Project No: A2113-191

NANYANG TECHNOLOGICAL UNIVERSITY

Optimization of High Performance UAV

Submitted by: Foo Ren Xiang

Matriculation Number: U1522802D

Supervisor: A/P Lim Meng Hiot

Project No: A2113-191

School of Electrical & Electronic Engineering

A final year project report presented to the Nanyang Technological University

in partial fulfilment of the requirements of the degree of

Bachelor of Engineering

2020

Table of Contents

Abstract i
Acknowledgements ii
List of Figures iii
Chapter 1 Introduction .....
1.1 Motivations 6
1.2 Objectives and Scope 6
Chapter 2 Literature Review 8
2.1 Current high-speed mUAV designs 8
2.2 Low drag aerodynamics development in aircraft/motorsports 9
2.3 Traction and Launch Control Systems in Formula One 11
2.4 UAV autonomous control systems 12
2.5 Machine Learning Pipeline 13
2.6 Machine Learning Regressors 14
2.7 Recurrent Neural Networks/Long-Short Term Memory Networks 15
2.8 Data Visualisation 16

Chapter 3 Finding the limits of mUAV Performance 18
3.1 Objective 18
3.2 Field Testing 19
3.3 Test Methodology 23
3.4 Results 27

Chapter 4 Optimisation of mUAV Aircraft Design 30
4.1 Objective 30
4.2 Design Process 31
4.3 Manufacturing Process 53

Chapter 5 Optimisation of mUAV Aircraft Design 56
5.1 Objective 56

5.2 URLC Design 58
5.3 Manufacturing Process 53
Chapter 6 Prototype mUAV Field Testing 90
6.1 Test Findings 90
6.2 Design Improvements 99

Chapter 7 Conclusions and Future Work 100
7.1 Conclusions 100
7.1 Recommendations in Future Work 100

Reflection on Learning Outcome Attainment 100
References 101
Appendix 108

Abstract

To achieve high flight speeds, multi-rotor Unmanned Aerial Vehicles (UAV) require sufficient power to combat exponentially increasing aerodynamic drag. To reduce aerodynamic drag, the UAV must be carefully designed to reduce its effective frontal area and drag coefficient. Part of this project investigates such UAV design.

Most UAVs also rely on Lithium-Polymer batteries (LiPos) for their energy store. Limited power delivery from LiPos becomes a limiting factor when operating at the speed performance limits of a UAV. LiPos will overheat and fail when too much continuous power is drawn. To optimise the UAV's burst endurance at sustained top-end speeds, the pilot must constantly and rapidly adjust the throttle input to demand maximum power while avoiding overheating the LiPos. His reaction time introduces an inherent delay into the system. To eliminate this delay, this project proposes a method, UAV Rolling Launch Control (URLC) which implements autonomous throttle control. Such control is achieved by streaming LiPos and powertrain data in real-time to a deep learning model that outputs appropriate corrections to the UAV's throttle. To prepare the model, Principal Component Analysis (PCA) is used to analyse the UAV flight logs, which contain a multitude of powertrain and LiPos sensor data. The Principal Components derived are then used in training to derive an optimal control model.

Acknowledgements

I would like to express my sincere thanks and gratitude to the following people. Without them, I would not have reached where I am today.

My parents and extended family for their strong support and encouragement when the going gets tough. An extra mention to my mother who encouraged me to explore my passion in engineering.

A/P Prof Lim Meng Hiot, my mentor throughout my University journey, for the countless encouragement, guidance and support.

OHR400 teammates: Zhong Bingchen, Pan Zexu and Victor Vong, who believed in my idea and ambition for developing the mUAV system, and worked tirelessly together to help materialise and test the mUAV concept.

Dr Felis Dwiyasa, for her mentorship in my robotics work and having discussions of machine learning that helped me to understand the subject better, in the course of formulating the solution proposed in this project.

Tee Chin Kiat, for the countless electronics hacking sessions and teaching me so much about good mUAV design and how to pilot them

I would also like to thank Kx Systems and the kdb+ community for providing the training sessions and guidance that helped me be fluent in the uniquely terse and effective q scripting language.

Foo Ren Xiang

May 2020

List of Figures

Figure 1.1: LiPo With Thermal Failure 3
Figure 2.1 Predecessor of the XLR-1, the VRX-190 8
Figure 2.2: World's fastest tilt-rotor. 9
Figure 2.3: SBG Race Watch. 16
Figure 2.4: TIBCO Spotfire. 17
Figure 3.1: Test mUAV. 19
Figure 3.2: Test mUAV, low aerodynamic seal. 20
Figure 3.3: Test mUAV, highaerodynamic seal 20
Figure 3.4: FPV View with real-time data displayed on on-screen display (OSD) 23
Figure 3.5: Test flight path 26
Figure 3.6: Logging of flight conditions 26
Figure 3.7: PLA warping on mUAV 27
Figure 3.8: Sample of GPS blackbox log 28
Figure 3.9: Sample of PID blackbox log 29
Figure 3.10: GPS Speed Run Analyser 29
Figure 3.11: Samples of summarised flight data collected 29
Figure 4.1: ARX-V2 frame 32
Figure 4.2: ARX-V2 frame reverse engineered in CAD 32
Figure 4.3 mUAV layout with electronics in front 32
Figure 4.4 Side profile view of mUAV 34
Figure 4.5 Side profile view of mUAV 35
Figure 4.6 Side profile view of mUAV 36
Figure 4.7 Side plate of LiPo carrier 37
Figure 4.8 Structural sections of side plate 37
Figure 4.9 Structural sections of back plate 37
Figure 4.10 Optimised side plate 38
Figure 4.11 Optimised back plate 38
Figure 4.12 Aeroelastic tail 38
Figure 4.13 CNC DXF Data for carbon plate manufacturing 39
Figure 4.14 S-duct, mUAV side view 41
Figure 4.15 S-duct, mUAV front view 41
Figure 4.16 S-duct, mUAV rear view 41
Figure 4.17 Other aerodynamic devices 42
Figure 4.18 Cross sections guides used to loft the bodywork 45
Figure 4.19 CFD results of the mUAV used for initial field testing 45

Figure 4.20 CFD results of the new mUAV 45
Figure 4.21 Clean main geometry 46
Figure 4.22 Tail with aggressive taper 46
Figure 4.23 S duct 47
Figure 4.24 Multiple stabilisers 47
Figure 4.25 Stabilisers at the bottom of the body 48
Figure 4.26 Turning vanes in front of arms 48
Figure 4.27 Center of mass, front profile view 50
Figure 4.28 Center of mass, side profile view 50
Figure 4.29 Final mUAV design, side view 51
Figure 4.30 Final mUAV design, without bodywork 51
Figure 4.31 View of electronics 52
Figure 4.32 Bottom view of LiPo carrier 52
Figure 4.33 mUAV at Red Bull Racing Formula One Team HQ 53
Figure 5.1 Speed and throttle behaviour during mUAV high speed pass 58
Figure 5.2 GPS Sampling Frequency vs GPS speed 60
Figure 5.3 GPS sampling frequency, GPS speed vs time 61
Figure 5.4 GPS Speed, Motor Output, LiPo Voltage vs Time 62
Figure 5.5 Sample counts vs GPS speed 63
Figure 5.6 Regression prediction of LiPo voltage 64
Figure 5.7 Regression prediction of GPS speed 65
Figure 5.8 Entity-Relationship Diagram of Database 67
Figure 5.9 Data Pre-processing for ML models 69
Figure 5.10 Machine Learning Model Training Pipeline 71
Figure 5.11 A simplified version of the model training cycle 71
Figure 5.12 Support Vector Regression with PCA on input data 73
Figure 5.13 Support Vector Regression with PCA on input data 73
Figure 5.14 Adaboost regression for GPS speed prediction 75
Figure 5.15 ELM regressor for LiPo Voltage prediction 75
Figure 5.16 Accuracy of Gaussian Process Regression for GPS speed prediction on small dataset (~300 samples) 76
Figure 5.17 Gradient Boosting GPS speed prediction on small dataset (~300 samples) . 75
Figure 5.18 LSTM performance on real flight data input. Orange denotes training data, green denotes test data, blue denotes LSTM prediction 79
Figure 5.19 LSTM performance on synthesized data. Blue denotes test data; Orange denotes LSTM prediction 80
Figure 5.20 LSTM performance with standard scalar on input sequence 81
Figure 5.21 A direct scalar based on the throttle encoding function was written, providing an accurate output scale. 81
Figure 5.21 Topology of networked machines running URLC as a Web Application 84
Figure 5.22 Flight path with pilot input and LiPo voltage in Spotfire Analyst 86
Figure 5.23 Regression model of data from telemetry in Spotfire Analyst 86
Figure 5.24 Real-time data distribution mapped to GPS speed ranges in Tableau 87
Figure 5.25 Real-time data distribution mapped to GPS Latitude in Tableau 87
Figure 5.26 Real-time URLC Throttle Prediction Performance evaluation in Tableau 88
Figure 5.27 URLC Application Topology 89
Figure 6.1 Test Prototype mUAV at Old Holland Road Flying Field 90

Figure 6.2 mUAV on launchpad 90
Figure 6.3 preparing for test mUAV landing 90
Figure 6.4 Max acceleration of 3.5G 93
Figure 6.5 Max speed of 178km/h 93
Figure 6.6 Max deceleration of 14.1G from aerodynamic drag 94
Figure 6.7 mUAV after crash 95
Figure 6.7 mUAV after crash 95
Figure 6.8 Crack at bottom of nose cone 96
Figure 6.9 Tail electronics unscathed. Right tail strut fractured 97
Figure 6.10 No cracks on tail bodywork despite impact with ground 98

Chapter 1
Introduction

The industrial applications of Unmanned Aerial Vehicles (UAV) have gained traction in the past decade. Due to its mechanical simplicity, usage of commodity electronics and software development by the open source community, the multi-rotor UAV (mUAV) have proven itself to be an inexpensive and reliable aerial photography platform to operate for both commercial and amateur users alike.

Another vital trait the mUAV possess is high manoeuvrability, made possible by its differential-thrust-based powertrain system. Instead of relying on air foils and control surfaces to generate lift and change its velocity vector, mUAVs manoeuvre by controlling thrust generated by each propeller independently. Compared to a fixed wing UAV platform, the mUAV have shorter turning radius, maintains good control responsiveness regardless of airspeed and possess acceleration vectors orthogonal to its velocity vector. They can also hover and fly backwards, just like a traditional helicopter platform.

Recent development of multi-rotor UAVs (mUAVs) have made them sufficiently small for safe handling due to small rotors and low mass. Acceleration and top speed capabilities of these machines have also increased with the use of efficient motors and inexpensive LiPo technology. These mUAVs are capable of superseding full scale helicopters for motorsports media coverage. In top tier motorsports, such as Formula One and World Endurance Championship, racing cars can reach speeds of up to 370km/h. [1] Being much smaller and lighter than helicopters, mUAVs equipped with lightweight, high-definition cameras can chase race cars at speed with lower risks and operating cost. This allows for more dynamic and closer up shots that were previously impossible.

The high thrust to weight ratio (TWR) of mini mUAVs (up to 30cm in dimension) allows them to accelerate quickly and maintain top end speeds. Flight data analysis of the high performance mUAV (Static TWR 13:1) used in this research indicates it possesses a peak acceleration of around 6G (58m s^{-2}) and top speeds of up to 298km/h (82.8m s^{-1}).

However, the mUAV is not a perfect piece of technology – its endurance and top speed performance is limited by its energy store technology, the Lithium Polymer Battery (LiPo).

Limited power delivery from LiPos becomes the main limiting factor when operating at the speed performance limits of a UAV. LiPos overheat and fail when too much continuous power is drawn. LiPos have 2 power discharge modes: constant and burst. The power discharge limits are usually expressed as 'C' ratings by LiPo manufacturers. Maximum power discharge is calculated based on the capacity of the battery in Ampere-Hours and its 'C' rating (i.e. the research mUAV is equipped with a 1.5Ah 18.5V (5S1P) LiPo with a C rating of 80C (continuous), 160C (burst), can theoretically deliver 2.2kW continuously and 4.4kW for up to 10 seconds. When operating in burst discharge mode, the battery will start to heat up rapidly and can reach temperatures of up to 70°C. When LiPos overheat and thermally fail, their lithium ion electrolytes on both the anodes and cathodes decompose [1], causing the cells to puff to approximately twice its size and have a big increase in internal cell resistance. These damaged cells become dangerous to use due to the formation of lithium metal and oxygen in the cells, a very volatile combination.

Fig 1.1 Identical LiPo models. Puffed LiPo from thermal failure shown in bottom.

The C ratings provided by manufacturers are usually over specified. Under real world operating conditions, it is difficult to achieve the power output specified by the C rating. Under full throttle conditions during acceleration, the motors can demand up to 150A at 18.0V (2.7kW), which the LiPo cannot handle, causing it to puff. Typically, the LiPo's burst discharge performance, vital for high performance mUAV applications, is dependent on various factors. It deviates a fair amount from its continuous and burst discharge model. Internal cell temperature and instantaneous power demand from the motors are two important parameters that govern available maximum power from the LiPo. From test flight data gathered, based on LiPo temperatures taken right before and after each flight, it has been observed that the cell temperatures can increase from 35°C to 70°C when operating at the edge of its power delivery envelope (i.e. maximum LiPo power delivery without encountering thermal failure). During the first 30 seconds of flight, when the cells have not reached the optimal temperature window of 65°C, maximum power delivery is lower, resulting in 30km/h slower top speed of the mUAV. As the cells discharge during flight and heat up, placing them in the optimal temperature window, the mUAV's top speed starts to increase.

Inherent human reaction times are present in other piloted vehicular systems. One area of interest that draws parallel is in motorsports, which also involve high speeds and a need for fast reaction speeds as a result. From 2000 to 2003, Formula One (F1) cars employed Launch Control (LC). F1 races typically involve standing starts, where the goal of the driver is to accelerate as fast as possible from a stationary position to racing speeds, in order to overtake his opponent (and avoid being overtaken) before having to brake for the upcoming corner. In a typical road car, by intuition, the driver would simply floor the throttle to get maximum acceleration by tapping upon the full power output of the engine. However, F1 race cars are equipped with very powerful engines. Full throttle application during a standing start will result in wheelspin, as the engine will provide too much power to the wheels, allowing the tyres to overcome static friction with the road surface, creating tyre slip. Tyre slip is undesirable as it results in less grip with the road surface, creating wheelspin and causing a decrease in maximum acceleration for the car off the start line and a wastage of energy.

