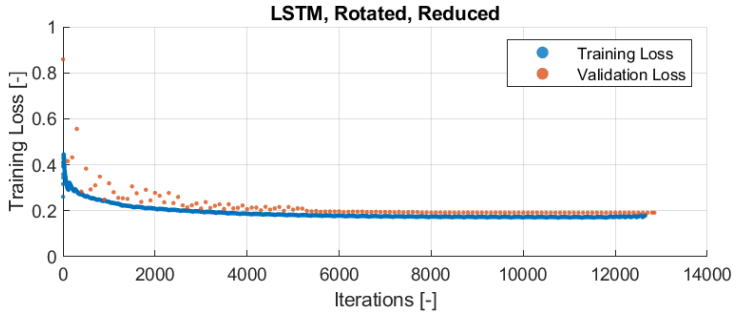


Figure 6.10: Training history of the LSTM NN with rotation and reduction applied.

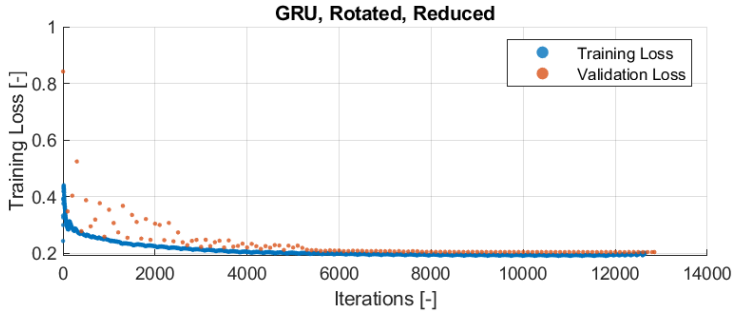


Figure 6.11: Training history of the GRU NN with rotation and reduction applied.

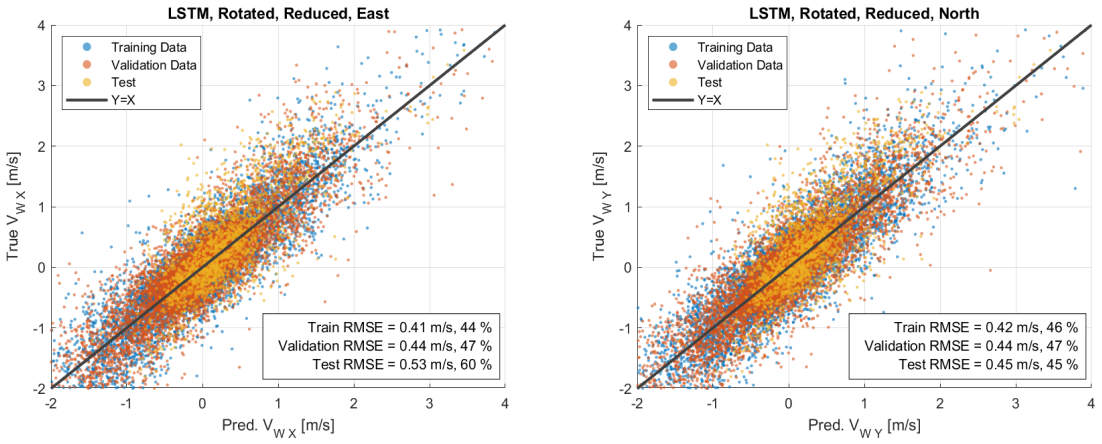


Figure 6.12: Scatterplot showing summary of performance of LSTM model with rotation and reduction applied.

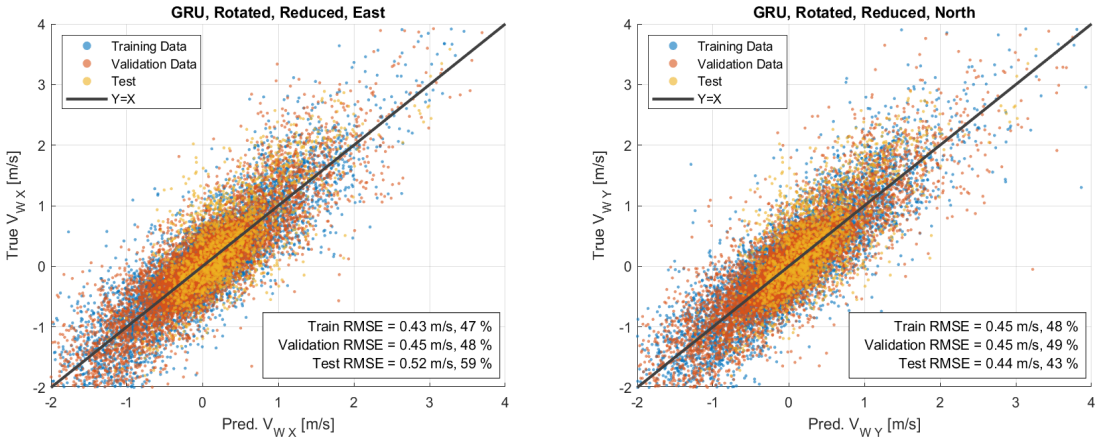


Figure 6.13: Scatterplot showing summary of performance of GRU model with rotation and reduction applied.

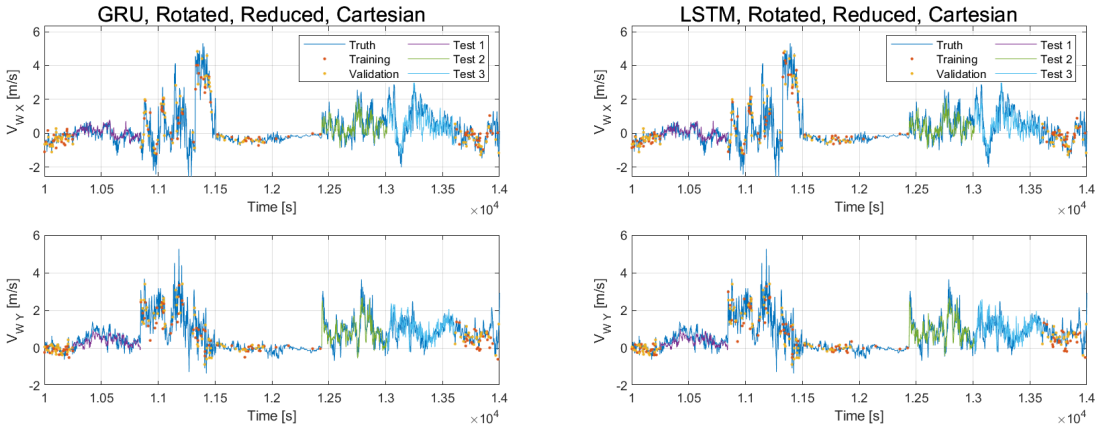
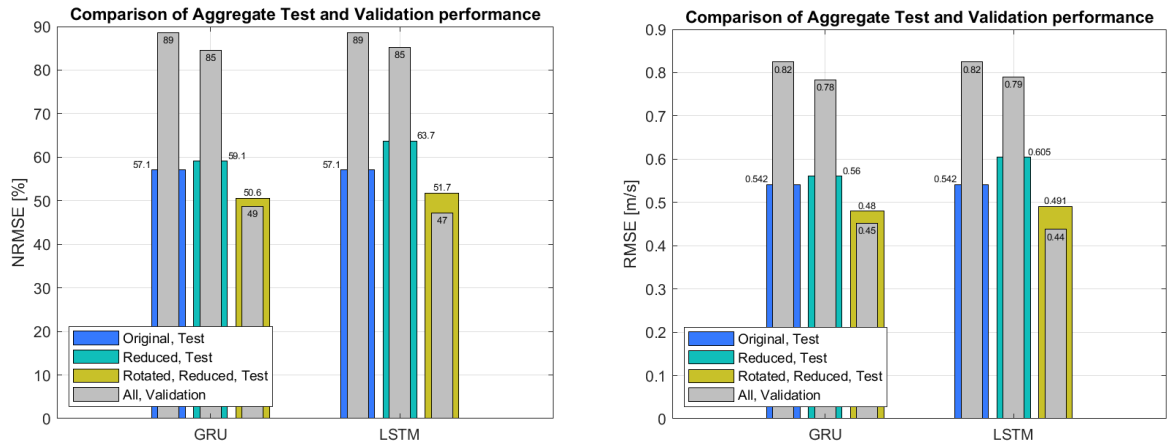


Figure 6.14: Left: Time series of three full test flights in testing data set for GRU model with rotation and reduction applied. Right: Same time series plot but for LSTM model.

Model	X Validation RMSE	X Test 1 / 2 / 3	Y Validation RMSE	North Test 1 / 2 / 3
LSTM, original	0.77 m/s (84%)	0.27 / 0.61 / 0.72 m/s (78% / 71% / 59%)	0.88 m/s (93%)	0.27 / 0.56 / 0.64 m/s (46% / 45% / 58%)
LSTM, reduced	0.75 m/s (83%)	0.30 / 0.69 / 0.83 m/s (84% / 81% / 68%)	0.83 m/s (87%)	0.29 / 0.61 / 0.71 m/s (48% / 49% / 64%)
LSTM, rotated and reduced	0.44 m/s (47%)	0.29 / 0.56 / 0.66 m/s (83% / 65% / 55%)	0.44 m/s (47%)	0.31 / 0.51 / 0.51 m/s (51% / 41% / 47%)
GRU, original	0.77 m/s (84%)	0.27 / 0.61 / 0.72 m/s (78% / 71% / 59%)	0.88 m/s (93%)	0.27 / 0.56 / 0.64 m/s (46% / 45% / 58%)
GRU, reduced	0.75 m/s (83%)	0.27 / 0.65 / 0.77 m/s (78% / 76% / 63%)	0.82 m/s (86%)	0.29 / 0.56 / 0.64 m/s (49% / 45% / 59%)
GRU, rotated and reduced	0.45 m/s (48%)	0.30 / 0.54 / 0.65 m/s (86% / 64% / 53%)	0.45 m/s (49%)	0.30 / 0.51 / 0.48 m/s (49% / 41% / 44%)

Table 6.1: Comparison of validation and test performance metrics for the LSTM and GRU models, and their variations.



(a) Normalized (b) Absolute

Figure 6.15: Bar plot comparison of aggregate test and validation performance.

which again shows monotonically decreasing loss which never increases. Both models achieve similar validation performance, meaning that we expect both models to perform similarly.

6.5 Performance Evaluation

Performance for each trained model is determined by accuracy, as measured by the root-mean-square-error (RMSE), on the validation and test data sets as well as the total training time. RMSE is used as the primary performance metric, as it provides an assessment of variance and bias errors and is in absolute units of [m/s] which is intuitive. Since this error is proportional to the size or variation of a signal, it is not useful for comparing different data sets. For these comparison, the normalized-root-mean-square-error (NRMSE) is used, which is calculated as the RMSE divided by the root-mean-square (RMS) of the true signal.