Thus, to get a good standing start, F1 drivers had to determine through trial and error during their practice sessions at the track, the correct amount of throttle and clutch to engage when performing their start and commit it to memory. In addition, the optimal power solution would also change with environmental conditions (e.g. friction coefficient of the track, track and tyre temperatures) and car conditions (fuel load, engine power modes). This made the optimal throttle application difficult to precisely and consistently replicate the perfect procedure for each crucial race start.

In order to minimize wheelspin and aid the driver in performing consistently optimal starts, Launch Control (LC) was developed and implemented in F1 cars. It was an automated standing start system that would automatically detect wheelspin and controlled the engine output under full throttle to deliver just enough power for maximum acceleration while ensuring the tyres did not slip. The system proved to be very effective and was subsequently banned by F1's regulatory body as it proved to be an overly effective driver aid; An exceptionally well-designed system could allow a car to overtake a quarter of the field right at the starting grid, bringing the competition from the driver to the engineer. Another variation of launch control, termed Rolling Launch

Control (RLC) can be found in sports car racing that uses rolling race start, where all cars would cruise at low speeds around the track until the start signal is given. The drivers would then attempt to perform full acceleration by precisely controlling their throttle and gear changes to ensure the engine output and revolutions remain in a narrow and optimal range.

Similarly, in mUAVs, to achieve the greatest acceleration and to maintain maximum velocity for a short period of time (i.e. 15 seconds) without damaging the LiPo, the pilot faces the difficult task of constantly monitoring the powertrain conditions in order to determine the correct throttle input at each instant. If too much throttle input is applied, the motors will try to reach an excessive RPM and demand too much power, causing the LiPo to overheat and thermally fail. If insufficient throttle is applied, the mUAV will fall short of the maximum acceleration allowed by LiPo power delivery, failing to reach top speed. This is due to the LiPo's limitation for burst power delivery, which only allows that maximum power delivery for a short time window before having to revert to a lower power output to avoid damage. Hence, in order to optimise the UAV's burst endurance at sustained top-end speeds, the pilot must constantly and rapidly adjust the throttle input to demand maximum power while avoiding overheating the LiPo. His reaction time introduces an inherent delay into the system.

To eliminate this delay, this project proposes a method, UAV Rolling Launch Control (URLC) which implements autonomous throttle control. In this project, an automated throttle control will be implemented on a high speed mUAV system capable of exceeding 300km/h. When activated by the pilot, URLC will take over from the pilot when a high-speed pass during flight is initiated. The goal of URLC is to maximize the quadcopter's peak average velocity over short distances (~100m). As each acceleration period from cruising speed (~150km/h) to max speed (~260-300km/h) only lasts for approximately 2 to 3 seconds, the human reflexes, even from an experienced pilot, are not fast enough to monitor and apply corrections to the rapidly fluctuating powertrain and battery conditions. By automating the throttle control, the quadcopter's throttle control can instantly respond to the rapidly fluctuating conditions

of the LiPo battery under high current. These instantaneous corrections allow power to be delivered optimally.

1.1 Motivations

For a mUAV to keep up with a Formula One car on the circuit, it must possess a similar top speed. Currently, the world's officially fastest mUAV is still too slow, with a top speed of 263km/h[2]. To design a much faster mUAV, aerodynamics, mechanical and power delivery optimization must be carried out to close the speed gap.

mUAVs use Lithium-Polymer batteries (LiPo) as energy storage. Increases in flight speeds are accompanied by exponential increases in LiPo power delivery. Power delivery has become the limiting factor of high speed performance; LiPos fail catastrophically when too much power is drawn by the powertrain. To sustain top-end speeds while avoiding damage, the pilot must constantly adjust the throttle input rapidly to avoiding overheating the LiPo. His limited reaction speed manifests as an inherent system delay.

As high speed mUAVs are still an extremely new technology, literature is lacking in this field. However, many parallel engineering concepts and knowledge can be drawn with well-established Formula One design techniques and considerations, alongside advances in Machine Learning. Upon exiting Formula One as a constructor team, Honda Racing F1 Team has published literature of their research on development of Traction Control and Launch Control systems[3], Aerodynamics Development [4] and modelling of a Formula One car's dynamic performance [5]. The research methodology and design topology for Formula One cars can be translated to mUAV design optimization and hardware/aerodynamic setup, allowing a much faster mUAV to be designed.

1.2 Objectives and Scope

This project investigates the design of a mUAV system capable of chase videography of a Formula One car. In parallel, the mUAV designed will be used to attempt the

Guinness World Record (Fastest ground speed by a battery-powered remote-controlled (RC) quadcopter), which is 265.86km/h at the time of writing of this report.

To perform the task, the mUAV system must achieve high sustained top speeds. This will be the design goal of this project.

There are several means to achieve the design goal:

1. Powertrain component selection
2. Keep the mUAV lightweight to minimize power required to sustain lift
3. Minimize aerodynamic drag to reduce power required to counter it
4. Develop a throttle specific autopilot system that can outperform a human pilot's control

These specific areas of optimization are covered in the work done in this project.

In order to maximize aerodynamic efficiency at high speeds, the mUAV physical design has to achieve a good ratio of small frontal area and drag coefficient, according to the drag equation.

A novel method, UAV Rolling Launch Control (URLC), which implements autonomous throttle control, is also proposed. Such control is achieved by streaming real-time UAV throttle data into a deep learning model that predicts the mUAV's future states and outputs appropriate corrections to the throttle. This deep learning model is trained based on historical flight data generated by the UAV.

As such, an optimized mUAV design and Machine Learning based throttle control system are developed during the course of this project.

Chapter 2
Literature Review

2.1 Current high-speed mUAV designs

In the realm of quadcopter racing drones, the current Guinness World Record for Fastest ground speed by a battery-powered remote-controlled (RC) quadcopter was 265.86km/h at the time of writing of this report. [6]

Fig 2.1 Predecessor of the XLR-1, the VRX-190

Concurrently, a faster drone design, the XLR-1, by Ryan Lademann reportedly reaching 315km/h (195.99mph) is challenging the current Guinness World Record. [7]. The former achieved high speeds using an over-voltage powertrain to achieve high power output [8], which is prone to component failure as it is operating beyond the designed parameters. These speeds are more than twice as fast as commercial delivery drones, which operate at around 161km/h (100mph) by USA's Federal Aviation Administration (FAA) regulations. [9]. These mUAVs rely on electric motors and propellers to create thrust, with LiPo batteries acting as a power source.

Fig 2.2 The fastest tilt-rotor aircraft to set the Guinness World Record for “fastest speed by a remote-controlled tilt-rotor aircraft” has a top speed of 240.6km/h

2.2 Low drag aerodynamics development in aircraft/motorsports

Aerodynamic drag faced by mUAV, aircrafts and race cars can be typically explained by airflow separation of the boundary layer of the vehicle, creating a turbulent wake downstream of the vehicle.

The boundary layer is a phenomenon caused by the viscosity property of fluids. When a body, I.E vehicle, is travelling at high speeds, airflow relative to the body has high velocity, and hence, high kinetic energy. When the high velocity fluid touches the body, which has zero velocity, relative to the body, the fluid encounters friction and slows down. Due to viscosity of the fluid, I.E. air, it forms a skin of fluid, known as the boundary layer, on the body. The velocity of the boundary layer (fluid) increasingly slows down as it moves downstream, due to friction. In regions where the boundary layer has sufficient velocity and kinetic energy, it remains attached to the surface. This flow of fluid remains laminar. If the boundary layer remains laminar

till it reaches downstream of the body, it will converge and form a laminar flow behind the body, creating a clean wake that fills in the vacuum behind the body. This creates a small pressure differential with minimal drag. A smooth and streamlined body geometry will reduce the friction encountered by the boundary layer. These factors are empirically measured and described by the drag coefficient of the body geometry.

However, if the drag coefficient of the body is high, I.E. the fluid forming the boundary layer encounters enough friction and decelerates enough to reach zero or negative velocity downstream, boundary layer reversal occurs, resulting in flow detachment. When flow detachment occurs, the fluid starts to behave in a turbulent manner and no longer conforms to the body. As a result, the volume at the back of the body will not be filled with fluid, causing a vacuum to form. This manifests as a large pressure differential, with a low pressure region at the tail of the vehicle, with a high pressure region at the nose of the vehicle, due to airflow impacting the frontal surface. This pressure differential creates aerodynamic drag[11].

Aerodynamic drag is described by the drag equation [12]**:**

$D = Cd * 0.5*(\rho * v^2) * A$

D: Drag force

Cd: coefficient of drag

ρ: fluid (air) density

v: fluid (air) velocity

In general, the common methods of reducing aerodynamic drag used in vehicular design include:

- Streamline body to reduce drag coefficient, by presenting less friction to the boundary layer, reducing flow separation [13]
- Vortex generators present on the upstream sections of the body. This works by mixing in laminar, high velocity airflow from above the boundary layer into the boundary layer, re-energising it to increase its flow velocity [14]

- Spoilers to change the upstream airflow vector to spoil turbulent flow expected downstream [15]

2.3 Traction and Launch Control Systems in Formula One

In Formula One, Launch Control automatically controls the torque and engine power driving the wheels for standing (stationary) starts. It constantly measures the tire slip of the driven wheels as input of the system, and correspondingly outputs the optimal power and torque required to rotate the wheels to achieve optimal acceleration. To model the problem, the key challenges are precise and low-latency control of torque and engine-clutch coordination. The solution implemented by Honda Racing F1 Team used a classical control systems approach of using feedback from onboard sensors that measure wheel speed, acceleration, that is used to calculate wheel RPM and slip, that is fed back to the rev limiter and torque controller. Tuning of the output parameter of the traction controller, known as target tire slip, was calculated directly from the tire slip feedback as well as indirect features derived from engine speed. The tuned output parameters were implemented as several torque controllers mapping functions, serving different environmental and vehicular states, that map both driver throttle input and acceleration of the vehicle, to a target tire slip value. These torque controller maps are effectively regression models that predict the optimal torque required for the wheels based on driver and IMU inputs. The limitation of the implementation was the time and labour required to perform the measurements and complete the tuning. [3]

URLC aimed to use a similar control feedback topology to calculate and tune the target optimal throttle output for the mUAV. However, it is able to bypass the time and labour requirement by using supervised machine learning regression models trained on historical mUAV flight data, to predict the feedback from the powertrain and target speed. This automated feedback generation is connected to a cascaded time-series supervised deep learning prediction model, that in turns serve as the predict the target

throttle output required to accelerate the mUAV and sustain top speed. The use of machine learning serves to automate the labour intensive manual tuning process, while achieving the same result of automated throttle control.

2.4 UAV autonomous control systems

State of the art UAV control systems equipped with GPS positioning govern in general control the pose of the UAV systems by using estimators that modelling the kinematic behaviour of the body and its motor response. The following properties are captured in the models[16][17]:

- Mass
- Gravitational constant
- Pose
- Angular momentum
- Position in GPS frame

Cascaded PID control loops with feedback from the onboard inertial measurement unit sensor (IMU) and sensors iteratively measure the UAV state and apply corrections, that is finally applied as direct motor control of the UAV to control its thrust pose control.

The estimators used for control are cascaded in the following manner:

- Position
- Speed
- Attitude
- Angular Rate
- Body Dynamics/Angular Acceleration

URLC builds upon this estimator cascade by modelling the LiPo characteristics during high power use, which has an impact of speed estimation behaviour that diverges at high speed and low speed estimation.

2.5 Machine Learning Pipeline

The general machine pipeline can be abstracted to the following[18][19]:

- Data Preparation for model training[20][21]
    - Data cleaning and rescaling for standardisation
    - Exploratory Data Analysis, where data is visualised, and its statistics are quantitatively analysed. This provides insight and an intuitive understanding of the data, in order to formulate the solution.
    - Feature Engineering, Dimensionality Reduction, if required
    - Splitting training data into train and test sets for the model training and validation
- Training of prediction model(s)
- Score and evaluate model prediction performance
    - Redevelop model if accuracy is poor
- Deploy the sufficiently accurate model for prediction on new data.
    - If the model starts to perform poorly on new data, remodel the new data and deploy the new model

Dimensionality reduction speeds up machine learning training and prediction as lesser features have to be transformed by the model. Principal Component Analysis is a commonly used method for unsupervised dimensionality reduction. Principal components are the normalised eigenvectors of the dataset's covariance matrix, arranged in decreasing eigenvalues. They capture covariances of the dataset in orthogonal, independent axes in eigenspace, and form an equal number of features as the original feature count. As the eigenvalues represent the covariance captured by the corresponding eigenvector (principal components), the eigenvectors with small eigenvalues may be dropped, thereby resulting in dimensionality reduction. To transform the data into dimension reduced eigenspace, the original dataset matrix is projected with each eigenvector. [22][23]

In a dataset, not all features may be useful in predicting the output of a sample. There may also be transformations of features that creates a useful input. Feature engineering is performed to solve these issues. It involves manual selection relevant features, based on insights gained during Exploratory Data Analysis and contextual knowledge of the problem being modelled. The dataset may also be rescaled or re-encoded into a new shape that is easier to manipulate during model training. [24][25] Deep feature synthesis may also be used to generate derivative features from original features, from which useful derivative features may be used as inputs for the model. [26]

2.6 Machine Learning Regressors

There are several machine learning regression models currently:

Linear Regression

- Linear regression fits a high dimensional linear hyperplane to the training dataset. Prediction on new data is performed by transposing the new data on the decision hyperplane. [27]

Support Vector Regression

- Support Vector Regression is an extension of Support Vector Machines, which fits decision hyperplanes in augmented pattern space to perform classification. Support vector regressors treats the decision hyperplane as the best fit line for the datapoints. To perform prediction on new data, the new data is projected onto the decision hyperplane and the prediction value is attained.[28]

Gaussian Process Regression

- Gaussian Process Regression is an extension of the Support Vector Regressor, using the Radial Basis Function kernel. The model examines the distribution of all training data and uses that distribution to create inference of the probability distributions of all possible inputs that will fit the initial distribution. This provides good prediction accuracy on future input data, but is computationally expensive. [29][30]

Extreme Learning Machine

- Extreme Learning Machines is a neural network with hidden neurons layer that possess randomly assigned weights that provide good performance for common regression problems. This allows the model to be trained quickly with sufficient accuracy using a single training iteration, as compared to iterative training required with the common gradient descent method used in common neural networks. [31]

Ensemble Learning Methods

- Adaboost
- XGBoost
- Random Forest Regressor
- Stack Generalizer

These ensemble learning methods use clusters of weak regressors to perform a collection of predictions on new data. The multiple predictions on the same data is reduced, usually with a weighted or unweighted voting mechanism, in order to give a stronger prediction result. The weak classifiers may be trained with adaptive weights or node trimming.