Total Training Time	LSTM	GRU
Original	1733 s	1775 s
Reduced	269 s	268 s
Rotated, Reduced	335 s	324 s

Table 6.2: Comparison of training time required for each model.

6.5.1 Model Accuracy

Figures 6.12 and 6.13 show a scatterplot comparison between the estimated wind values and true wind values in each principle coordinate of the global reference frame. Performance is quantified in the training error, the validation error, and test errors all in absolute and normalized units. For clarity, only plots are shown for the LSTM and GRU rotated and reduced variations, but all performance metrics are summarized in Table 6.1. A time series comparison is shown for both the LSTM and GRU in Figure 6.14, indicating the segmentation between the test data sets and the grid reduction selection of the training and validation sets. Since there are multiple performance metrics to track, Figure 6.15 shows the aggregate test and validation performance metrics for each model providing a simpler comparison. This data is shown with both absolute and normalized units.

There is a very clear benefit to providing data rotation, as it effectively improves the estimation in generalized global coordinates which is seen by the large difference in validation performance. The test performance metric, which only consists of data from the original data set, improves too but by not as much. This indicates that rotating and reducing the data set improves the generalization of the results into new, complete flights. The LSTM reaches the best validation performance of 0.44 m/s RMSE or 47% NRMSE, while the GRU achieves the best test performance of 0.48 m/s RMSE or 50.6% NRMSE. The validation performance reached here is worse than that reached by [40], but the test performance is better for the same data. The similarity between the test and validation performance metrics likely indicates that this is much closer to realistic performance given sensor biases and day-to-day variations.

6.5.2 Model Training Time

In addition to comparing the accuracy performance benefits of rotating and reducing the data set, we consider the benefits in the training time which is primarily due to the decreased number of data points used for training. This comparison is shown in Table 6.2. The most significant different in training time is between the variations with data reduction, and the variation without with an increase in training time of at least six times relative to the fastest trained model. It should be noted that training models on the entire rotated data set without reduction applied took an excessive amount of time, over eight hours, which is why these models were not developed for comparison.

6.5.3 Limits of Performance

As this is an experimental study, the sources of error limit the achievable accuracy of the developed models. In our analysis we have assumed a spatially homogeneous wind field which only varies in time to obtain true wind speed labels, the validity of which we investigate here. As previously described in Section 6.2.2, two ultrasonic anemometers placed at different locations when flying the drone are used to assess the spatial variation of the wind. This method for

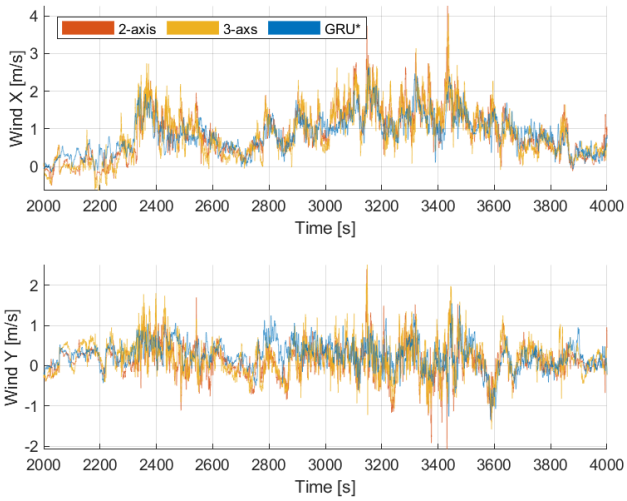


Figure 6.16: Comparison of time series of 3-axis anemometer reading, 2-axis anemometer reading, and GRU model estimations.

measuring the spatial variation is much more effective than previously considered convection models [40].

Figure 6.16 compares the measurements by the 3-axis anemometer, the 2-axis anemometer and the predictions by the GRU model with data rotation and reduction applied. This is a representative time series in order to show fine details, but in reality this time series is longer. This comparison is made on the complete original data set with both anemometers signals available, including training examples which is generally bad practice. This is not expected to be an issue, as the data reduction process leaves an excess of unseen data and training data only accounts for 1.7% of this data set. We expect the variance between each signal to be comparable, which is generally true except for brief deviations. For example, there is a temporary deviation around 2800 seconds in global Y wind estimation. It is unknown whether these deviations are a failure of the model, a gust of wind experienced by the drone but not the anemometers, or simply a particular sensors inaccuracy issue. To quantify the comparison between the anemometers and the drone prediction, we compute the RMSE of differences between the 2-axis and 3-axis anemometer and compare this to the errors within the model predictions. This is shown in Figure 6.17, which also considers how this RMSE decreases with increasing averaging filter time. This comparison is useful for showing what level of accuracy is attainable when less-frequent measurement is required. When applying a 100 second averaging filter, the models reach estimation accuracies of less than 0.2 m/s RMSE. For comparison, Crowe's results are included as well, however it is important to note that we do not have the time series data for this performance available meaning that we cannot normalize these results for comparison.

6.6 Discussion

These results show the possible accuracy performance achievable by multirotor drones estimating wind by use of a ML based disturbance observer. In this paper we have taken as a given that the hybrid set of drone state inputs yields good results, which is thoroughly discussed in [40]. The primary investigation is the effects of applying data reduction and rotation, which decrease

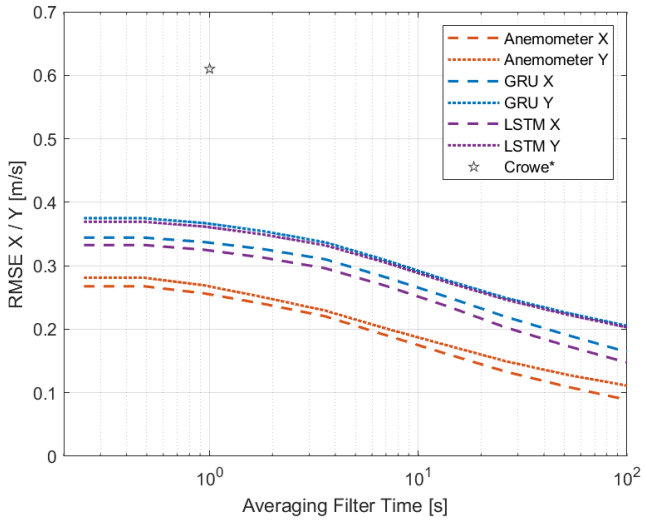


Figure 6.17: Comparison of how the spatial variation of the wind, and the model estimation accuracy decrease with increasing averaging time, in absolute units. *Crowe [31].

the required training data, decrease the training time, and improve model performance over generalized data sets. Validation and performance metrics were selected that showed how the models performed on both randomly selected data and complete test flights. While random selection is very common in assessing ML models to avoid bias, there is likely variation in sensor performance between flights and including unseen complete flights assesses how our ML models generalize to this.

While a large number of data points were collected and then reduced to a reasonable number for training, this procedure is not required when replicating or applying this work. We can look at the method for reducing the data set as a framework for what types of test flights are important for training these ML models. Generally, it is desirable to achieve an approximately uniform density of data points in body frame drag coordinates which can be obtained by flying in high wind conditions or flying aggressively in low-wind conditions. Inevitably due to take-off, landing, and loitering, there may be an excess of data points corresponding to low-drag conditions which can be addressed by the same reduction technique, or time-based data segmentation and removal. During flight, it is important to fly the drone in varying orientations with respect to the airspeed direction in order to fully explore the body-drag space. Intuitively this is understood by the fact that the models learn the drag characteristics of the drone in an arbitrary orientation, which is achieved by including training examples of the drone experience ranges of airspeeds in all orientations.

While Figures 6.16 and 6.17 indicate that the spatial variation of the wind is a significant factor in determining the limits of performance given our experimental method, there is still a gap of about 20% between the anemometer variance and model accuracy. This could be caused by spatial variation not captured by the two anemometers, or model/sensor input error. An important factor is that the same configuration of anemometers was used for varying wind conditions, where the primary wind direction is changing. We expect a higher spatial correlation of wind when measured parallel to the primary direction of wind rather than perpendicular [47]. Other sources of error include variation in sensor performance from day to day, which the ML models assume is uniform. For example, the GPS accuracy depends on how long the ground station has been stationary for. Therefore, if multiple test flights are conducted consecutively

the last test flight will have higher GPS accuracy than the first.

6.7 Conclusions and Future Work

In summary, we have presented a ML based disturbance observer for multirotor drones which can be used to implicitly measure the wind field, a tool useful in environmental monitoring, wind farm site surveying and other applications. The developed disturbance observer builds upon previous works by improving the accuracy over generalized flight conditions, applying data rotation and reduction techniques. Data rotation allows the ML models to learn rotational invariance, by including training examples in arbitrary global coordinates. Data reduction provides appropriate weighting to training examples, by removing data point imbalances which originally favoured low-drag conditions. This also offers the benefit of reducing the training time significantly relative to training on unreduced data. The trained GRU model reached 0.48 m/s RMSE on unseen testing data, which was comprised of complete test flights, which properly assesses how this model will perform in realistic generalized conditions.

Given the current experimental method, performance is limited by the spatial variation of wind. It is recommended that future work be focused on improving the experiment rather than changing or tuning the models used, as this will likely yield the most significant improvements to performance. Either observed laminar flow conditions in the field or enforced in a wind tunnel can be used to decrease the spatial variation, although use of large wind tunnels with room for dynamic flight are much less commonly available to researchers. A more expensive LIDAR anemometer that can measure the wind field over spatial dimensions could be used in the field to improve the method. Additionally, this work represents a demonstration of wind estimation by multirotor drone that uses post processing exclusively. This is suitable for some applications, but not when information is to be used for real-time decision making. It would be highly valuable to demonstrate a real-time disturbance observer, making autonomous decisions about localizing unknown gas sources or modifying planned flight trajectories based on the estimated wind.

6.8 Supplementary Material

This chapter describes the application of data rotation and reduction techniques in order to improve the generalizability of the trained models. The data pre-processing involved first rotating, then reducing the data based on a grid reduction method in the body-reference-drag space in order to select the training and validation sets. However these methods could be applied in different ways, for example first reducing the data and then rotating the reduced data set to produce the training data. Analysis of these methods is included in Appendix D. Generally it was found that these variations performed less consistently and worse than the models presented in this chapter. These models reach good validation performance metrics, less that 0.4 m/s RMSE, but poor test metrics generally around 0.7 m/s RMSE.