Regression models are used to model the kinematics of the mUAV in the URLC machine learning pipeline

2.7 Recurrent Neural Networks/Long-Short Term Memory Networks

Recurrent Neural Networks (RNN) are neural networks that have a feedback mechanism and memory to deal with inputs with sequences. Its output is fed back into its input recursively until the required output length is required, creating memory of past input sequences in the network. However, vanilla RNN suffer from the vanishing gradient descent problem, which is solved by its Long-Short Term Memory(LSTM) variant. This problem is avoided by implementing logic gates functions in the RNN that uses activation functions on the latest input to determine if past output data should be

forgotten or fed back into the latest prediction step. The data being passed into the latest step is always normalised to [0,1], ensuring its magnitude does not decrease with additional loops. [32][33]

URLC uses LSTM to predict optimal future throttle sequences from current throttle sequences.

2.8 Data Visualisation

In Formula One, dashboard software such as SBG Race Watch and TIBCO Spotfire are used to visualise data and output prediction from models.

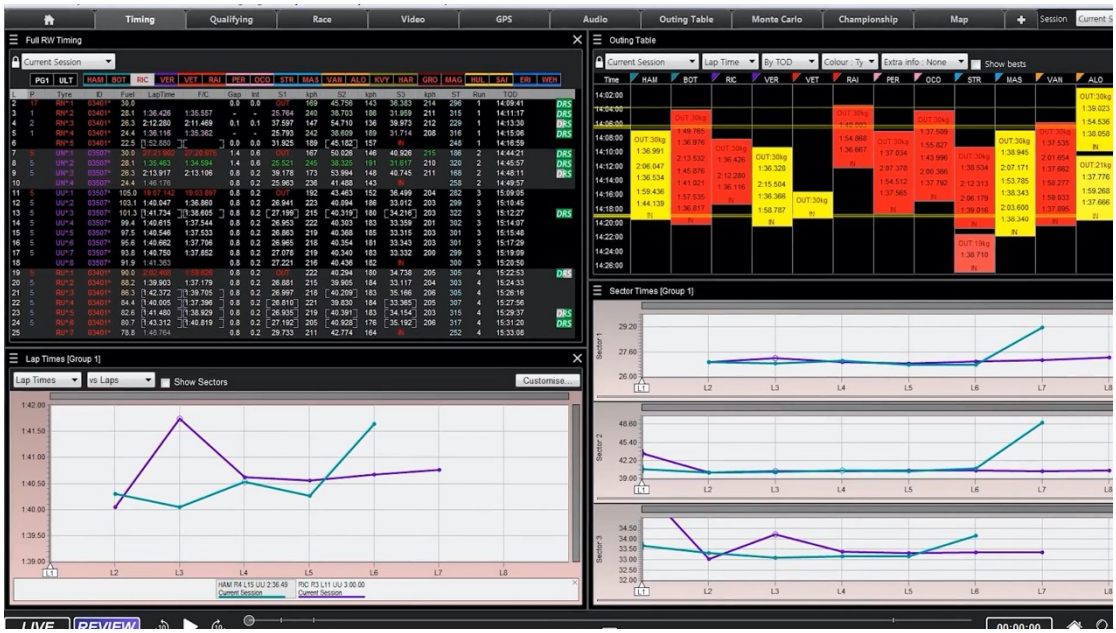


Fig 2.3 SBG Race Watch

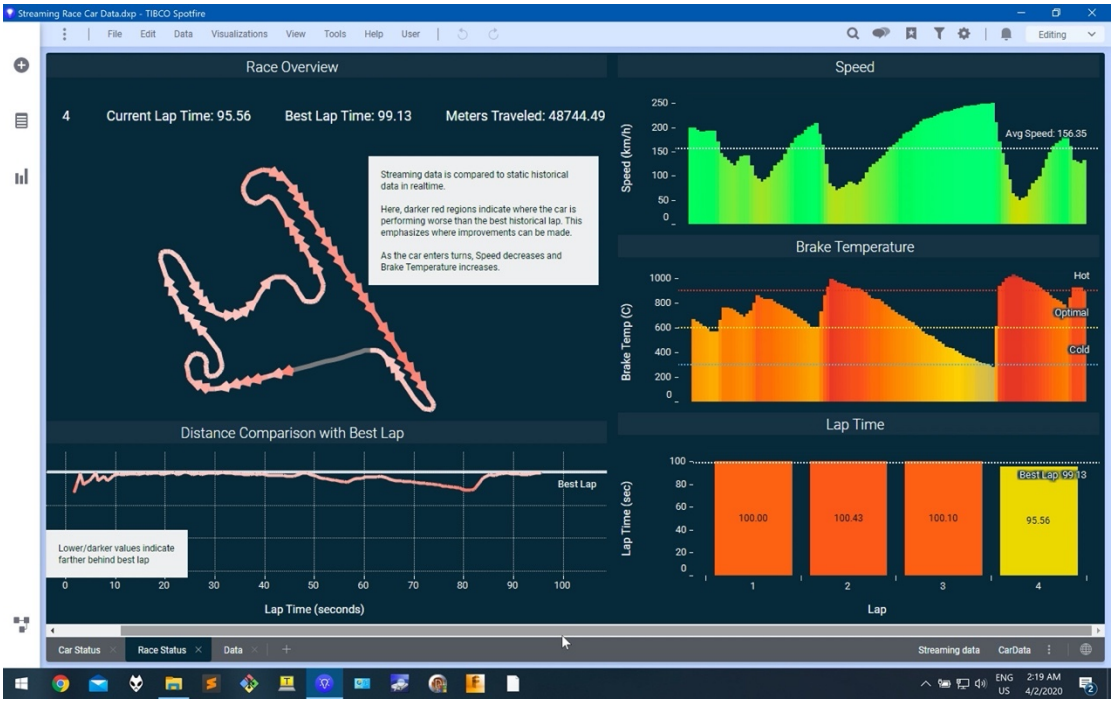


Fig 2.4 TIBCO Spotfire

Chapter 3

Finding the limits of mUAV Performance

3.1 Objective

Before mUAV hardware and software optimisation can begin, it is necessary to determine the flight and powertrain characteristics of the mUAV to determine areas to target during optimisation. The key constraints of high speed mUAV performance are aerodynamic drag and thrust produced to create lift and forward acceleration.

The general equation describing the constraints is:

Thrust * cos(θ) = Lift = - (Weight of mUAV)

Thrust * sin(θ) = Forward Acceleration Force = - (Drag Force)

Drag force and required thrust for forward acceleration increases exponentially with fluid velocity, as described by the drag equation[34]:

$D = Cd * 0.5*(\rho * v^2) * A$

D: Drag force

Cd: coefficient of drag

ρ: fluid (air) density

v: fluid (air) velocity

Reducing to:

$D = k\ v^2$

$Thrust * \sin(\theta) = k\ v^2$

3.2 Field Testing

Before development commenced on a new mUAV prototype, multiple test flights were done on an existing high performance mUAV. The flight plan involved high speed runs in opposing directions of 250-300m.

The metric of performance was calculated as follows:

- In the GPS data, create minimum sized sliding windows of coordinates that form a distance of at least 100m
- Determine average speed and direction of travel in each 100m window
- Select the fastest average speed over 100m distance of both directions and average

This provides the average top speed over 100m for the mUAV, hereby known as average top speed.

The test platform used was a high performance mUAV of an older design, capable of reaching 255km/h average top speed and peak top speed of 298km/h.

Fig 3.1 Test mUAV

The tests covered the following aspects:

Impact of various levels of aerodynamic seal:

Due to the modular nature of the aerodynamic bodywork (allowing for quick replacements and easy access to the LiPo battery), there are gaps and seams in the bodywork that create drag by causing flow separation. By testing similar bodywork with varying levels of aerodynamic seal, the impact of such gaps can be observed.

Fig 3.2, 3.3

Left: bodywork sealing electronics

Right: completely sealed side panels in bodywork

Durability of the PLA FDM manufactured aerodynamic bodywork

Polylactic Acid (PLA) plastic is a commonly used material in Fused Deposition Modelling. Its suitability for use on the new mUAV was also evaluated, by impact testing and durability against heat of the sun

Optimal choice of powertrain components

Different permutations of LiPo battery, motors and propeller specifications were tested:

- Motor stator size (impact on torque and power consumption), motor RPM

a. 2207 stator, 2600kv
b. 2207 stator, 2300kv
c. 2208 stator, 2522kv
d. 2408.5 stator, 2522kv

The stator size is denoted in xxyy notation. xx represents the diameter of the stator in mm, yy represents the height in mm. Generally, the larger the stator, the more torque and power it can produce.

The kv rating denotes the rotations per minute per volt applied to the motor when unloaded. I.E. 2522kv with an 18.5V LiPo would have a theoretical rotation of 46.6K RPM.

- Propellers

The key metrics were efficiency in converting electrical power into thrust, while being able to sustain thrust at the desired speeds. This required a compromise of:

a. Largest propeller diameter usable.

As the motors are operating at extremely high RPMs, the propeller blade tips can easily reach a tangential velocity faster than 330m/s, the speed of sound.

For example, a motor spinning at 56K RPM, based on a 2522kv rating on 22.2V, a propeller blades longer than 4.43” (11.25cm) will break the sound barrier. Such high blade tip speeds will generate supersonic shockwaves that create high drag, massively impacting the efficiency of the propellers.

The calculations are as follows:

KV * voltage = RPM

RPM / 60 * 2 * π = ω (rad/s)

v = D / 2 * ω

D = v * 2 / ω

ω = 56000 / 60 * 2 *pi = 5,864.31 rad/s

D = 330 * 2 / 5,864.31 = 11.25cm = 4.43”

RPM: rotations per minute

ω: angular frequency

D: diameter of propeller

v: linear velocity

In practice, the diameter of the propeller blade may be slightly longer, as the drag caused by the propeller will slow the motor down, resulting in a lower maximum RPM.

b. Highest propeller pitch (to increase airflow per RPM)
c. Lowest Torque requirement (allows higher RPM with same torque to increase airflow per pitch)
d. Highest speed where thrust is more than lift + drag

- LiPo Configuration
  a. 5S1P (18.5V) and 6S1P (22.2V) cell configuration

     Voltage is a factor of maximum RPM available from a motor. Insufficient voltage would result in the motors not developing peak power to produce thrust. Running excessively high voltage would risk causing huge aerodynamic inefficiencies when the propeller tips break the sound barrier
  b. 1300mAh and 1500mAh capacity

     Capacity is a factor of maximum sustainable current a LiPo can provide)

     Sustained current ~= 80 * capacity in mAh
- Electronic Speed Controller (ESC) durability testing
- Understanding how parameters of the LiPo battery affects sustained top speed performance:
  a. Measure LiPo temperature before and after flight and compared to top speed
  b. Maximum sustainable current draw over 5 seconds (sufficient time for a high speed pass)

c. Maximum sustained current draw time

d. Maximum usable energy

e. Burst current draw and duration of burst (to achieve a very high speed pass, alongside other slower but relatively high speed passes during flight)

The test flights were performed at the big field at Old Holland Road. Flights were performed in accordance with Civil Aviation Authority of Singapore (CAAS) regulations and safety guidelines for Unmanned Aircraft (UA). In total, about 50 test flights were performed.[35][36]

3.3 Test Methodology

Test flights were conducted in a team, with the following roles:

- UAV pilot flying the mUAV through First Person View (FPV system)
- Navigator (Line of sight/FPV)
- Spotter (Line of sight)

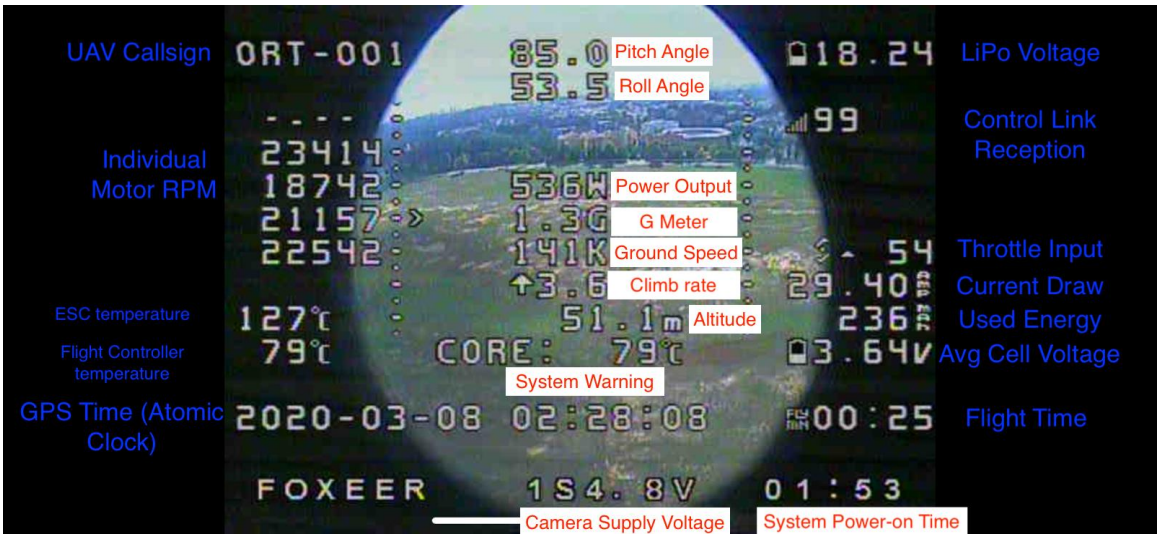


Fig 3.4 FPV view with real-time data displayed on on-screen display (OSD)

The pilot and navigator watch a real-time analog video feed from the mUAV during flight.