Chapter 7

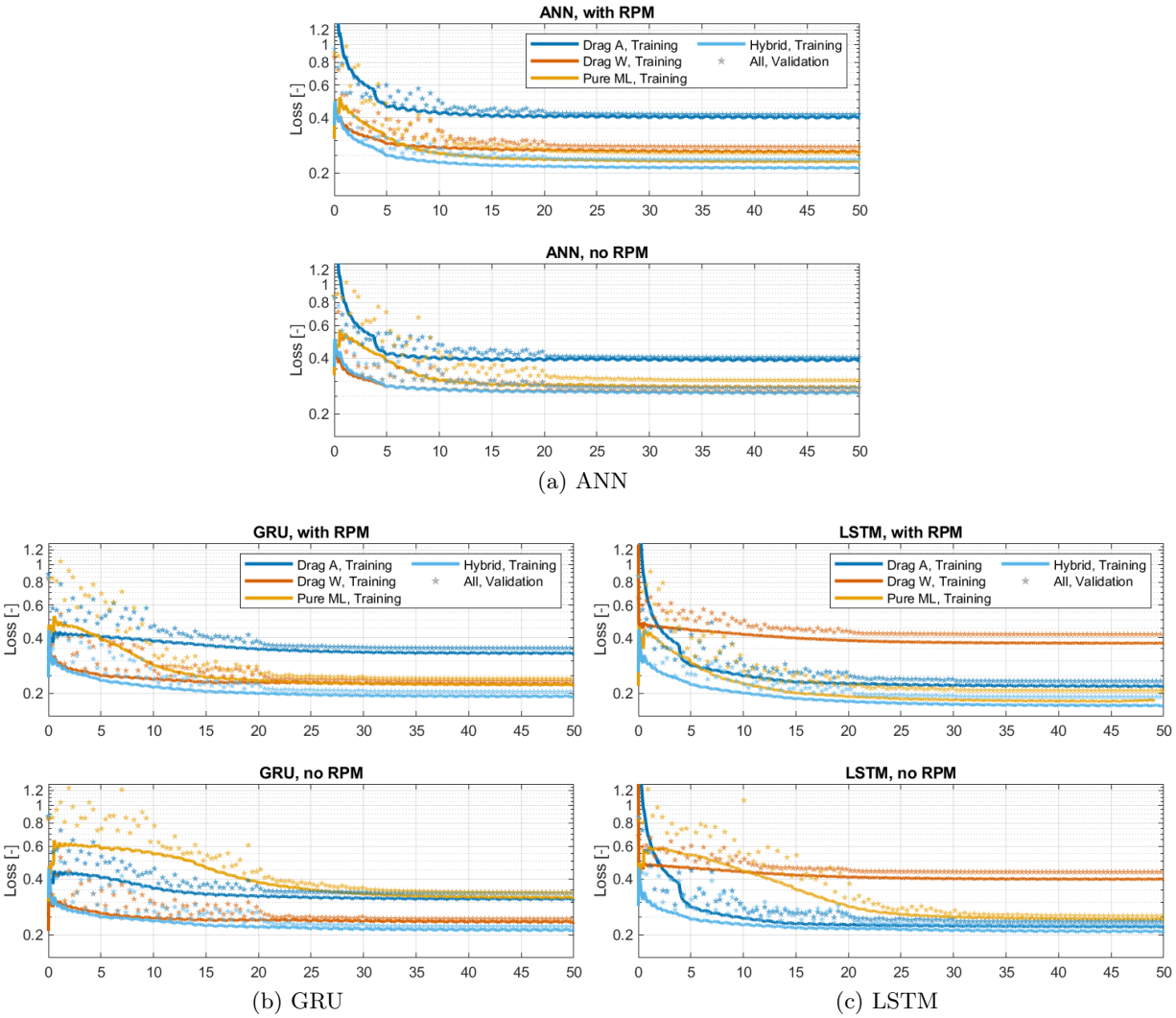
Discussion

Chapter 5 establishes using ML models for the problem of wind estimation by multirotor drone, the best of which is an LSTM based on a hybrid physics-ML approach achieving 0.34 m/s RMSE or 48% NRMSE on randomly selected training data and 0.55 m/s RMSE or 57.5% NRMSE on complete, unseen test flights. This identified a key issue being the decrease in performance when moving from randomly selected unseen data to complete flights. Chapter 6 improves upon this work utilizing data augmentation techniques, namely data rotation and reduction. This teaches the ML models rotational invariance, an inherent property of the physical system and removes data imbalance issues that improperly weight training examples. Both of these procedures were applied to an LSTM and a GRU based on the same hybrid approach, with the GRU reaching 0.48 m/s RMSE on unseen complete flights which is much more consistent with randomly selected data. This work also showed that the same models trained in Chapter 5 did not generalize to the rotated data. This section revisits the problem of model selection comparing the different approaches of applying ML with and without motor speeds as an input with data augmentation by rotation and reduction universally applied. Section 7.1 describes this comparison, giving a full understanding of how performance varies for different models. ML model capacity and learning rate variations are considered to see if they have a significant effect on model performance. Then, Section 7.2 explains how the performance of the best of these models varies across different flight phases and in reference to the expected experimental limits of accuracy similar to discussions in each of Chapters 5 and 6.

7.1 Improvements to Methods

7.1.1 Framework Comparison

We can revise our analysis in Chapter 5 to apply the improvements of data rotation and reduction, and to consider all four ML frameworks. That is, we will compare models in the drag-to-airspeed approach, the drag-to-windspeed approach, the pure ML approach and the hybrid approach with variations for including and omitting the motor speed signal all with rotation and reduction applied. For all approaches each of an ANN, a GPR, a LSTM and a GRU model is trained. Meaning that this comparison uses a total of 32 models. For consistency all model structures are kept the same as described in Chapters 5 and 6. All training parameters, learning rates and schedules, and hyperparameters are kept the same. The training histories for the parametric models, that is the ANN, GRU and LSTM models, are shown in Figure 7.1. Figure 7.2 shows the aggregate achieved performance metrics of each model. The term aggregate refers to the fact that these performance metrics are computed on in two cartesian directions. Overall the GRU and LSTM with the Hybrid approach and motor speed included as a feature achieve the best performance which is consistent with our expectations.



(a) ANN

(b) GRU

(c) LSTM

Figure 7.1: Training history for all parametric models of scope variations

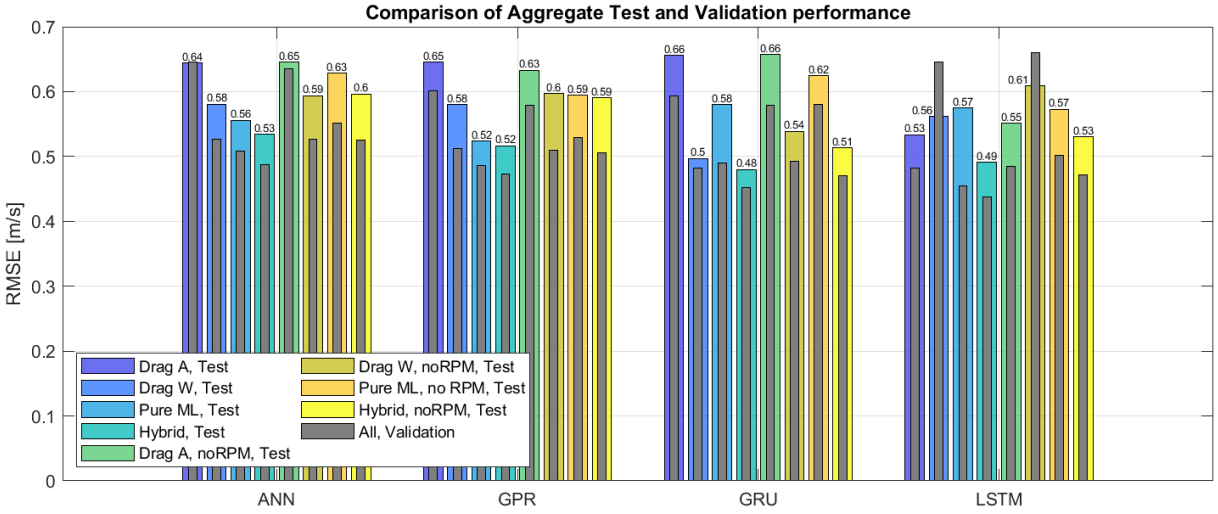


Figure 7.2: Comparison of all models on all scopes with data rotation and reduction applied.

7.1.2 Model Capacity Comparison

When assessing the performance of a ML model it is important to consider variations to the capacity or hyperparameters. In the case of a NN, this means considering models of varying number of layers and hidden states. The term "capacity" typically refers to the number of tunable parameters within a NN; models with larger capacities generally can learn more complex relationships at the risk of overfitting. Given that the GRU and LSTM models perform well when using this hybrid approach, we now consider a coarse attempt at capacity tuning by comparing variations with increased number of layers. Since overfitting is generally not an issue which is observed in the training history plots in Chapter 6, we consider increases in model capacity summarized by Table 7.1. Each of these models are trained using the same procedure described previously, specifically using Adam as an optimizer and a piecewise learning rate which starts at 1×10^{-3} and decreases by 80% every 10 epochs. The models are trained to a maximum of 50 epochs.

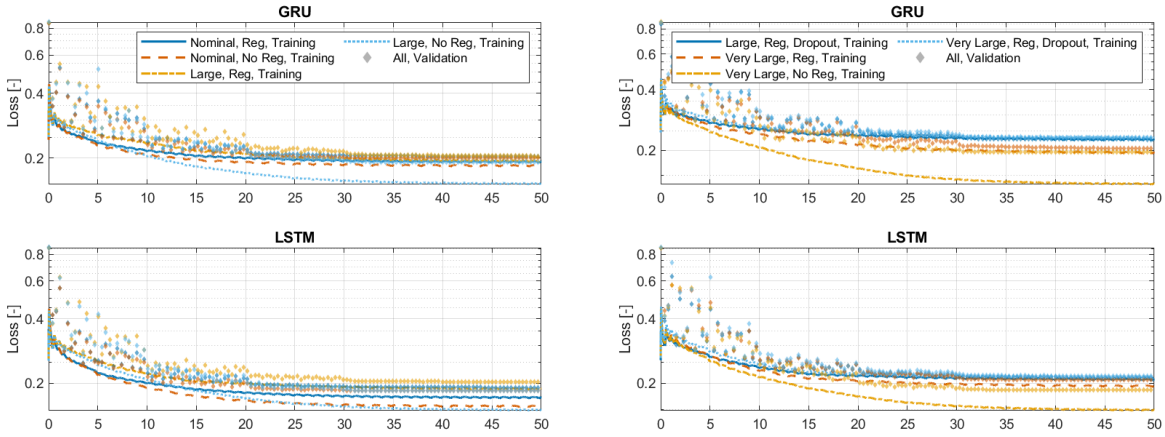
Figure 7.4 shows the comparison of the achieved performance of each model. Generally these results are very robust, with the larger models being able to decrease the validation error but not the test error. For the GRU, the nominal model with regularization (which was the original model developed for comparison in Chapter 6) still performs the best on the testing data set. For the LSTM, this best performing model is the most complex model with regularization and dropout. These results do not vary significantly though, with the total number of hidden states changing from 138 in the nominal model to 1158 corresponding to an approximately tenfold increase in model capacity.