The UAV pilot's role is to focus solely on the control the mUAV and landing/take-off. Due to the agility and speed required of the mUAV during operation, pilot assist systems such as self-levelling and automatic position hold systems are not in use. The flight mode used requires the pilot control the mUAV by manually controlling the throttle and rate of rotation on the 3 flight axis (pitch, roll, yaw). This requires much concentration from the pilot. As such, only limited attention can be paid to real-time data, which are provided in the middle of the on-screen display (OSD)

The navigator plays the key role of observing the crowded real-time telemetry stream, I.E. LiPo voltage, maximum current draw and peak speed during each pass, overheating of electronics, stability of control link and satellite lock etc. He summarises the data and provides Feedback is provided to the pilot of changes in the mUAV operating conditions. This feedback is crucial to prevent the mUAV from being operating beyond its safe limits. The navigator also assists during landing by providing Line of Sight altitude information to the pilot. This is required due to limited visibility of the ground when the mUAV transitions to a hover position, caused by the limited FPV camera Field of View and high camera angle required for high speed flight.

The spotter watches the ground where the mUAV is flying and identifies people or objects at the flight area and warns the pilot to avoid them for safety reasons.

For each test flight, following procedures were conducted:

- The follow data were recorded in a logbook:
    a. Weather conditions and ground temperature
    b. mUAV configuration
    c. LiPo temperature right before flight
    d. LiPo temperature right after flight

- Flight data from mUAV's PID controller (sampled at 1KHz) and GPS receiver (sampled at ~5Hz) were logged to a SD card Blackbox onboard the mUAV
- FPV video feed with telemetry on OSD was recorded to a SD card using a DVR on the pilot's FPV goggles
- A pre-flight inspection to ensure the components are not damaged and properly secured. With the propellers off, the PID controller and IMU is tested for stability.
- The same LiPo will be used for all flights
- A warmup flight is conducted to perform control systems check and warm the LiPo up to ~50°C. Temperature measurements are done on LiPo surface temperature with an infrared thermometer.
- The half-depleted LiPo will be recharged back to full capacity at 5C (6.5A for 1.3Ah LiPo). After recharging, the LiPo would be around 40°C
- With the warmed up LiPo, high speed passes of ~250m in opposing directions will be performed. The mUAV will reach sustained top speeds (~250km/h) during each pass. During this pass, the LiPo will heat up significantly and overheat, with voltage sag due to increased internal resistance, occurring at the end of the pass.
- At the end of each pass, the mUAV will slow down to ~160km/h and circle around for the opposite direction pass. This allows power draw to drop, for the LiPo to cool off and internal resistance to drop.
- Around 6 passes are performed during each flight, depending on power consumption
- The typical flight path is shown below.

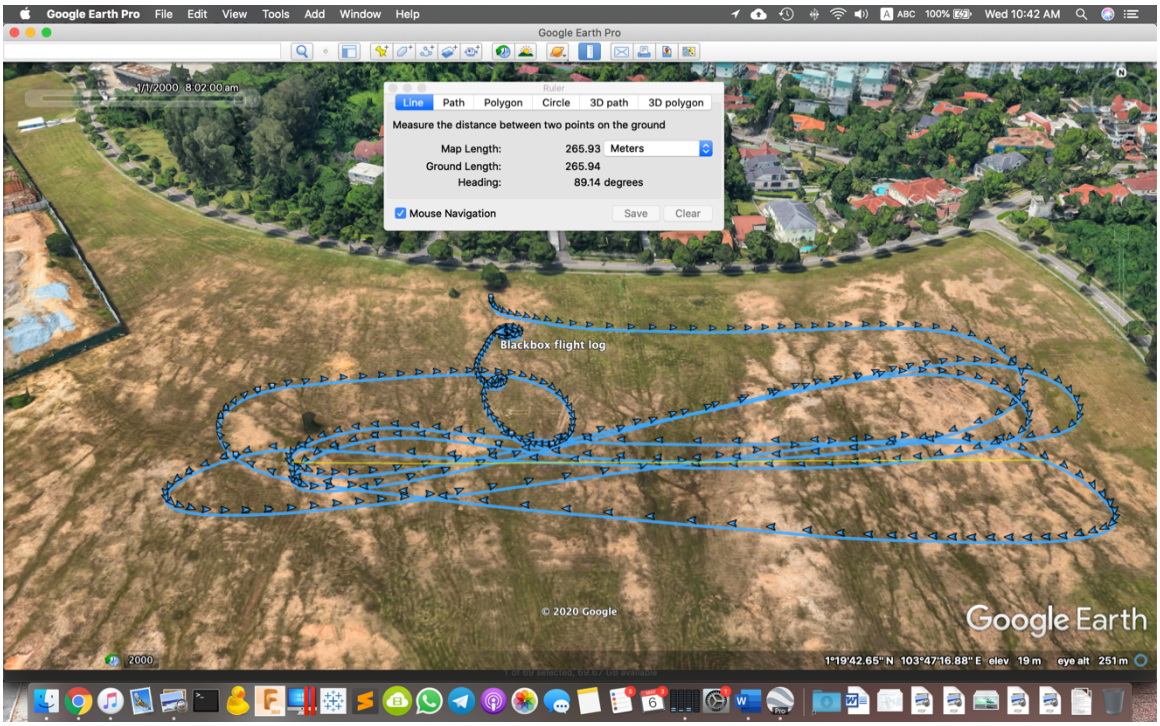


Fig 3.5 Test flight path

Fig 3.6 Zhong Bingchen (Navigator/Spotter) logging flight conditions

3.4 Results

Heat Sensitivity of PLA

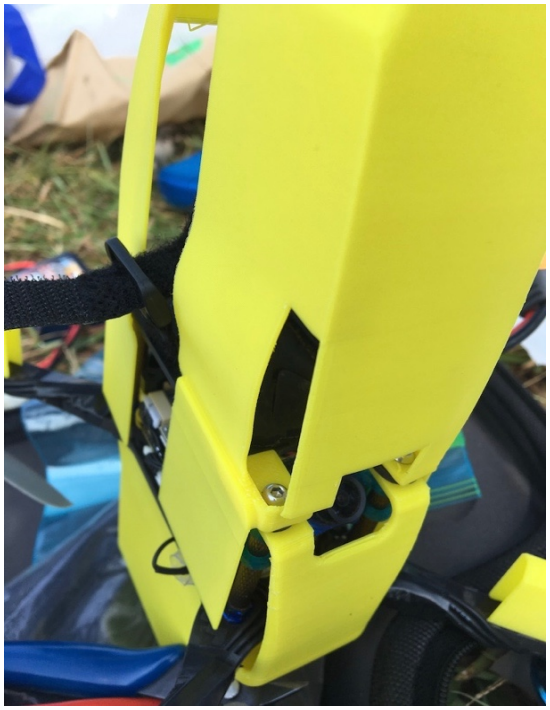


Fig 3.7 PLA warping on mUAV

It was discovered that PLA would warp when the LiPo battery became sufficiently hot at ~60°C. A more heat-resistant material had to be used

Optimal Powertrain Configuration

Comparing the various powertrain configurations, the follow configuration provided the highest sustained top speeds:

- 2408.5 2522kv motors
- 5.2x6.0” propellers
- 55A ESC
- 5S (18.5V) 1.5Ah LiPo

It was determined the LiPo was able to produce the most power at temperatures of 65°C, while the maximum temperature it could reach before thermal failure occurs was 70°C.

With the finalised powertrain, high speed passes of increasing maximum throttle input till LiPo encountered thermal failure. To capture maximum safe throttle value (for throttle limiter). Key metrics were current draw (Not exceeding 120A), voltage sag (not dropping below 16V), top UAV speed, maximum motor RPM, abnormal noises from the motors due to excessive prop slip or blade flattening from excessive thrust. The maximum throttle was locked to 94%, at which the mUAV would draw 125A at 18.5V (2.3kW), with maximum motor RPM of 33K.

Flight data logs were exported from the .BBL format generated by the Betaflight flight controller into .CSV files, using the Betaflight Blackbox Tools utility.

The data was stored for Exploratory Data Analysis using Excel and q/kdb+ for use in the development of UAV Rolling Launch Control (URLC) machine learning application.

time (us)	GPS_numSat	GPS_coord[0]	GPS_coord[1]	GPS_altitude	GPS_speed (m/s)	GPS_ground_course
61486787	7	1.3286602	103.7882237	72	33.08	161.8
61686859	7	1.3286099	103.7882198	73	29.84	177.1
61886789	7	1.3285663	103.7882044	75	26.54	191
62086848	7	1.3285292	103.7881771	77	25.02	207.2
62377424	7	1.3284979	103.7881388	78	26.31	224
62487325	7	1.3284724	103.788091	80	28.62	236.3
62687182	7	1.3284529	103.7880331	81	31.41	246.9
62887633	7	1.328439	103.7879628	82	36.23	255.2
63088186	7	1.3284289	103.7878786	82	43.08	261.4
63378179	7	1.3284215	103.7877817	83	50.67	264.6
63488307	7	1.3284162	103.7876747	83	57.05	266.5
63688453	7	1.3284114	103.78756	84	61.7	267.6
63888238	7	1.3284066	103.78744	84	65.64	267.6
64088581	7	1.3284014	103.7873164	85	67.76	267.7
64378685	7	1.3283947	103.7871906	85	69.55	267.4
64488910	7	1.3283866	103.7870637	86	70.23	266.6
64688824	7	1.328378	103.7869357	87	70.98	266.1
64889491	7	1.3283688	103.7868077	88	71.35	266.2
65089401	7	1.3283595	103.7866793	89	71.15	265.6
65379775	7	1.3283514	103.7865506	89	71.58	266
65489448	7	1.3283452	103.7864227	88	71.35	266.7
65689525	7	1.3283402	103.7862949	86	70.91	267.9
65899675	7	1.3283347	103.7861673	84	70.98	268
66100141	7	1.3283307	103.7860449	81	71.01	267.5
66379871	7	1.3283297	103.7859348	79	65.01	268.6
66490004	7	1.3283329	103.785836	76	57.24	270.3
66700293	7	1.3283351	103.7857453	74	52.86	273.5
66890520	7	1.3283229	103.785659	71	48.02	270.1
67100826	7	1.3282961	103.7855812	68	48.65	256.5
67280920	5	1.3282961	103.7855812	68	43.41	246

Fig 3.8 Blackbox log segment of GPS data with 7 fields

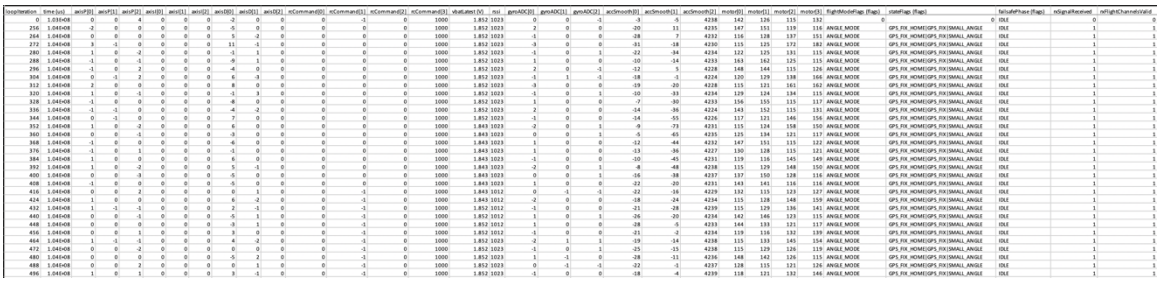


Fig 3.9 Blackbox log segment of PID controller data with 32 fields

The flight data was analysed using the GPS Speed Run Analyser published by Ryan Lademann[37]

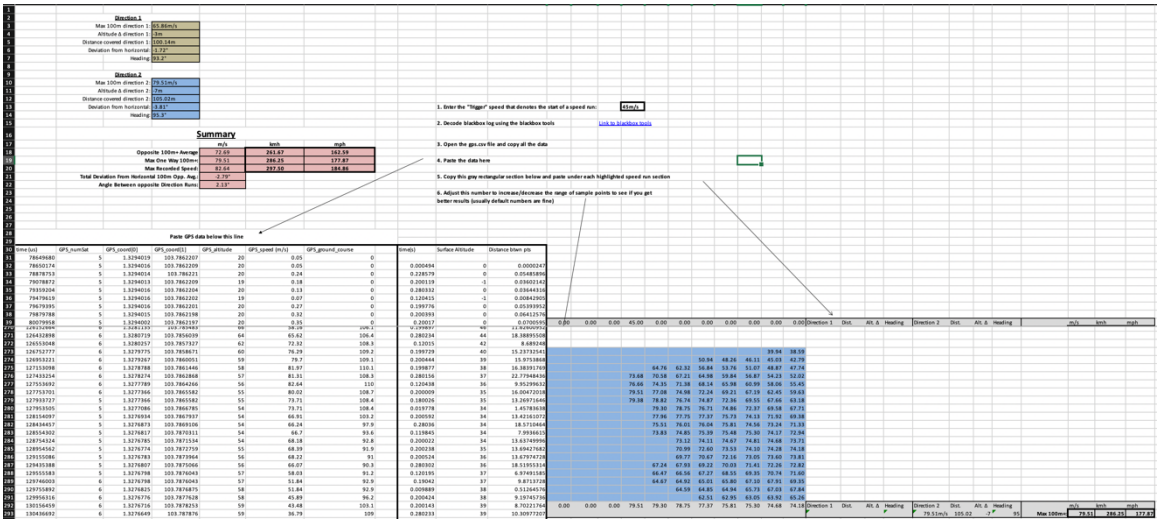


Fig 3.10 GPS Speed Run Analyser

The results were further summarised and tagged to flight logs

A	B	C	D	E	F	G	H	I	J	K
Flight #	DVR Name	LogFile	Opp 100m Average (World Record Metric)	Peak Speed/kph	Direction 1 100m Speed/kph	Direction 1 Heading	Direction 1 Alt Delta	Direction 2 100m Speed/kph	Direction 2 Heading	Direction 2 Alt
1		49	245.85	256.25	250.5395901	249.2°	0m	236.8978802	89.4°	-3m
2		51	227.66	231.66	221.6029675	270.9°	3m	225.5079658	84.8°	6m
3		53	233.64	244.8	240.2432065	274.5°	4m	235.1825344	91.9°	7m
4		56	245.81	266.04	263.8134215	117.88m	-3m	226.6891274	83.2°	-1m
5		58	255.41	262.8	241.2212521	263.2°	-1m	250.5941058	73.3°	-1m
6		59	250.33	258.23	256.1537089	251.0°	-6m	243.2229027	86.9°	0m
7		60	237.55	247	230.2019663	250.6°	9m	238.0559908	86.0°	-2m
8		62	230.32	238.43	227.3160934	255.8°	3m	230.6567277	84.5°	-16m
			Opposite 100m average from all flights	257.204kph	263.813kph			250.594kph		
			Wind Speed	6.610kph						

Flight #	DVR Name	LogFile	Opp 100m Average (World Record Metric)	Peak Speed/kph	Direction 1 100m Speed/kph	Direction 2 100m Speed/kph	Direction 1 Alt Delta	Direction 2 Alt Delta
1	PICT0004	39	216.38	223.45	218.46	214.3000281	2m	-4m
2	PICT0005	40	216.38	223.45				
3	PICT0006	42	209.22	239.72	180.2936883	238.1528376	0m	-14m
4	PICT0007	43	232.06	245.92	220.257248	243.8558569	2m	0m
5	PICT0010	44	241.22	254.81	229.6247726	252.81	-6m	1m
6	PICT0011	45		248.58		246.99		

Fig 3.11 Samples of summarised flight data collected

Chapter 4

Optimisation of mUAV Aircraft Design

4.1 Objective

The objective of the mUAV aircraft design process is to create a fuselage that has low losses from aerodynamic drag and lift required to sustain level flight at high speeds.