7.1.3 Learning Rate Considerations

The learning rate is a key parameter in training parametric ML models, which controls how fast the gradient-based training methods move towards the local optimum. Generally a learning rate that is too high can cause the validation performance to diverge or not find an optimum point, but one that is too low may take excessively long to train. To ensure that the nominal learning rate used provides reasonable results, the initial learning rate was varied over four orders of magnitude, from 1×10^{-2} to 1×10^{-5}. Each of these cases uses the same learning rate drop schedule, which is piecewise decreasing by 80% every 10 epochs. The nominal initial learning rate, which was used for the previously described analysis is 1×10^{-3}. The results of these trained models are shown in Appendix E, but they generally show that changing the learning rate does not offer any significant performance improvements in the trained local optimum. Generally these, and the model capacity comparisons described in Section 7.1.2 show that the

Abbreviation	Layers	Units	L_2 Regularization	Dropout
Nominal, Reg	2	90, 48	0.0001	None
Nominal, No Reg	2	90, 48	None	None
Large, Reg	4	360, 180, 90, 48	0.0001	None
Large, No Reg	4	360, 180, 90, 48	None	None
Large, Reg, Dropout	4	360, 180, 90, 48	0.0001	3 layers, 0.5
Very Large, Reg	5	480, 360, 180, 90, 48	0.0001	None
Very Large, No Reg	5	480, 360, 180, 90, 48	None	None
Very Large, Reg, Dropout	5	480, 360, 180, 90, 48	0.0001	4 layers, 0.5

Table 7.1: Summary of model capacity variations and abbreviations for reference.



(a) Nominal to Large (b) Large to Very Large

Figure 7.3: Results of varying model capacity, showing training history.

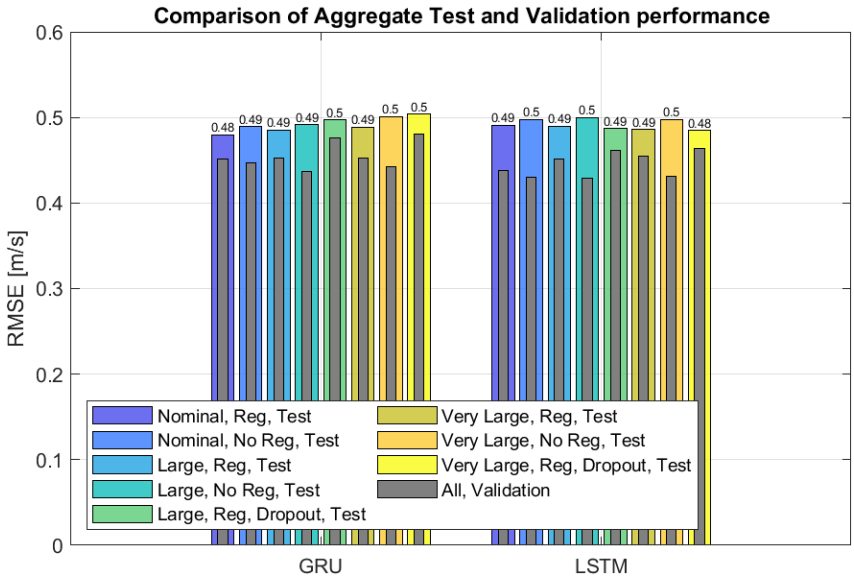


Figure 7.4: Results of varying model capacity, showing aggregate performance metrics.

performance is robust to changing network size and training parameters, indicating that we are likely nearing experimental limitations which is explained further in Section 7.2. For the remainder of this work comparing the model estimations to the expected experimental limits of performance in Section 7.2, the nominal models with regularization and the nominal learning rate are used.

7.2 Performance Considerations

The sources of error in this experimental study determine the limits of performance, with the expected largest contributing error being the spatial variation of the wind. To assess this, we consider the variation between the 2-axis and 3-axis anemometers. Figure 7.5 compares the estimation between the GRU model, and the wind as measured by each anemometer on a 2000

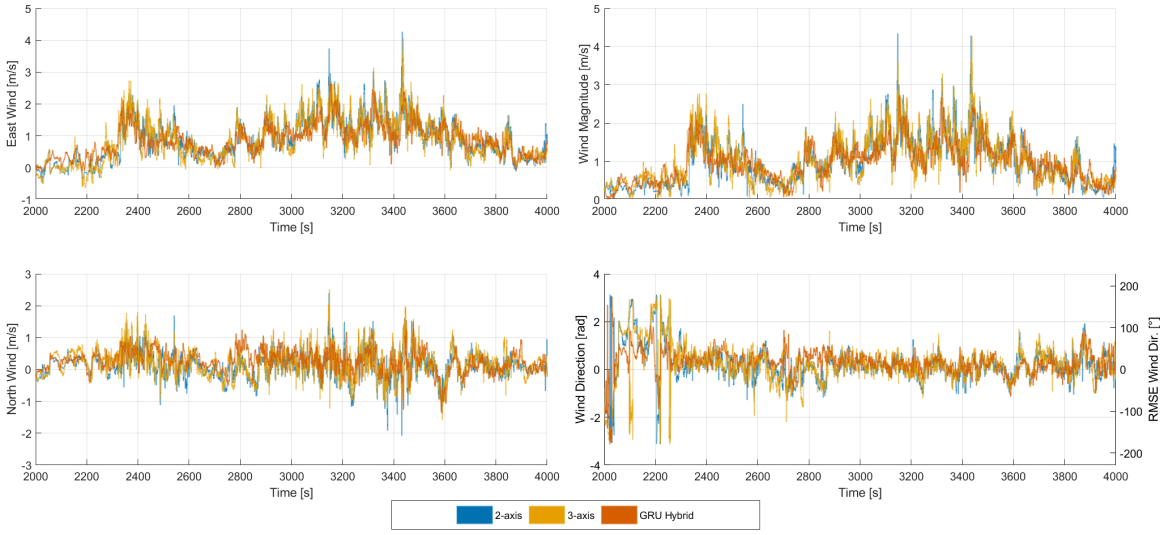


Figure 7.5: Comparison of GRU model and both anemometers measurements.

second section of data. Generally the estimations in cartesian coordinates are accurate, but deviate in particular time instances like near 2800 seconds. It is unclear whether these errors are a failure of the model, a particular gust of wind not measured by the anemometers or some other source of error like variation in sensor performance. These cartesian estimations are converted into polar coordinates, comparing wind magnitude and direction. It is important to note that this comparison involves a mix of training, validation, testing data, and completely unseen data neither in testing or validation. This is done to provide an estimation on the full, ordered time series data for which the training data was randomly selected from. While it

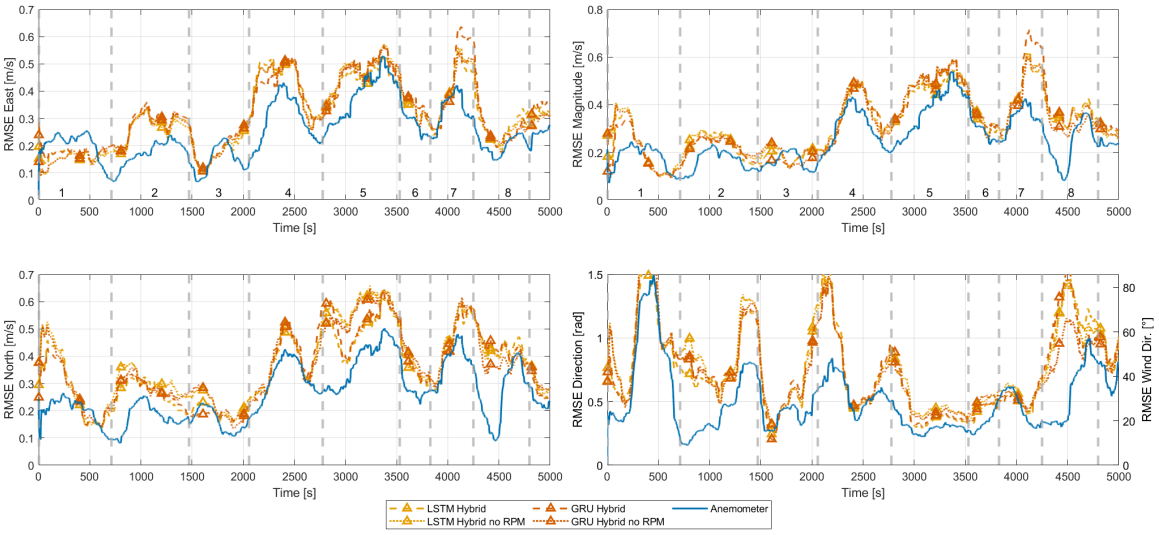


Figure 7.6: Comparison of how the RMSE changes over time for final models, relative to spatial variation.

is generally bad practice to report estimations made on the training data, the data reduction technique used means that only 1.4% of this data is from the training set.

To quantify how the error changes over time, the RMSE calculation was modified to be a function of a moving average of error. This was used to produced Figure 7.6, quantifying how the error changes in different flight phases. That is, the RMSE in this plot is calculated by

$$RMSE_i = \sqrt{\frac{\sum_{j=i}^{i+800}(\hat{y} - y)^2}{800}}, \tag{7.1}$$

for a specified moving average period of 800 data points. This is plotted for the GRU and LSTM using the hybrid approach both with and without RPM, and the anemometer variation. Generally the variation between the anemometers increases when the drone estimation increases, further indicating that the spatial variation is the leading cause of error. When the wind magnitude is high, as is true around 3200 to 3400 seconds, generally the wind direction estimation accuracy improves while the magnitude accuracy gets worse. The opposite of this is true, which can be seen looking near 2000 seconds when the wind magnitude is approximately 0.5 m/s. Here the wind magnitude error improves while the direction error increases. These comparisons are made referencing both Figures 7.5 and 7.6, noting the difference in time scales.

Figure 7.7 shows how the error decreases for an increased moving averaging period which is important when less-frequent data is of interest. This is done on the same data set where both anemometer's data is available to include their variation in the comparison. The anemometer comparison represents the limits of accuracy of this method, meaning that it will be impossible to push the accuracy of an implicit disturbance wind observer better than this without a revised experimental method such as wind tunnel testing. All models performance however is consistently above the anemometer accuracy meaning that there are likely other sources of error present in the study that limit performance. There is a clear benefit in performance relative to previously published works by Neumann and Crowe, however it is important to acknowledge that we cannot make a one to one comparison here as these are computed on different datasets. Presented in Neumann's thesis [23], his estimations are made on high wind from 1-5 m/s, which explains why the wind magnitude estimations are worse and the direction estimations are better. For another comparison, we consider this smoothing plot performed on the second and third testing datasets where the wind is higher, up to 4 m/s. This is shown in Figure 7.8, where the direction estimations are much closer to that achieved by Neumann while still outperforming in terms of wind magnitude estimation. There is also a much bigger difference in terms of whether the RPM is included as a feature. Here, the models that include RPM are consistently at least 0.1 m/s RMSE better than the other models in terms of magnitude.