The general design constraints of the mUAV are subjected to the following constraints:

1) Bodywork of mUAV must be optimised for low aerodynamic drag
2) mUAV must be optimised for low mass
3) LiPo batteries have limited power output and duration of peak power output
4) LiPo batteries have a small operating temperature window for peak performance
5) Electric motors in use have a limited RPM range where they can produce efficient peak power and sufficient torque.
6) Propellers are limited to below 6 inches in length for safety reasons
7) Frame must be sturdy to reduce transmission of vibration to IMU
8) Frame must be strong to withstand heavy impacts in the event of high speed crash (80G of instantaneous G force)
9) Bodywork must be easily detachable from frame so LiPo can be swapped easily (safety reasons)
10) Placement of camera system must provide sufficient field of view

Working around the design constraints, the areas for mechanical/aerodynamic optimisation were:

1) Packaging the electronics and LiPo to a minimal volume for central Centre of Gravity (COG) and reduced aerodynamic profile for drag reduction and flight stability
2) Sturdy chassis that provides sufficient protection for electronics and flammable LiPo in the event of strong impact while having minimal material usage
3) Mass reduction to reduce required lift in level flight
4) Having minimal frontal aerodynamic surface for drag reduction
5) Maintaining a low drag coefficient
6) Sufficient cooling for electronics that can get hot during operation (>100 Celsius)
7) LiPo must be easily accessible for safety and performance reasons
    a. Quick installation to minimise temperature drop before flight begins
    b. Quick removal in case of combustion
8) Design has to feasible for manufacture without complex tools or processes
9) Maintain modularity for maintenance and quick repairs
    a. Many electronic modules are present. In case of failure, each part must be easily replaceable without going through a complex procedure of dismantling a lot of frame components and electronics.
    b. Access to ESC for periodical replacement before failure
    c. Blackbox files are stored on SD card on flight controller. Flight controller must be easily assessible for easy retrieval

4.2 Design Process

The mUAV hardware was designed using Computer Aided Design (CAD), using Autodesk Fusion 360 and Autodesk Flow Design. In CAD, the geometry and physical

properties of all parts are captured with high precision and displayed in a 3D view. Mechanical simulations can be carried out on the full assembly as well, using Finite Element Analysis (FEA).

The mUAV uses the arms of the ARX-V2 mUAV frame from quadstardrones.com. Due to long lead-times and high costs from testing and manufacturing a similar low-drag arm design from scratch, an existing design was chosen. This specific design was chosen due to the vertically aligned arms that have minimal frontal area in the direction of airflow, resulting in low drag.

To begin the process of CAD design, the dimensions of the ARX-V2 frame and mUAV electronics were accurately captured and modelled in the CAD workspace. Parts were measured with up to 0.10µm precision using vernier calipers to reduce rounding errors that can cascade when aligning other parts to the base models.

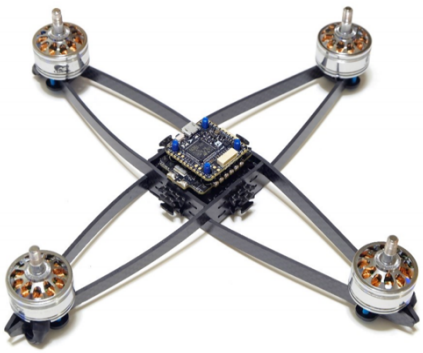


Fig 4.1 ARX-V2 frame

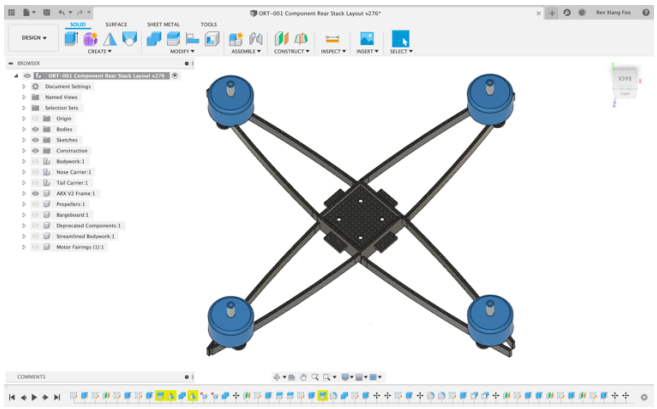


Fig 4.2 ARX-V2 frame reverse engineered in CAD

The electronics includes the following components:

- Flight Controller
- Electronic Speed Controller
- Video Transmitter
- Control Link Receiver

- GPS Receiver
- LiPo Battery
- First Person View Camera

The existing models would serve as the constraints (mounting positions of the arms, packaging of electronics) in the design of the frame (which serves as the electronics carrier and crash structure) and bodywork.

Initially, a standard mUAV electronics layout was considered, with the electronics and LiPo battery placed on the front of the mUAV.

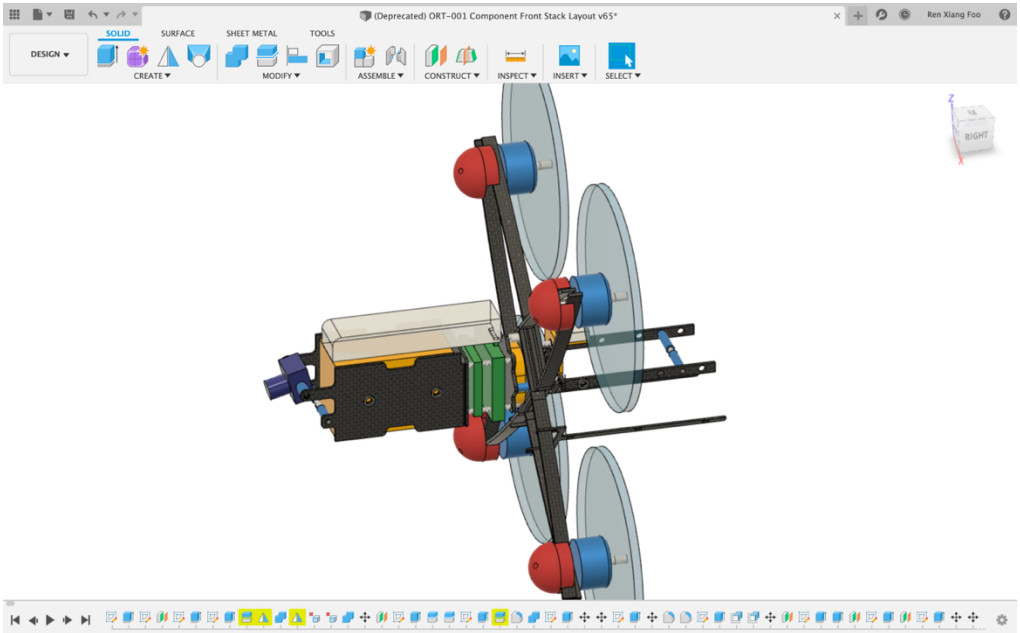


Fig 4.3 mUAV layout with electronics in front

This layout proved to be space inefficient, as low drag aerodynamic geometry generally requires a taller side profile that tapers towards the rear (to prevent flow detachment).

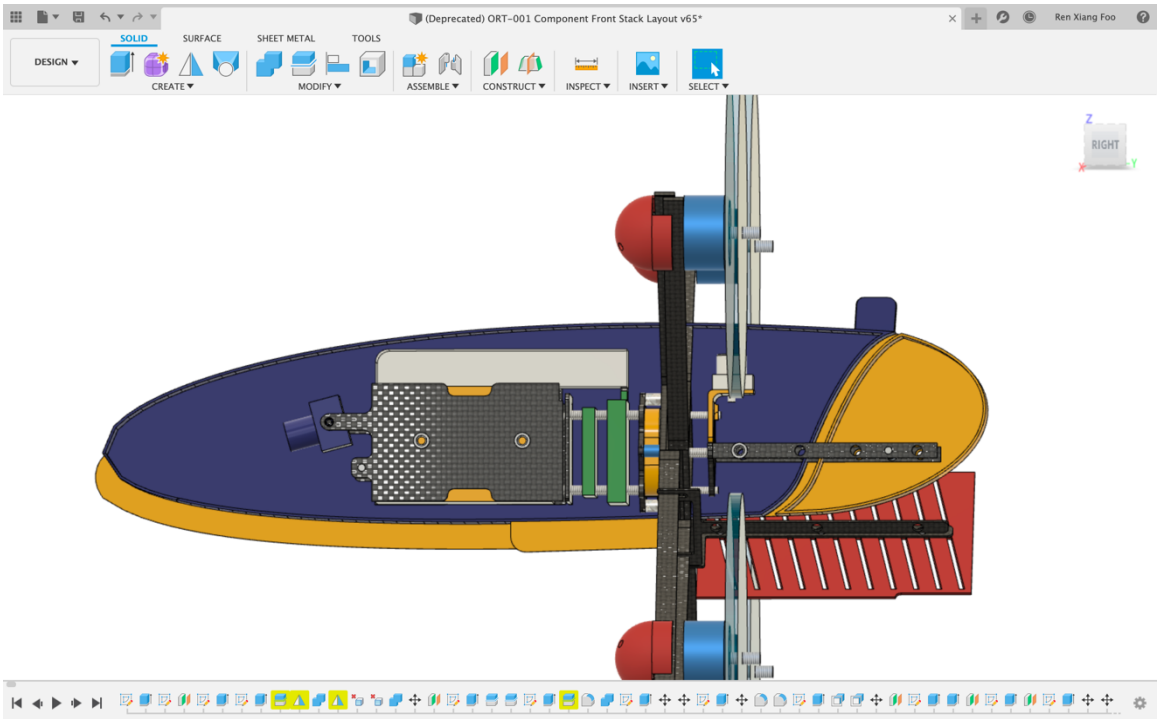


Fig 4.4 Side profile view of mUAV

The solution found was to place LiPo battery, which was tall and wide, at the front, while shifting the rest of the electronics, which were tall but narrow, into the rear and close to the center of the frame. This allowed the bodywork to be narrower with a smaller frontal area in the direction of airflow.

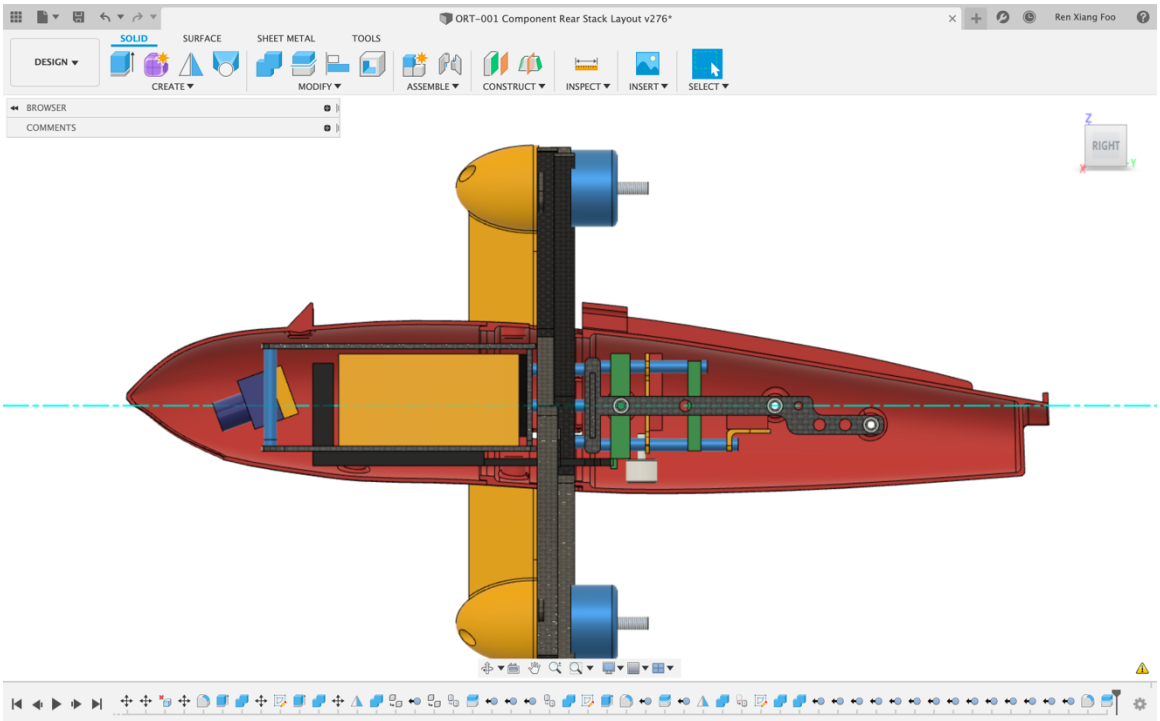


Fig 4.5 Side profile view of mUAV

With the electronics laid out, the rest of the frame were designed. The frame acts as a support structure for mounting the electronics and drag reduction bodywork and as a crash structure to protect the LiPo and electronics in the event of a high speed crash. The frame was made out of carbon fiber plates, which are strong and light. It is also a relatively accessible material that can be easily machined from CAD data using Computer Numerical Control (CNC) machines.