7.3 Wind Signal Properties

Thus far, comparisons have been made between the GRU model estimation and true wind signal raw time series in terms of instantaneous comparisons or averaged comparisons. In wind farming applications and others, the actual wind speed distribution or properties of the wind signal are of interest. We can compare the wind speed distributions of the estimated and true signals, as well as the frequency spectrum of each to compare how physically realistic the estimated wind signal is. This is an important consideration when using ML models, as a ML model may produce an estimate that performs well in terms of loss but doesn't produce a physically realistic signal. These comparison are made in Figure 7.9, where the GRU and LSTM achieve very accurate reporting of the mean wind in each principle coordinate (X and Y). They tend to

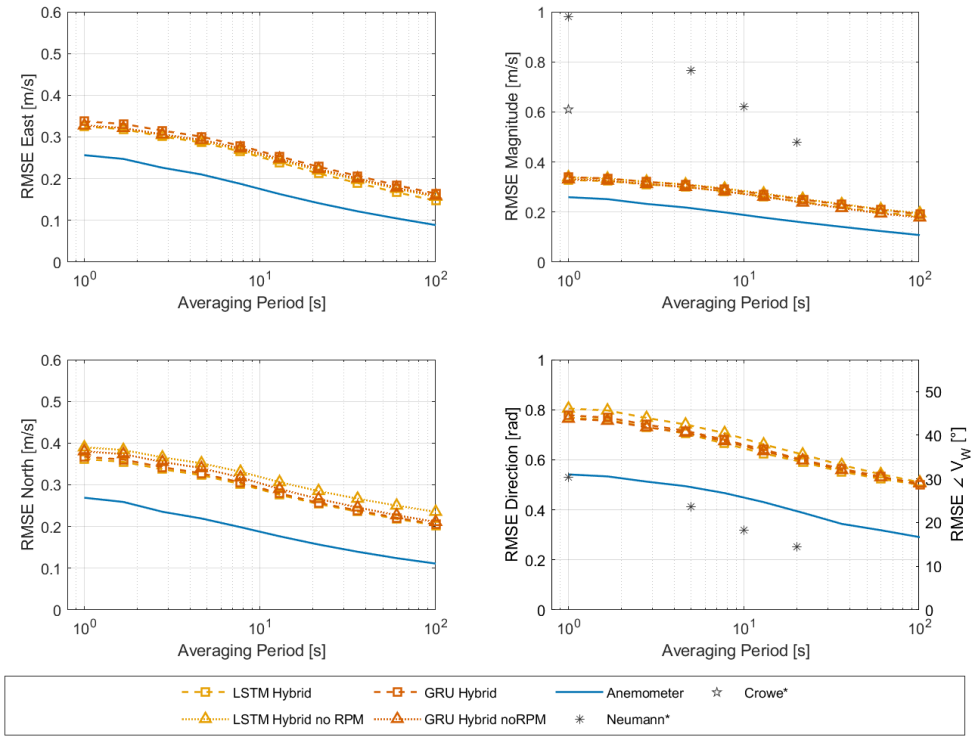


Figure 7.7: Comparison of Final models performance with increasing moving average filter, relative to spatial variation. *[22, 31].

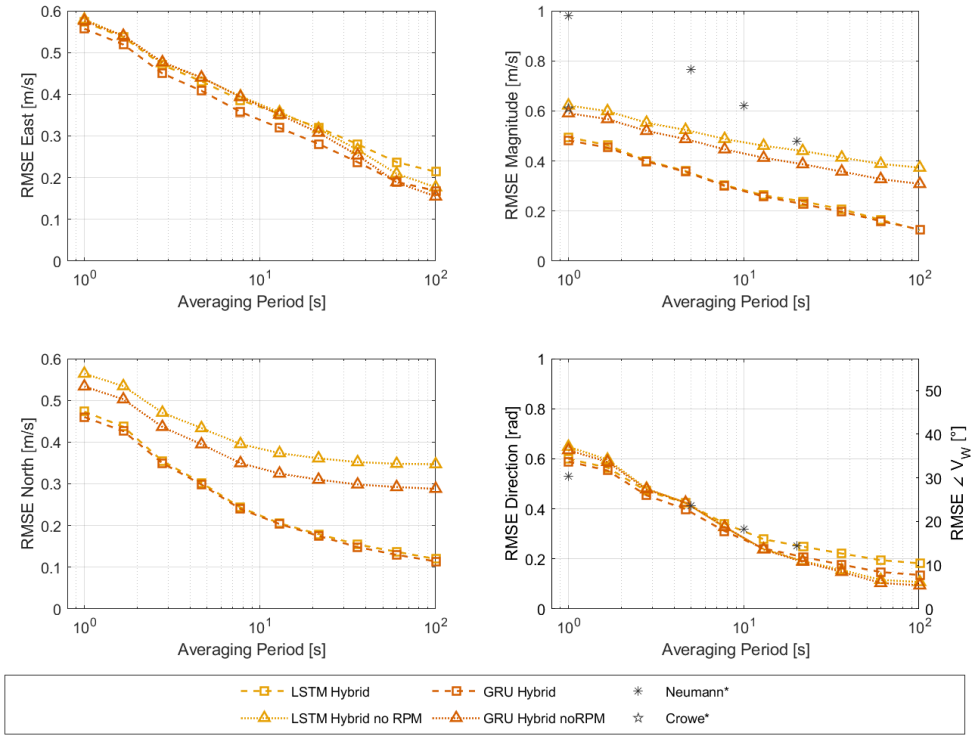
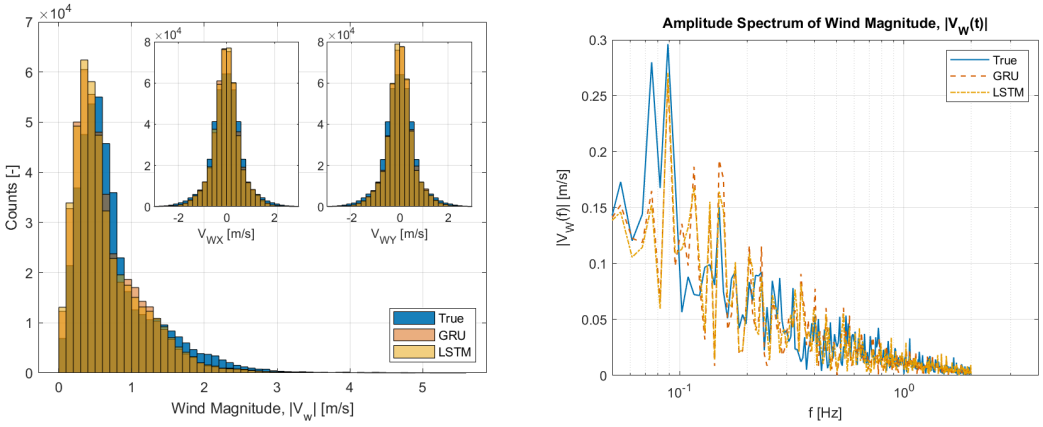


Figure 7.8: Comparison of Final models performance with increasing moving average filter, relative to spatial variation. Computed on the second and third testing datasets. *[22, 31].

underestimate each tail of the distribution and over estimate wind close to zero. When converting these estimations into magnitude, this results in a wind speed distribution that slightly under estimates the mode by up to 0.2 m/s. The frequency spectrum comparison was computed using a fast-fourier-transform on the time series data of the third test flight (September 12 2021 Flight 2) to avoid the frequency changes when appending time series together. Overall though, the histograms and frequency spectrum agree quite well, and the wind estimation models produce physically realistic and accurate wind signals.



(a) Histogram

(b) Frequency spectrum

Figure 7.9: Comparison of wind speed distribution and spectrum to check how realistic estimated wind signals are.

7.4 Discussion of Results

Overall this work combines data augmentation by rotation and reduction to produce robustly performing ML models, the best of which is the GRU which reaches 0.48 m/s RMSE on complete unseen flight data for winds up to 4 m/s and dynamic flight, although the LSTM performs very similarly. This work approaches the experimental limit of performance which is governed by the spatial variation of the wind as we cannot locate an anemometer at the drones location. The spatial variation as measured by two anemometers offset in space accounted for 73-80% of the GRU error, depending on whether the comparison is made on X wind, Y wind or polar magnitude. As justified by Figures 7.5, 7.6 and 7.7, this is the primary limit of model performance; ML models will not achieve better accuracy than this without first changing the experimental method of obtaining a true wind signal at the drones location.

It is unknown whether the source of the remaining 27% of error is due to uncaptured spatial variation of the wind, modelling error, or sensor error and day-to-day performance variations. Based on the robust performance of the ML models to changes in capacity and learning rate variations, it is suspected that model error is low relative to the other sources of error. Based on the similar histogram and frequency spectrum properties of the models and true wind signals, it is expected that this remaining error is primarily due to uncaptured spatial variation of the wind. When we consider the drone flight paths relative to the two anemometers location, (shown in Figures 3.5) there is a high chance that there are gusts of wind being experienced by the drone not fully captured by either of the 2-axis or 3-axis anemometer. However, this is difficult to confirm without knowledge of the wind at the drones location.

Chapter 8

Conclusion

This work presents a comprehensive study of the development of a disturbance observer for wind estimation using multirotor drones. Initially, a purely physics-based modelling approach was considered which has a number of advantages such as inherent rotational invariance, improved interpretability and simpler validation. These efforts determined that there was a difficult to identify non-linear changing drag coefficient at an arbitrary drone orientation, which is the primary motivation for applying ML models despite their challenges. Further, ML models are not constrained by a zero-lift assumption. Chapter 5 presents the first study applying ML, comparing an ANN, GPR, and LSTM on different input-output scopes. The best performing scope which was consistent across model types was a developed hybrid approach, which utilized the equations of motion to produce drag estimates as an additional input. These models perform well on unseen randomly selected data, to 0.34 m/s RMSE, but don't generalize well to unseen complete test flights. The primary drawbacks identified in this study were lack of rotational invariance considerations, where no attempt was made to generalize the training procedure to different coordinate frames, and data imbalance favouring low-drag, hover conditions. Chapter 6 improves upon these identified drawbacks developing data rotation procedures to teach ML models generalization to rotated global coordinate frames and grid-based data reduction procedures addressing data imbalance. Using these procedures with an LSTM and GRU on the hybrid approach yields much more consistent validation and test metrics of 0.45 m/s and 0.48 m/s RMSE respectively indicating that this likely approaches the true model performance.

Chapter 7 perhaps presents the most valuable portion of this work, where the main results of each previous chapter are integrated and the performance is compared for changing flight phases with respect to the expected experimental limits of performance. ANN, GPR, LSTM and GRU models are compared on 4 varieties of model input-output scopes, further confirming that the time series based models (LSTM and GRU) perform the best overall with the hybrid modelling approach. Changes in the implementation of the ML models show that performance is robust to changing capacity and learning rates. The experimental limits of this work are largely determined by the spatial variation of the wind, which is measured by the variance in two anemometers located at different spatial locations. The variation between each anemometer was measured to be 0.26 m/ RMSE, which accounts for 79% of the error in the GRU predictions at 0.33 m/s RMSE on the same data set in wind magnitude. The anemometer error at low wind speed also contributes to this, although this was shown to be negligible in Appendix C. Furthermore, there is strong similarity between the anemometer error and the drone error in Figures 7.6 and 7.7. This shows that the experimental method limits the performance, and that no amount of sensor improvements or ML model tuning will improve the results beyond this. New methods of obtaining the true wind at the drone location are required to develop a disturbance observer capable of exceeding this limit of accuracy. It is unknown whether the approximately 20% difference between the GRU model predictions and the anemometer variance is due to model underfitting, sensor errors, or further spatial variation of the wind not captured by the two anemometers. Based on the model consistency for changes in structure and learning schedule, and the realistic nature of the produced wind signal, it is expected that this 20%

is primarily due to uncaptured spatial variation although this cannot be confirmed without improved knowledge of the wind at the drones location.

This work is a valuable resource for those looking to apply a similar method for flux estimation, gas source localization, or other studies requiring a spatially precise estimate of the wind field. The general method can be applied to any generic multirotor, only requiring training flights in the presence of an anemometer for the ML models to implicitly learn the appropriate drag coefficient properties of the drone used. The data reduction methods can be used to guide the desired training data, where it is necessary to achieve an approximately uniform density of data points in body-reference-drag coordinates. This is best achieved by flying the drone such that it experiences drag of varying magnitudes in all relative orientations.

8.1 Future Work

To improve upon implicit wind estimation accuracy by multirotor, an improved experimental method is required. This could be achieved by flying in more laminar wind conditions, where both anemometers are oriented parallel to the primary wind direction and the multirotor hovers upwind or downwind of them. This method has the advantage of lower cost and a purely field-based data-driven approach, but is subject to the changing wind conditions at location and time of data collection. Wind tunnel testing with controlled and known wind conditions could yield improved results, however this decreases the accessibility as only researchers with access to wind tunnels that can facilitate drone flights can replicate this work. Hardware changes are required moving from indoors to outdoor flight, as GPS sensors only work with a sufficient view of the sky.

This work without improvement can be directly applied to emissions measurement and gas source localization, only requiring demonstration by coupling implicit wind estimation with an onboard methane or other pollutant concentration sensor. This work was centered around post-processing methods, where the flight data was saved, downloaded and processed after the test flight to produce wind estimations. Methods like gas source localization require a real-time implementation, where a previously trained ML model is implemented within the drone hardware to produce wind measurements. Then, this real-time signal is combined with emissions concentration measurement to make automatic decisions about moving towards and finding an unknown gas source.

Thus far, this work has not considered the sensitivity of the models to changing the plant parameters or using a completely new multirotor. Replicating this work with a new multirotor would require a new set of training data, as there is a completely new drag-airspeed relationship for models to learn depending on the drones size and shape. It is unknown however how changing the plant parameters such as mass, or changing a sensor source will influence a trained ML model on a known source. Changing a sensor source is expected to have minimal effect as these models have been trained with respect to relevant units rather than raw sensor output, meaning so long as appropriate conversions are made sensor modifications wont have a significant effect on performance. Changes to aggregate plant parameters such as the mass, or the propeller thrust constant are built into the hybrid approach where they are known and used to produced drag estimates. This improves the robustness of hybrid models relative to pure ML approaches, however this has not been well validated. Significant changes to the drag-airspeed relationship are expected to have the most significant changes to model performance. Future work would address how changing plant parameters at varying scales affects the accuracy of the pre-trained models. This inability to easily adapt to a changing system is a fundamental drawback of ML based modelling, where physics-based modelling has the transparency to easily

make modifications.

Bibliography

[1] S. Prudden et al. "Measuring wind with Small Unmanned Aircraft Systems". In: *Journal of Wind Engineering and Industrial Aerodynamics* 176 (2018), pp. 197–210. ISSN: 0167-6105. DOI: https://doi.org/10.1016/j.jweia.2018.03.029. URL: https://www.sciencedirect.com/science/article/pii/S016761051730942X.

[2] E. G Nisbet et al. "Methane Mitigation: Methods to Reduce Emissions, on the Path to the Paris Agreement". In: *Review of Geophysics* (2020).

[3] Adil Shah et al. "A Near-Field Gaussian Plume Inversion Flux Quantification Method, Applied to Unmanned Aerial Vehicle Sampling". In: *Atmosphere* 10.7 (2019). ISSN: 2073-4433. DOI: 10.3390/atmos10070396. URL: https://www.mdpi.com/2073-4433/10/7/396.

[4] Johan H. Scheller, Mikhail Mastepanov, and Torben R. Christensen. "Toward UAV-based methane emission mapping of Arctic terrestrial ecosystems". In: *Science of The Total Environment* 819 (2022), p. 153161. ISSN: 0048-9697. DOI: https://doi.org/10.1016/j.scitotenv.2022.153161. URL: https://www.sciencedirect.com/science/article/pii/S0048969722002510.

[5] Yibo Sun et al. "A UAV-Based Eddy Covariance System for Measurement of Mass and Energy Exchange of the Ecosystem: Preliminary Results". In: *Sensors* 21.2 (2021). ISSN: 1424-8220. DOI: 10.3390/s21020403. URL: https://www.mdpi.com/1424-8220/21/2/403.

[6] Jacob Shaw et al. "Methods for quantifying methane emissions using unmanned aerial vehicles: a review". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 379 (2021). URL: https://doi.org/10.1098/rsta.2020.0450.

[7] Javier Burgués et al. "Smelling Nano Aerial Vehicle for Gas Source Localization and Mapping". In: *Sensors* 19.3 (2019). ISSN: 1424-8220. DOI: 10.3390/s19030478. URL: https://www.mdpi.com/1424-8220/19/3/478.

[8] Patrick P. Neumann et al. "Gas source localization with a micro-drone using bio-inspired and particle filter-based algorithms". In: *Advanced Robotics* 27.9 (2013), pp. 725–738. DOI: 10.1080/01691864.2013.779052. eprint: https://doi.org/10.1080/01691864.2013.779052. URL: https://doi.org/10.1080/01691864.2013.779052.

[9] Chiara Ercolani and Alcherio Martinoli. "3D Odor Source Localization using a Micro Aerial Vehicle: System Design and Performance Evaluation". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 6194–6200. DOI: 10.1109/IROS45743.2020.9341501.

[10] Norman Wildmann, Sarah Bernard, and Jens Bange. "Measuring the local wind field at an escarpment using small remotely-piloted aircraft". In: *Renewable Energy* 103 (Nov. 2016). DOI: 10.1016/j.renene.2016.10.073.

[11] Matthew Marino et al. “An Evaluation of Multi-Rotor Unmanned Aircraft as Flying Wind Sensors”. In: *International Journal of Micro Air Vehicles* 7.3 (2015), pp. 285–299. DOI: 10.1260/1756-8293.7.3.285. eprint: https://doi.org/10.1260/1756-8293.7.3.285. URL: https://doi.org/10.1260/1756-8293.7.3.285.

[12] Jing Shi and Ergin Erdem. “Chapter 3 - Estimation of Wind Energy Potential and Prediction of Wind Power”. In: *Wind Energy Engineering*. Ed. by Trevor M. Letcher. Academic Press, 2017, pp. 25–49. ISBN: 978-0-12-809451-8. DOI: https://doi.org/10.1016/B978-0-12-809451-8.00003-5. URL: https://www.sciencedirect.com/science/article/pii/B9780128094518000035.

[13] Justyna Zalewska, Krzysztof Damaziak, and Jerzy Malachowski. “An Energy Efficiency Estimation Procedure for Small Wind Turbines at Chosen Locations in Poland”. In: *Energies* 14.12 (2021). ISSN: 1996-1073. DOI: 10.3390/en14123706. URL: https://www.mdpi.com/1996-1073/14/12/3706.

[14] Vahram Stepanyan and Kalmanje S. Krishnakumar. “Estimation, Navigation and Control of Multi-Rotor Drones in an Urban Wind Field”. In: *AIAA Information Systems-AIAA Infotech @ Aerospace*. DOI: 10.2514/6.2017-0670. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2017-0670. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2017-0670.

[15] Jack Langelaan, Nicholas Alley, and James Neidhoefer. “Wind Field Estimation for Small Unmanned Aerial Vehicles”. In: *Journal of Guidance, Control, and Dynamics* 34 (July 2011), pp. 1016–1030. DOI: 10.2514/1.52532.

[16] Javier Moyano Cano. “Quadrotor UAV for wind profile characterization”. MA thesis. 2013.

[17] S. Martin, J. Bange, and F. Beyrich. “Meteorological profiling of the lower troposphere using the research UAV ”M^2AV Carolo””. In: *Atmospheric Measurement Techniques* 4.4 (2011), pp. 705–716. DOI: 10.5194/amt-4-705-2011. URL: https://amt.copernicus.org/articles/4/705/2011/.

[18] Sam Prudden et al. “A Flying Anemometer Quadrotor: Part 1”. In: Oct. 2016.

[19] Tomoya Shimura et al. “Estimation of Wind Vector Profile Using a Hexarotor Unmanned Aerial Vehicle and Its Application to Meteorological Observation up to 1000 m above Surface”. In: *Journal of Atmospheric and Oceanic Technology* 35.8 (2018), pp. 1621–1631. DOI: 10.1175/JTECH-D-17-0186.1. URL: https://journals.ametsoc.org/view/journals/atot/35/8/jtech-d-17-0186.1.xml.

[20] W. Thielicke et al. “Towards accurate and practical drone-based wind measurements with an ultrasonic anemometer”. In: *Atmospheric Measurement Techniques* 14.2 (2021), pp. 1303–1318. DOI: 10.5194/amt-14-1303-2021. URL: https://amt.copernicus.org/articles/14/1303/2021/.

[21] Derek Hollenbeck et al. “Pitch and Roll Effects of On-board Wind Measurements Using sUAS”. In: *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2019, pp. 1249–1254. DOI: 10.1109/ICUAS.2019.8797707.

[22] Patrick Neumann and Matthias Bartholmai. “Real-time wind estimation on a micro unmanned aerial vehicle using its inertial measurement unit”. In: *Sensors and Actuators A: Physical* 235 (Nov. 2015), pp. 300–310. DOI: 10.1016/j.sna.2015.09.036.

[23] Patrick Neumann. “Gas Source Localization and Gas Distribution Mapping with a Micro-Drone”. PhD thesis. June 2013.

[24] Jia-Ying Wang et al. "A Wind Estimation Method with an Unmanned Rotorcraft for Environmental Monitoring Tasks". In: *Sensors* 18.12 (2018). ISSN: 1424-8220. DOI: 10.3390/s18124504. URL: https://www.mdpi.com/1424-8220/18/12/4504.

[25] Teodor Tomić et al. "The flying anemometer: Unified estimation of wind velocity from aerodynamic power and wrenches". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 1637–1644. DOI: 10.1109/IROS.2016.7759264.