To design the plates, the frame mounting points and volume required to cover the electronics were determined. Next, the plates were oriented to create the volume with the

required cut-outs.

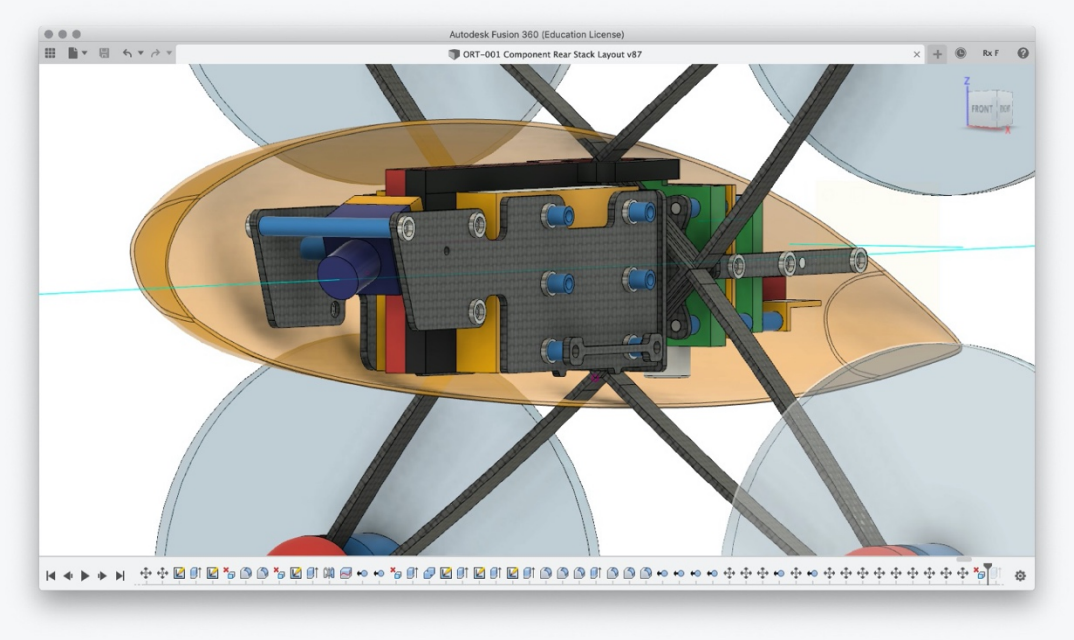


Fig 4.6 Side profile view of mUAV

At this point of the design, the carbon plates are generally a continuous volume. Some of the material used are in fact non-structural, due to the high strength of the material. To reduce weight, excess carbon fiber material has to be removed.

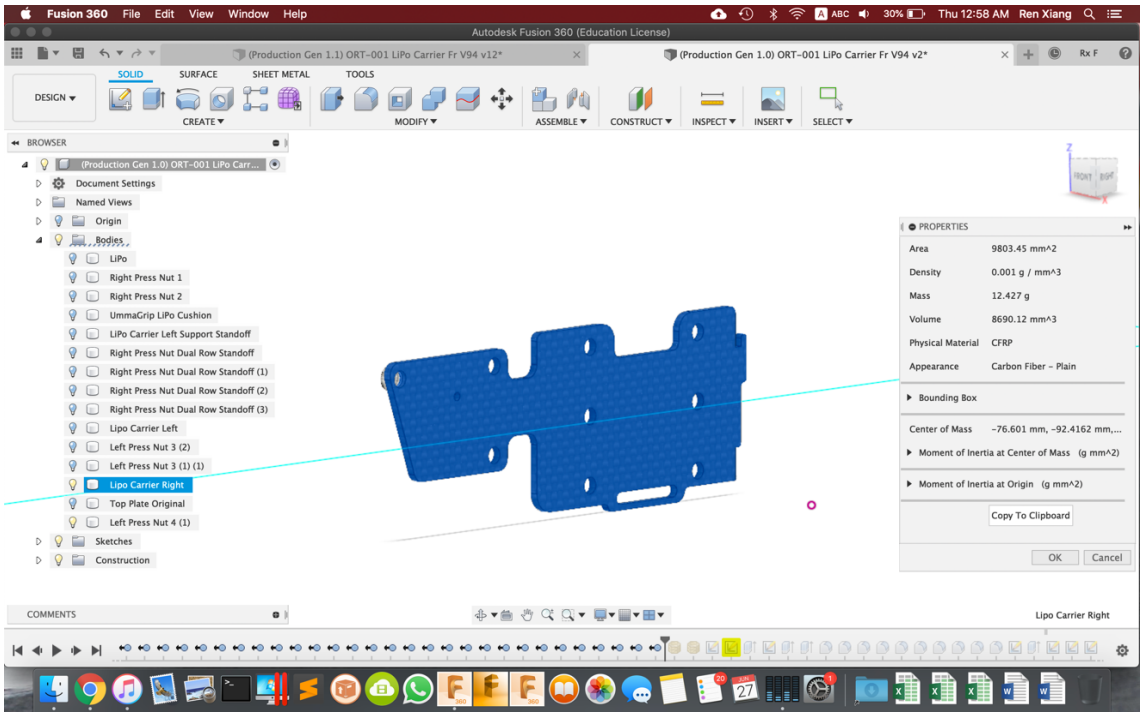


Fig 4.7 Side plate of LiPo Carrier

To reduce the additional material, Finite Element Analysis (FEA) was used to identify load-bearing volumes of the plates that are providing key structural support. FEA simulation was performed by defining loads and constraints that the frame will experience when in flight and during a high impact crash. To avoid unexpected behaviour during analysis, constraints were set to include areas of the plate which may not be altered (e.g. mounting holes or crucial geometric features). FEA was performed using Fusion 360's FEA functionality. During analysis, relevant components were imported directly from the CAD drawing and specified for analysis.

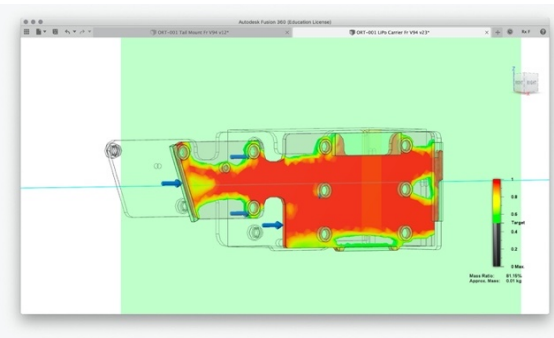


Fig 4.8 Structural sections of side plate

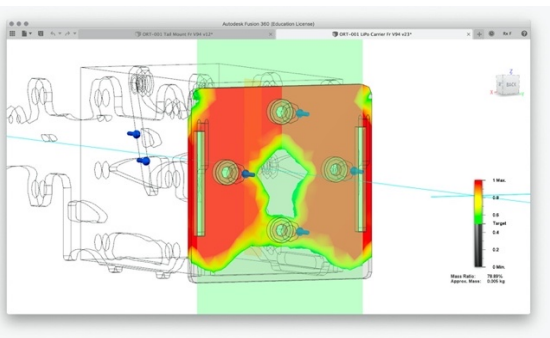
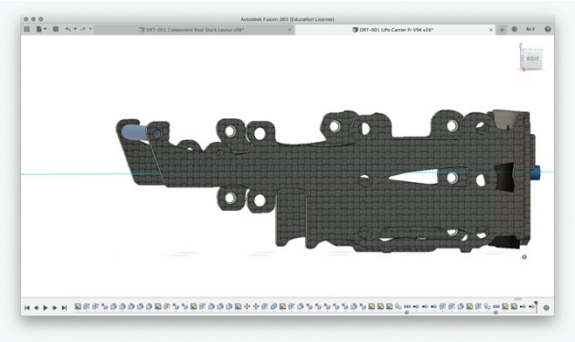


Fig 4.9 Structural sections of rear plate

The results of the simulation colours in the volume that are structural, while leaving the excess materials transparent. The simulation results are saved as a mesh and ported back into the CAD drawing, on which it will serve as a reference for removing excess material.



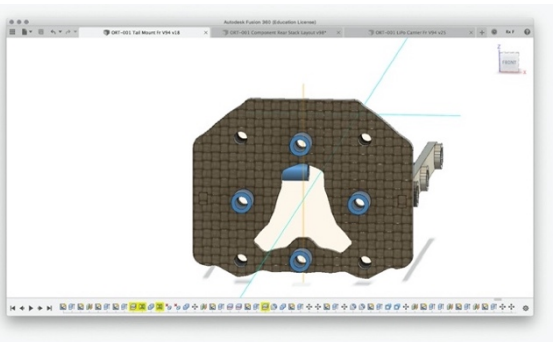


Fig 4.10 Optimised side plate

Fig 4.11 Optimised rear plate

For this mUAV design, ~30% frame weight is saved by removing excess material using FEA.

An alternate version of the frame's tail was also designed based on the aeroelastic concept. The tail piece was designed to be sufficiently strong to withstand impacts with fracturing, but is semi-flexible. At high speeds, the drag force increases and develops sufficient torque to flex the tail section upwards. This reduces the frontal surface area exposed to incoming airflow, reducing drag. The force required and displacement during tail flex was determined by using FEA.

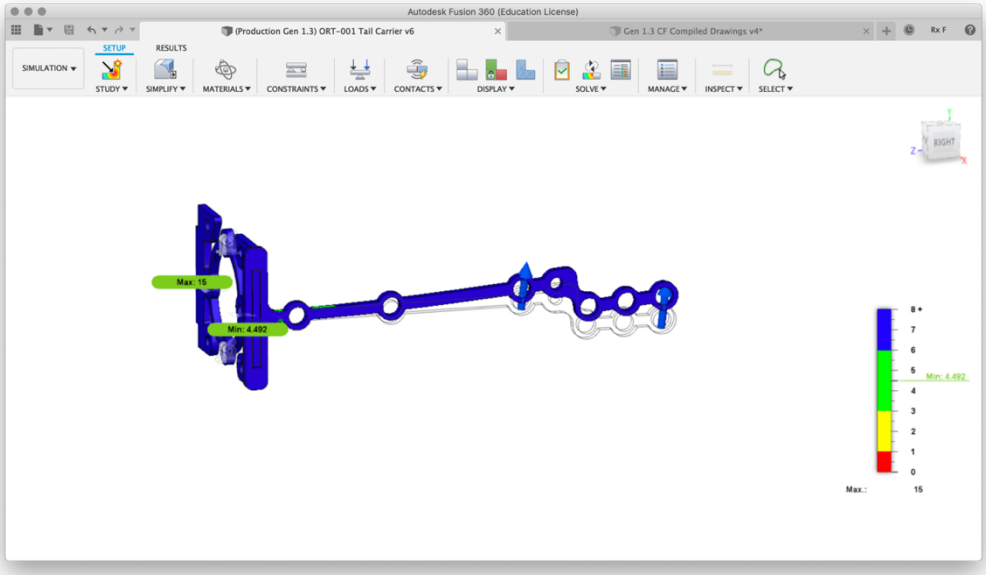


Fig 4.12 Aeroelastic tail

The final carbon fiber designs used in production are shown:

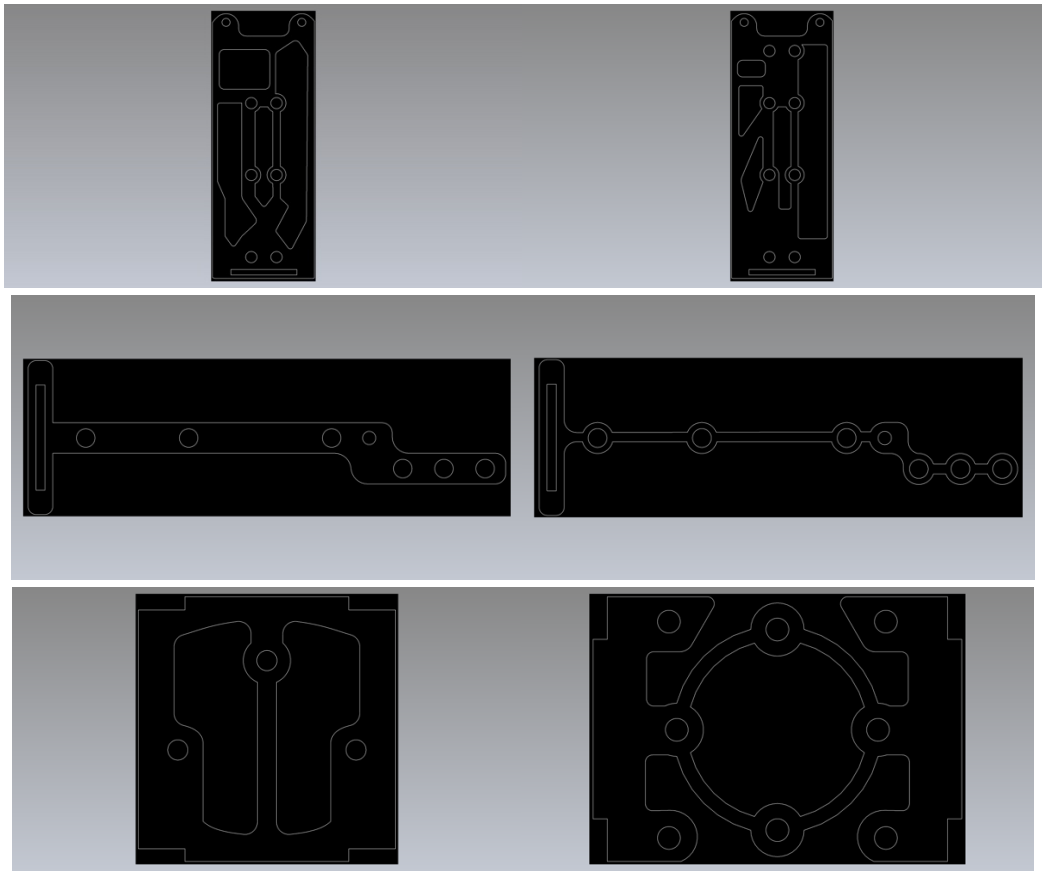


Fig 4.13 CNC DXF Data for carbon plate manufacturing

The drag reducing aerodynamic bodywork was designed around the frame structure. It fully encloses the frame in order to manage the incoming airflow at high speeds. The main aerodynamic design investigated was:

- Streamlined main geometry to reduce drag coefficient
- Large ratio of width:height for the main geometry to some produce lift at high speeds
    - This reduces the angle of attack required to produce lift with thrust. A lower angle of attack presents a smaller frontal surface area, reducing drag.
- Shortened tail to reduce frontal area at positive Angle of Attack (pitch)
- Vortex generators on bodywork to create vortices upstream that induce high energy airflow with boundary
- Spoilers to stop turbulent flow generated by the arms and catch flow reversal observed at the tail
- Motor fairings to reduce pressure drag on motors exposed fast airflow
- Turning vanes on motor fairings to push propeller wash outboard from the mUAV
    - Heavily turbulent airflows are produced by the propellers
    - Turbulence that interacts with the boundary layer on the tail of the drone will cause early flow detachment
- An S-duct that at high angle of attack, channels laminar airflow from the belly of the mUAV to inject out of the tail, in order to re-energise the disappearing boundary layer and reduce turbulent wake behind the drone

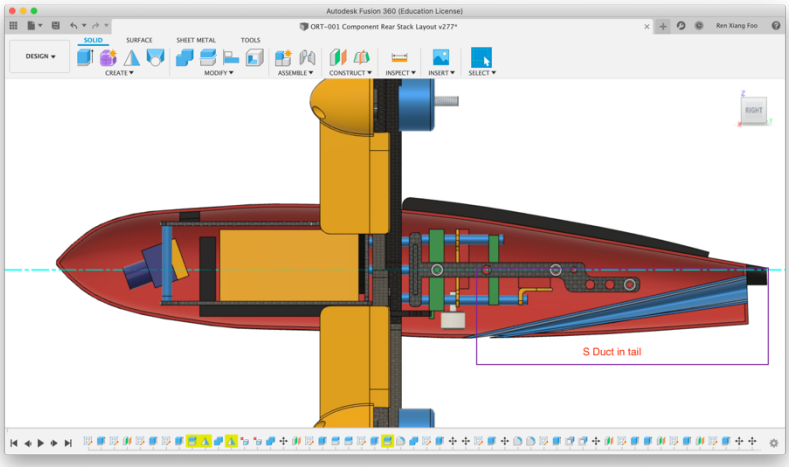


Fig 4.14 S-duct, mUAV side view

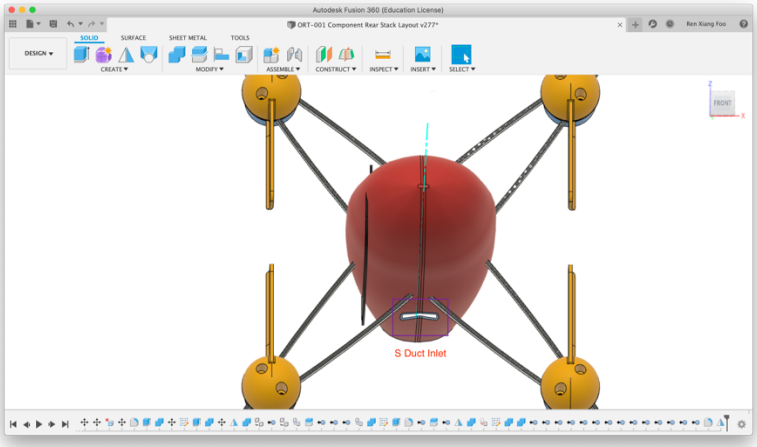


Fig 4.15 S-duct, mUAV front view

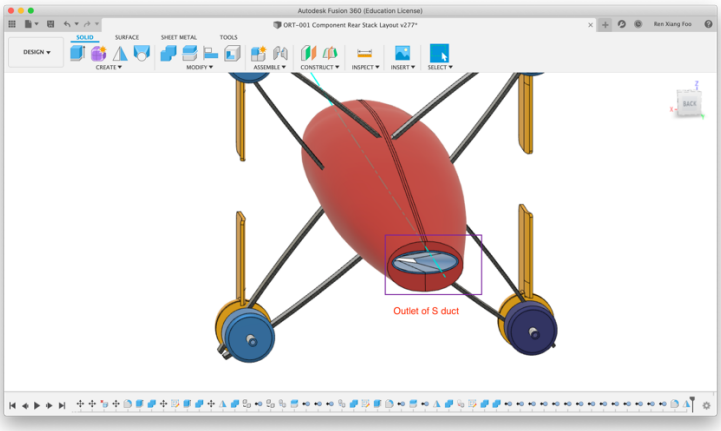


Fig 4.16 S-duct, mUAV rear view

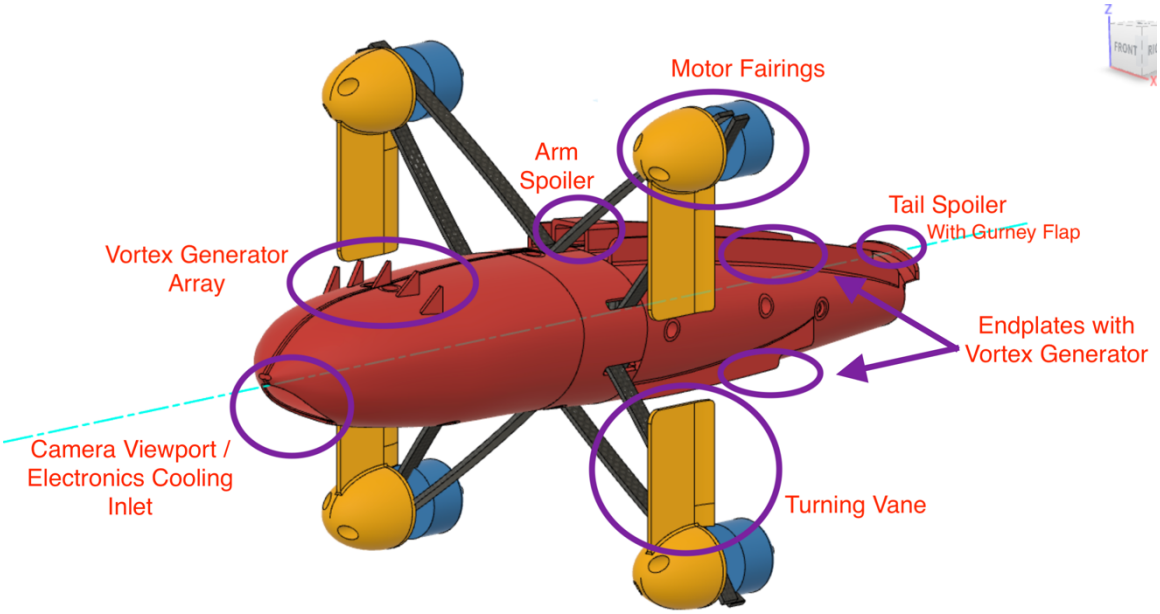


Fig 4.17 Other aerodynamic devices

The bodywork was drawn using the following process:

- A side cross section was drawn. This was based on a modified NACA 2412 air foil.
- Using the side section as a guide, additional cross sections that encompass the electronics and frame were drawn on the perpendicular frontal plane
- A solid body was then lofted using the cross sections
- The solid body was shelled to 2mm thickness to create the bodywork

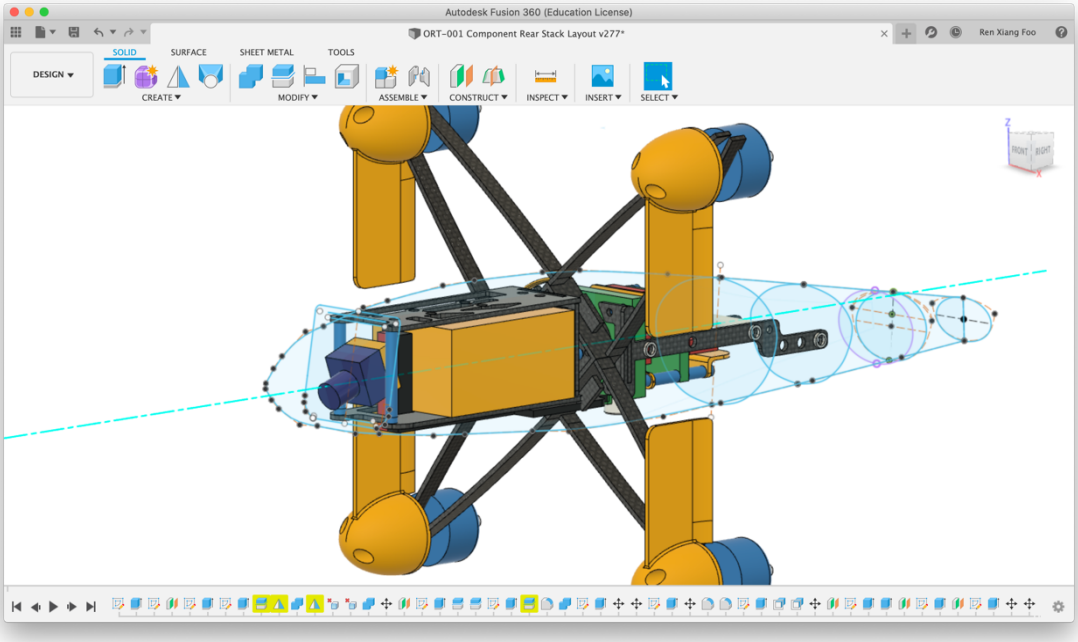


Fig 4.18 Cross sections guides used to loft the bodywork

Computational Fluid Dynamics (CFD) is used to determine quantitatively simulate drag forces acting on the mUAV at high speeds with different geometry. CFD also provides a visualisation of fluid flow vectors, indicating direction of flow and speed of flow. High pressure regions at the front of the mUAV caused by airflow impacting the mUAV, as well as turbulent flow, caused by flow separation of incoming airstream due to surface disruptions are observable in the CFD simulation. Turbulent flow causes a low pressure wake to form behind the mUAV at speed, which forms a suction force that is the major contributor of aerodynamic drag.

Performing CFD simulations require a deep understanding of Fluid Dynamics to tune to a high level of accuracy and precision. An incorrectly tuned CFD model would result in poor correlation between real-world behaviour and CFD simulation results. Typically, CFD simulation tuning requires some real-world testing of a variant of the test aircraft to gather real-world behaviour, that is in turn used to tune the simulation parameters until simulation

results converge with real-world behaviour. From that point, the tuned parameters are used to simulate mostly similar design to investigate differences in behaviour.

Such a process requires access to substantiable knowledge of fluid dynamics, measurement devices and an actual working mUAV of the target design, which are not attainable during the design phase of this project. However, CFD is still useful at a lower precision, when used with a generalised model, such as the one implemented in Autodesk Flow Design. It can be used to determine relative drag levels between designs, while visualising flow separation of geometric features larger than 2mm.

While designing the bodywork, the CAD files of each design iteration was exported and ran in Autodesk Flow Design. The simulations were run with the following setting:

- 10 degrees of positive pitch ($\theta = 10°$)
- Flow velocity of 320km/h ($v = 89m/s$)
- Voxel size: 0.0002m

The simulation was left to iteratively solve until convergence. This was confirmed by ensuring the status indicated changed from ‘transient’ to ‘stabilized’ and looking at the drag coefficient values calculated between consecutive iterations remaining stable. It would typically take 1-3 days for each simulation to reach convergence, due to the computationally intensive nature of CFD simulations. The simulations were ran from a dual core CPU. In total, 33 designs were made and tested.

Comparing the drag levels of the test mUAV and the new mUAV design, the final design saw a drag force reduction of 39.85%, from 27.795N to 16.717N. Drag coefficient was reduced by 50% from 0.62 to 0.31.