[26] Ross T. Palomaki et al. "Wind Estimation in the Lower Atmosphere Using Multirotor Aircraft". In: *Journal of Atmospheric and Oceanic Technology* 34.5 (2017), pp. 1183–1191. DOI: 10.1175/JTECH-D-16-0177.1. URL: https://journals.ametsoc.org/view/journals/atot/34/5/jtech-d-16-0177.1.xml.

[27] Javier González-Rocha et al. "Wind Profiling in the Lower Atmosphere from Wind-Induced Perturbations to Multirotor UAS". In: *Sensors* 20.5 (2020). ISSN: 1424-8220. DOI: 10.3390/s20051341. URL: https://www.mdpi.com/1424-8220/20/5/1341.

[28] Magdalena Simma, Håvard Mjøen, and Tobias Boström. "Measuring Wind Speed Using the Internal Stabilization System of a Quadrotor Drone". In: *Drones* 4.2 (2020). ISSN: 2504-446X. DOI: 10.3390/drones4020023. URL: https://www.mdpi.com/2504-446X/4/2/23.

[29] Sam Allison, He Bai, and Balaji Jayaraman. "Wind estimation using quadcopter motion: A machine learning approach". In: *Aerospace Science and Technology* 98 (2020), p. 105699. ISSN: 1270-9638. DOI: https://doi.org/10.1016/j.ast.2020.105699. URL: https://www.sciencedirect.com/science/article/pii/S1270963819324034.

[30] Liyang Wang, Gaurav Misra, and Xiaoli Bai. "A K Nearest Neighborhood-Based Wind Estimation for Rotary-Wing VTOL UAVs". In: *Drones* 3.2 (2019). ISSN: 2504-446X. DOI: 10.3390/drones3020031. URL: https://www.mdpi.com/2504-446X/3/2/31.

[31] David Crowe et al. "Two Supervised Machine Learning Approaches for Wind Velocity Estimation Using Multi-Rotor Copter Attitude Measurements". In: *Sensors* 20.19 (2020). ISSN: 1424-8220. DOI: 10.3390/s20195638. URL: https://www.mdpi.com/1424-8220/20/19/5638.

[32] Wen-Hua Chen et al. "Disturbance-Observer-Based Control and Related Methods—An Overview". In: *IEEE Transactions on Industrial Electronics* 63.2 (2016), pp. 1083–1095. DOI: 10.1109/TIE.2015.2478397.

[33] Teppo Luukkonen. "Modelling and control of quadcopter". In: *Independent research project in applied mathematics, Espoo* 22 (2011), p. 22.

[34] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[35] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: 10.48550/ARXIV.1412.6980. URL: https://arxiv.org/abs/1412.6980.

[36] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[37] CE. Rasmussen and CKI. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. Cambridge, MA, USA: MIT Press, Jan. 2006, p. 248.

[38] Jie Wang. *An Intuitive Tutorial to Gaussian Processes Regression*. 2020. DOI: 10.48550/ARXIV.2009.10862. URL: https://arxiv.org/abs/2009.10862.

[39] *Performance data*. Mar. 2021. URL: https://www.apcprop.com/technical-information/performance-data/.

[40] Steven Zimmerman et al. "Wind estimation by multirotor dynamic state measurement and machine learning models". In: *Measurement* 198 (2022), p. 111331. ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2022.111331. URL: https://www.sciencedirect.com/science/article/pii/S0263224122005693.

[41] Seyed Amin Bagherzadeh. "Nonlinear aircraft system identification using artificial neural networks enhanced by empirical mode decomposition". In: *Aerospace Science and Technology* 75 (2018), pp. 155–171. ISSN: 1270-9638. DOI: https://doi.org/10.1016/j.ast.2018.01.004. URL: https://www.sciencedirect.com/science/article/pii/S127096381730514X.

[42] Michael A. Neilson. *Neural Networks and Deep Learning*. Determination Press, 2015.

[43] Derek Hollenbeck, Demitrius Zulevic, and Yangquan Chen. "Advanced Leak Detection and Quantification of Methane Emissions Using sUAS". In: *Drones* 5.4 (2021). ISSN: 2504-446X. DOI: 10.3390/drones5040117. URL: https://www.mdpi.com/2504-446X/5/4/117.

[44] W. Thielicke et al. "Towards accurate and practical drone-based wind measurements with an ultrasonic anemometer". In: *Atmospheric Measurement Techniques* 14.2 (2021), pp. 1303–1318. DOI: 10.5194/amt-14-1303-2021. URL: https://amt.copernicus.org/articles/14/1303/2021/.

[45] Julia Ling, Reese Jones, and Jeremy Templeton. "Machine learning strategies for systems with invariance properties". In: *Journal of Computational Physics* 318 (2016), pp. 22–35. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2016.05.003. URL: https://www.sciencedirect.com/science/article/pii/S0021999116301309.

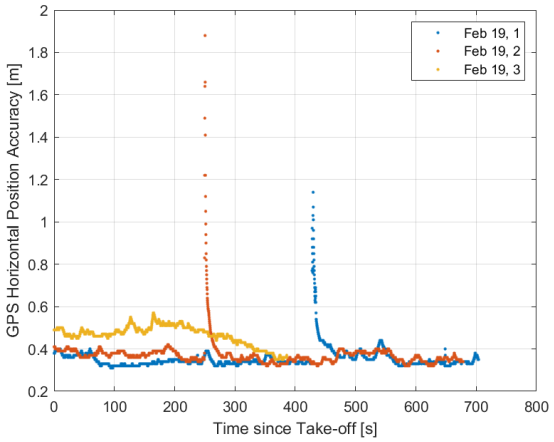
[46] Jules Matz et al. "Parameter identification for nonlinear models from a state-space approach". In: *IFAC-PapersOnLine* 53 (Jan. 2020), pp. 13910–13915. DOI: 10.1016/j.ifacol.2020.12.905.

[47] Lars Morten Bardal and Lars Sætran. "Spatial correlation of atmospheric wind at scales relevant for large scale wind turbines". In: *Journal of Physics: Conference Series* 753 (Sept. 2016), p. 032033. DOI: 10.1088/1742-6596/753/3/032033.

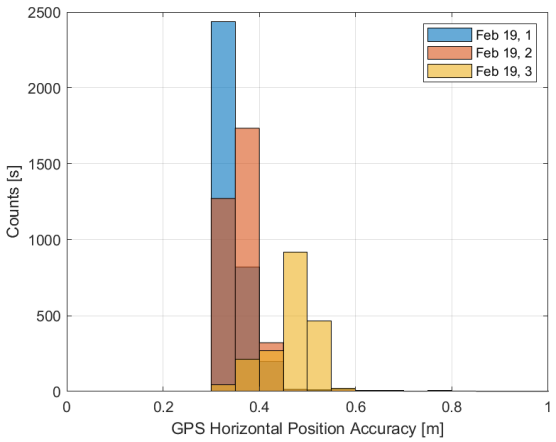
Appendix A

RTK GPS Self-reported accuracy

Examples of RTK GPS self-reported accuracy are shown for 3 flights on February 19th 2022 in Figures A.1 and A.2.

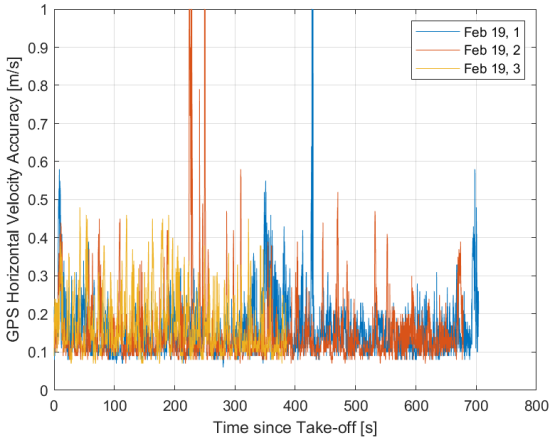


(a) Time series

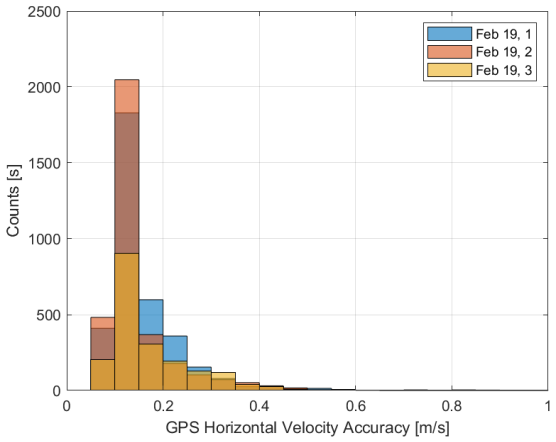


(b) Histogram

Figure A.1: Self-reported RTK GPS horizontal position accuracy



(a) Time series



(b) Histogram

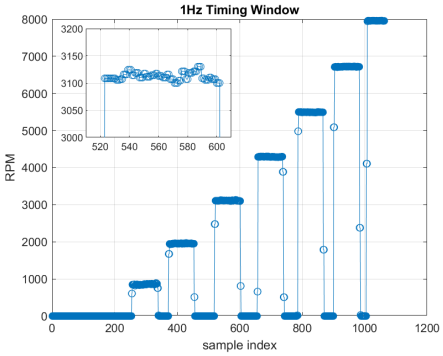
Figure A.2: Self-reported RTK GPS horizontal velocity accuracy

Appendix B

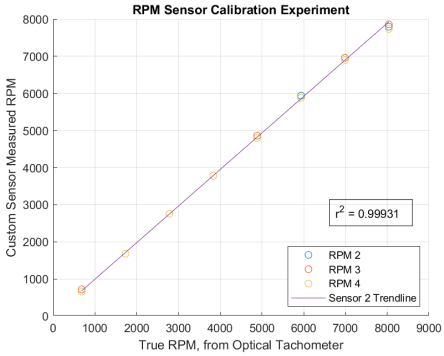
RPM Sensor Design and Calibration

To validate the motor speed sensors to ensure they functioned properly and took accurate measurements, a laboratory experiment was run comparing the motor speed readings to a handheld optical tachometer. The motors were throttled up at discrete positions from 0 to 100%. It was found that there was excellent agreement between the custom sensor and the off-the-shelf sensor readings which determined that no offsetting was required. It was observed however that there a was a variance in the motor speed measurement at each throttle position. As the optical motor speed sensor did not provide instantaneous measurement, only averaged, it could not be used to assess this error. As well, it is unclear whether this variance is due to errors in the motor speed measurement or simply variance in the true motor speed as controlled by the drone. To consider whether this error is important, we assume that it is completely due to measurement error. Figure B.2 compares how the range of each motor speed measurement varies as the motor speed increases, which is converted to an estimated thrust error using a linear approximation of (2.7). Since this error is on the order of 0.2 N, and the thrust signals being measured are on the order of 30 N this is considered negligible. This only represents the worst case accuracy that has been validated to using the instruments available, it could be the case that this variance is partially due to a true change in motor speed.

To compute motor speed from rising and falling edge signal, the pulse counting or pulse timing method can be used. The pulse counting method functions by counting the number of rising and falling edges observed over a fixed amount of time, and then the speed converted from this to a meaningful value like RPM. The pulse timing method works by observing the time difference between two consecutive pulses to compute the speed. Both of these methods are similar but have different implications for measurement quality at different speeds. When using the pulse counting method, the measurement resolution is limited by each one pulse per timing window which is constant for changing motor speed. The measurement resolution of the pulse timing method depends on the clock frequency used to measure the time difference, and increases quadratically with increasing motor speed. Because of this the pulse timing method is generally used for low-speed applications and pulse counting for high speed applications. Figure B.3 shows a comparison of these two methods with different length timing windows for the pulse counting method. This motor speed measurement resolution is also converted into a thrust estimation resolution using a local linear approximation of (2.7). This shows that the pulse counting method is preferred for this application which is expected, as well as determines the thrust resolution to be at most 0.05 N at the maximum expected operating speed of 6000 RPM which is negligible. The final settings used for the motor speed sensor were using the pulse counting method with a measurement frequency of 4Hz, which is set to match the 2-axis sonic anemometer.

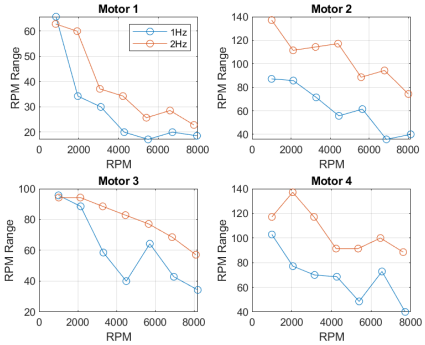


(a) Timeseries plot of motor speed sensor calibration tests.

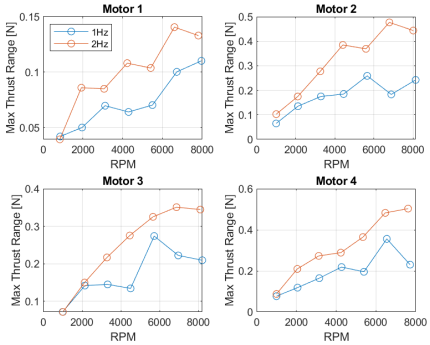


(b) Scatterplot of motor speed readings and handheld optical tachometer readings.

Figure B.1: Benchtop calibration experiments used for testing custom motor speed sensors.

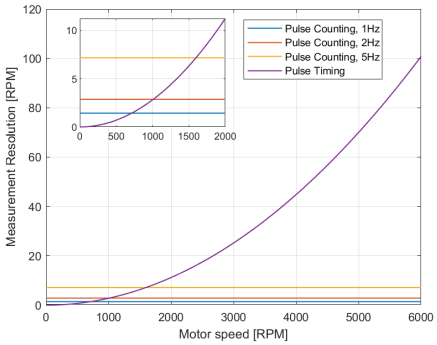


(a) Max range of each motor speed measurement, in RPM.

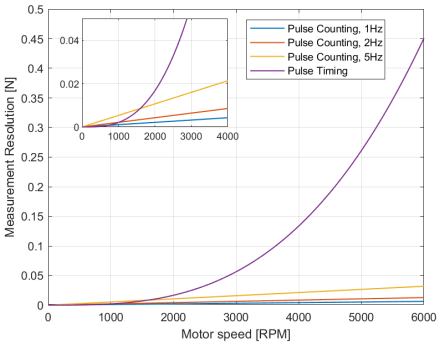


(b) Max range of each motor speed measurement, converted to thrust.

Figure B.2: Determination of instantaneous motor speed accuracy, in both RPM and predicted thrust.



(a) Motor speed, RPM



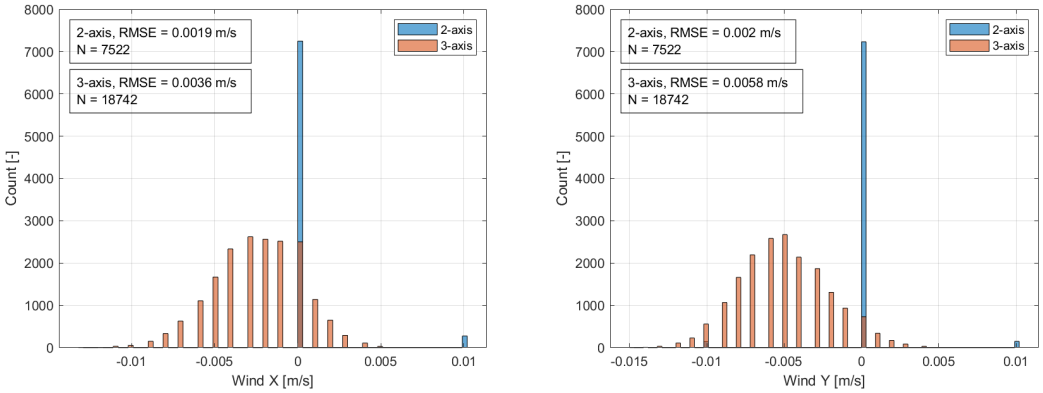
(b) Estimated Thrust, N

Figure B.3: Comparison of measurement resolution in motor speed and estimated thrust.

Appendix C

Anemometer Accuracy

According to the sonic anemometer datasheets, the reported accuracy is ±2% at 12 m/s. It is unknown however how this accuracy changes as the wind speed approaches zero, which is primarily of interest as the wind speeds measured are in the range 0 - 5 m/s. To consider this, a simple experiment was conducted where both the 2-axis and 3-axis anemometers are held within plastic bags, which corresponds to true zero wind speed, and the measurements recorded for approximately 30 minutes. Figure C.1 shows the measurements recorded, and the RMSE of each instrument. The measurement error in this case is comparable to the measurement resolution of each intstrument, where the 2-axis anemometer has a resolution of 0.01 m/s and the 3-axis anemometer a resolution of 0.001 m/s. Even though the datasheet for the 3-axis anemometer reports the resolution of 0.01 m/s, this device is able to be set in "high resolution" mode which increases this. In general, the errors measured at this zero wind case are considered negligible, as they are orders of magnitude less than the estimated spatial variation of the wind, which is discussed in Chapter 7.



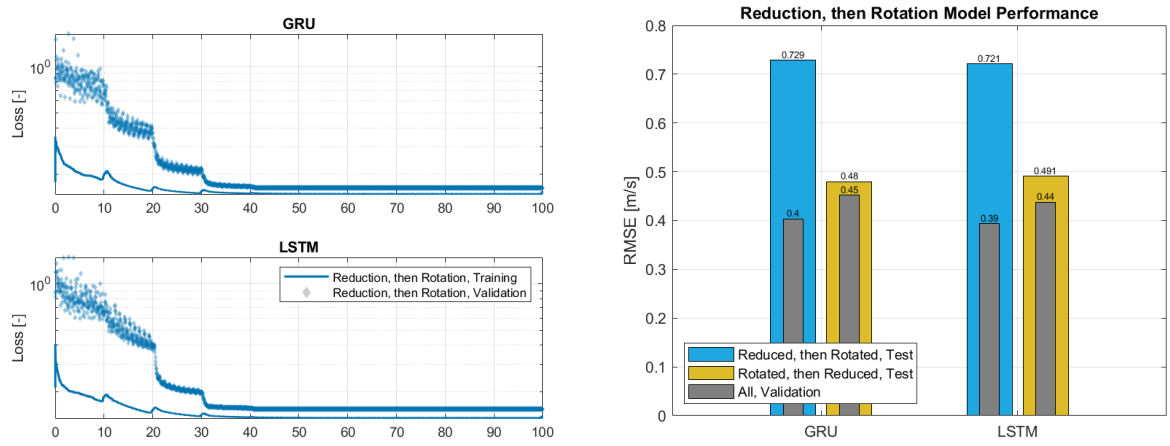
(a) X Wind (b) Y Wind

Figure C.1: Measurement histograms for zero wind test case.

Appendix D

Variations of Data Rotation and Reduction

As supplementary material to Chapter 6, variations of applying data rotation and reduction are considered here. Originally presented was models created by first applying data rotation, and then grid-based data reduction. However, one could also consider a model based on applying reduction first on the original data set and then rotation. Such a model is considered here, where the models are trained using the same structure, hyperparameters and learning schedule as described in Chapter 6. Figure D.1 shows a comparison of achieved model performance and training histories. Generally these models achieve a better validation performance but significantly worse testing performance, which is why they were omitted.



(a) Training history

(b) Test / Validation

Figure D.1: Comparison of model performance when applying reduction first, then rotation.

Appendix E

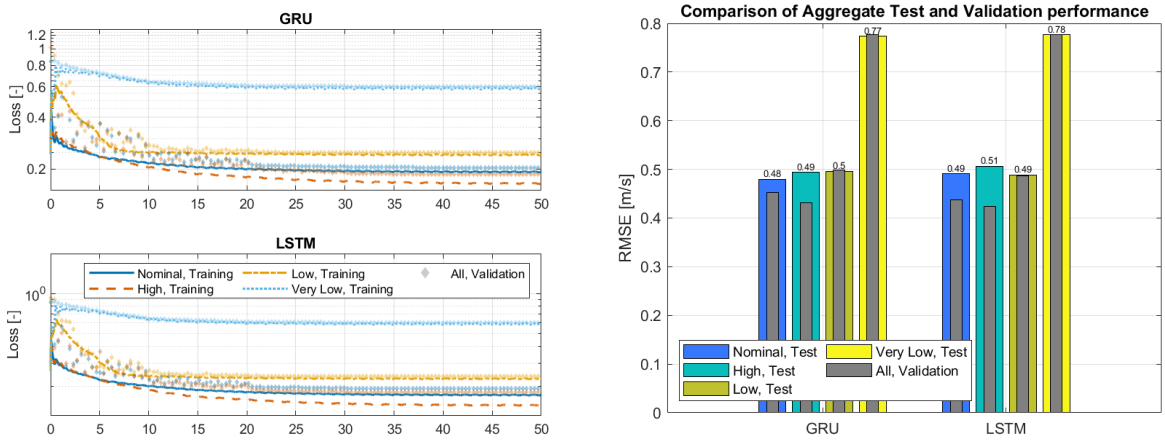
Learning Rate Considerations

The inital learning rate was varied to see if this training hyperparameter significantly affected performance. The variations considered are shown in Table E.1.

Abbreviation	Initial Learning Rate	Drop Schedule
Nominal	1×10^{-3}	80% / 10 Epochs
High	1×10^{-2}	80% / 10 Epochs
Low	1×10^{-4}	80% / 10 Epochs
Very Low	1×10^{-5}	80% / 10 Epochs

Table E.1: Summary of learning rate variations and abbreviations for reference.

The training history for each learning rate variation and aggregate performance metrics are shown in Figures E.1.



(a) Training history

(b) Test / Validation

Figure E.1: Results of training learning rate variations, with training history and aggregate performance.