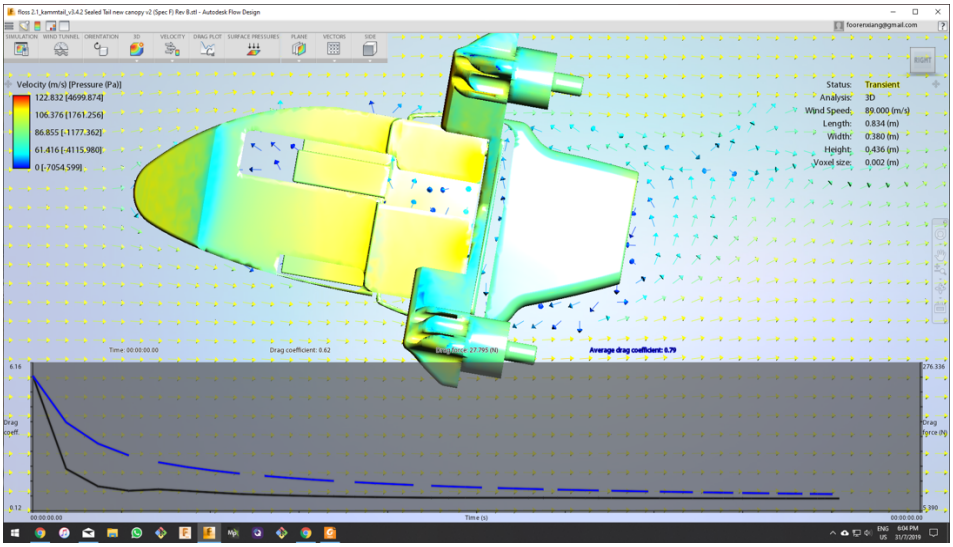


Fig 4.19 CFD results of the mUAV used for initial field testing

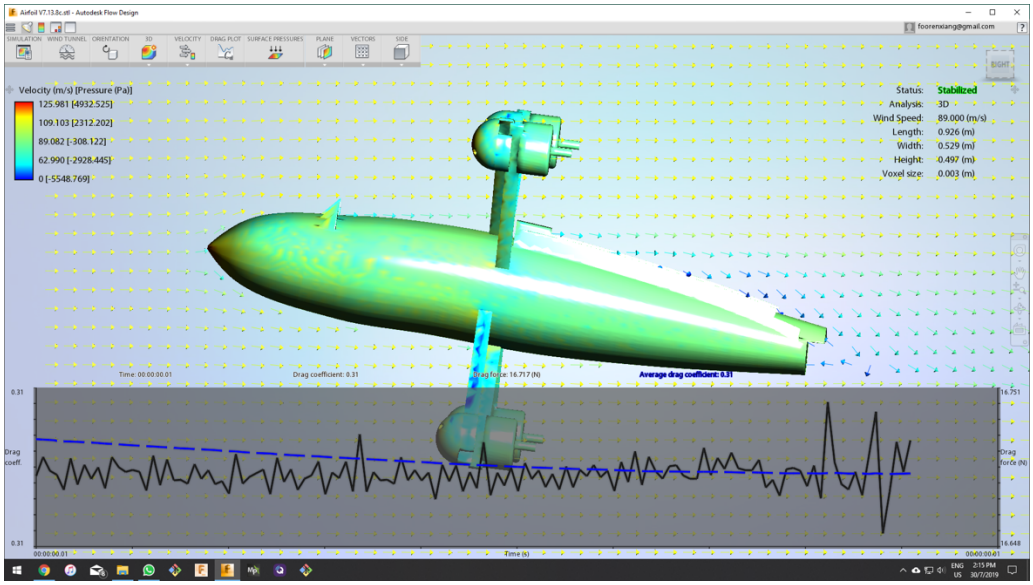


Fig 4.20 CFD results of the new mUAV

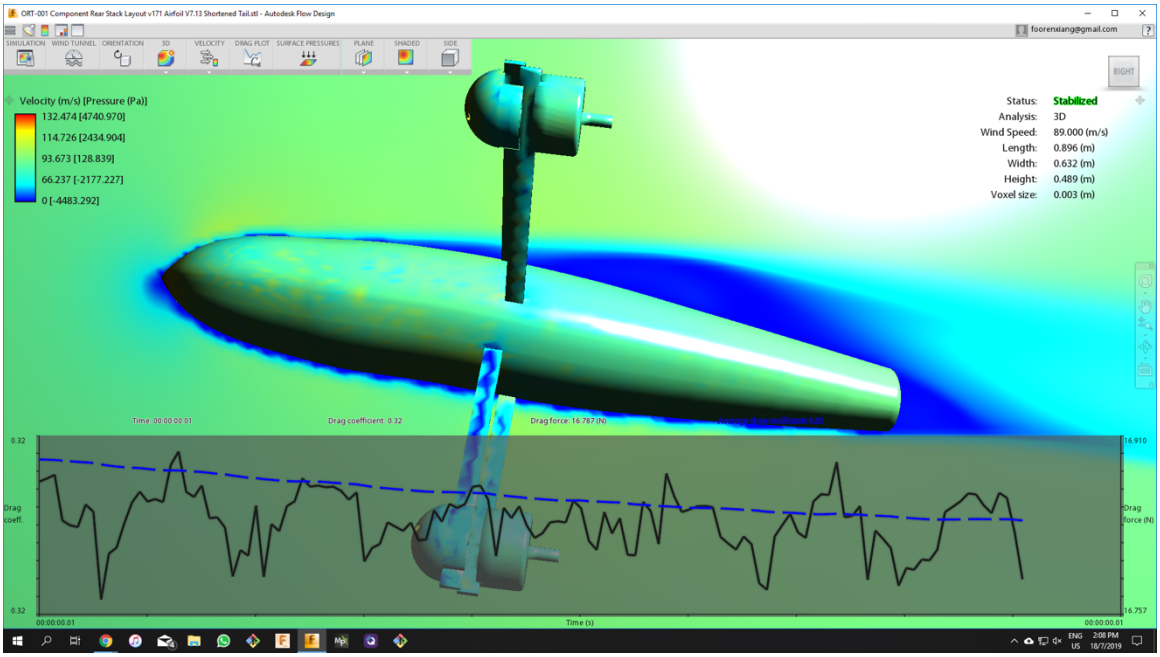


Fig 4.21 Clean main geometry

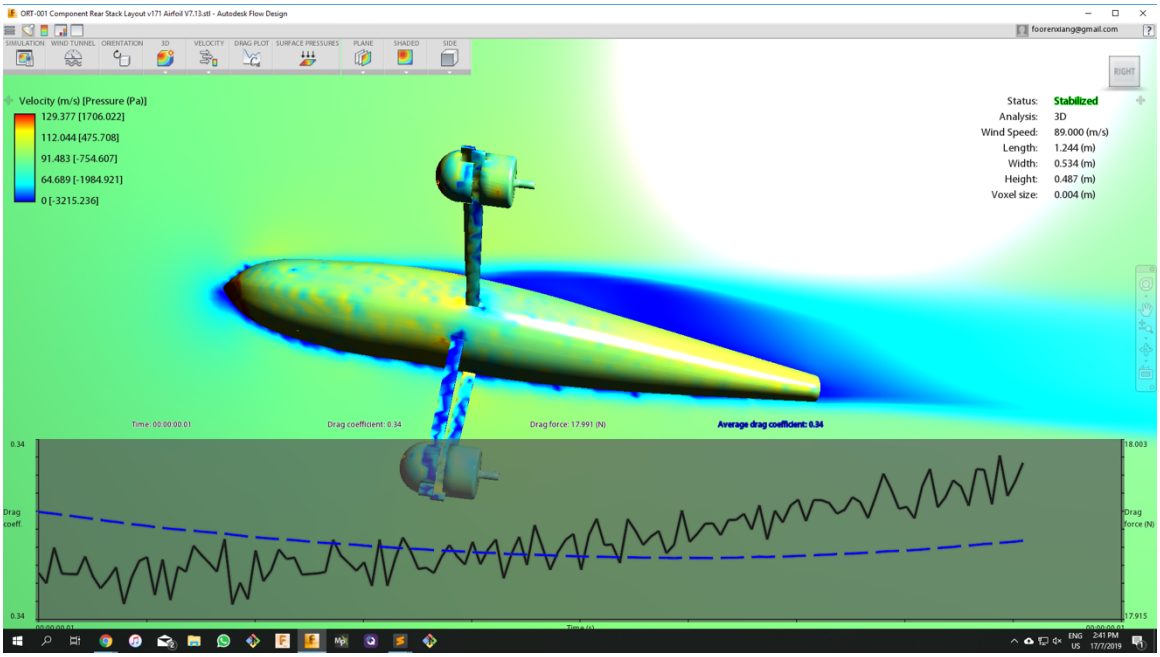


Fig 4.22 Tail with aggressive taper

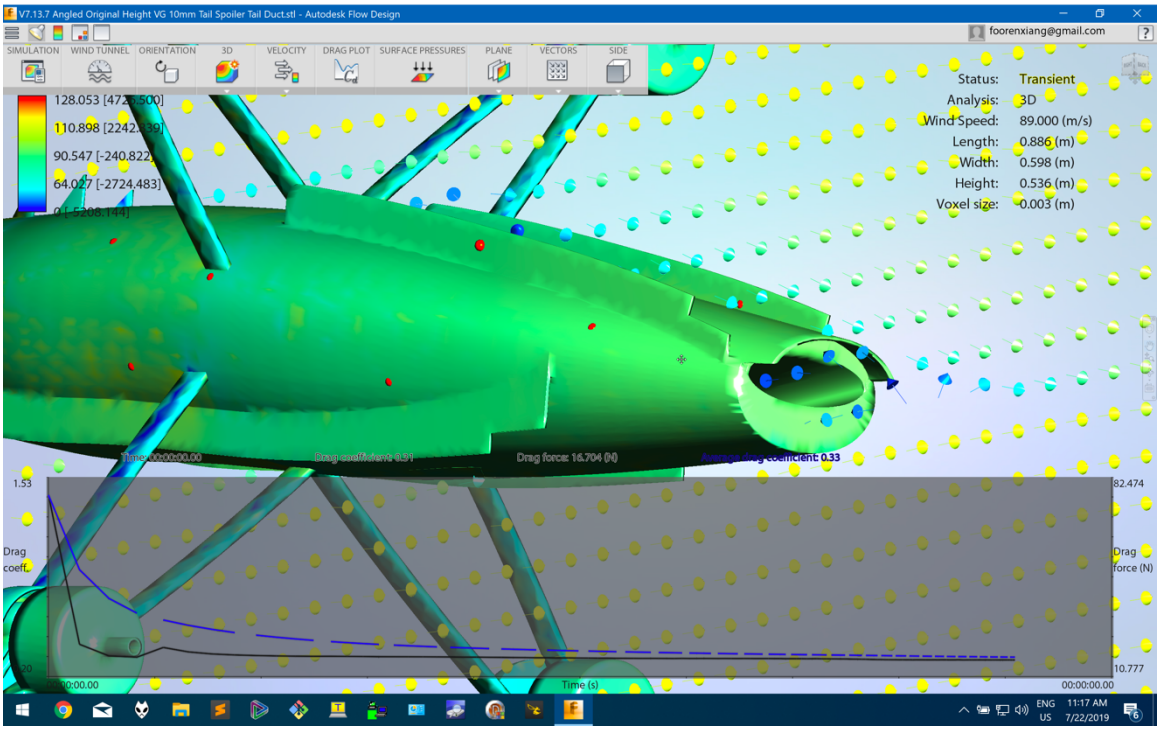


Fig 4.23 S duct

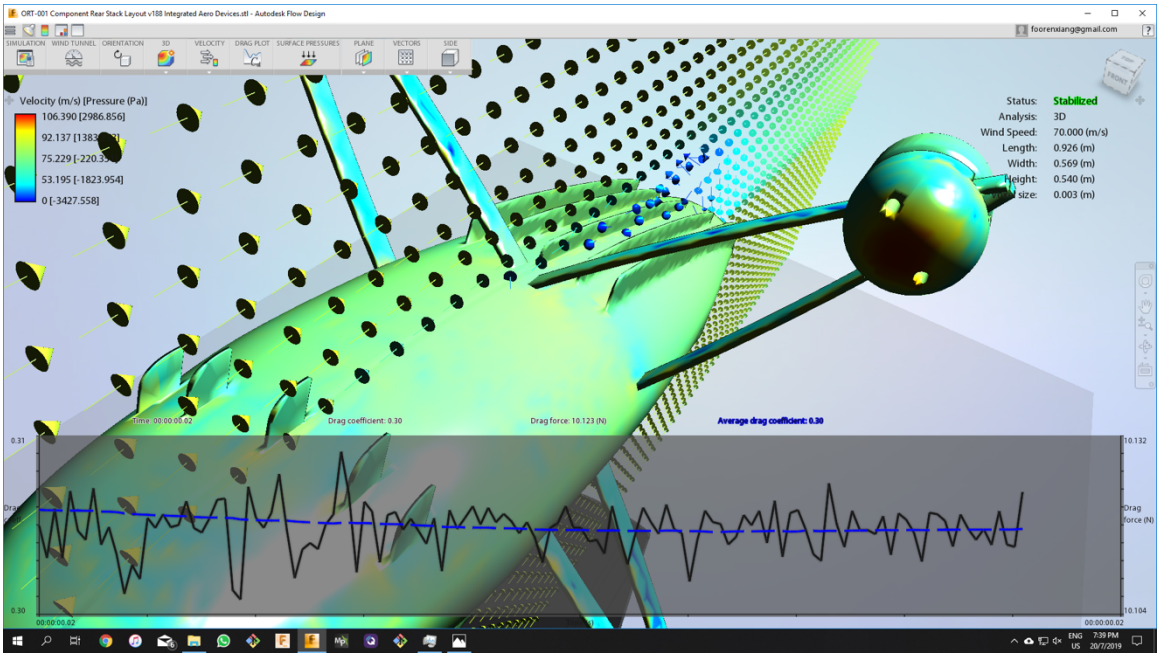


Fig 4.24Multiple stabilisers

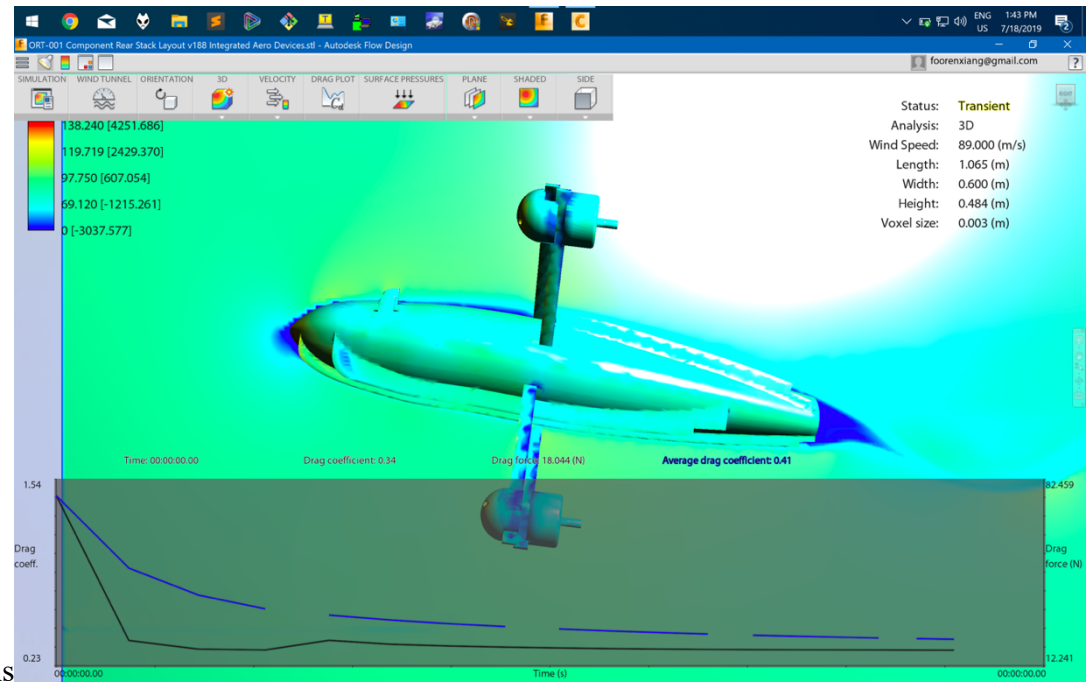
fins 

Fig 4.25 Stabilisers at the bottom of the body

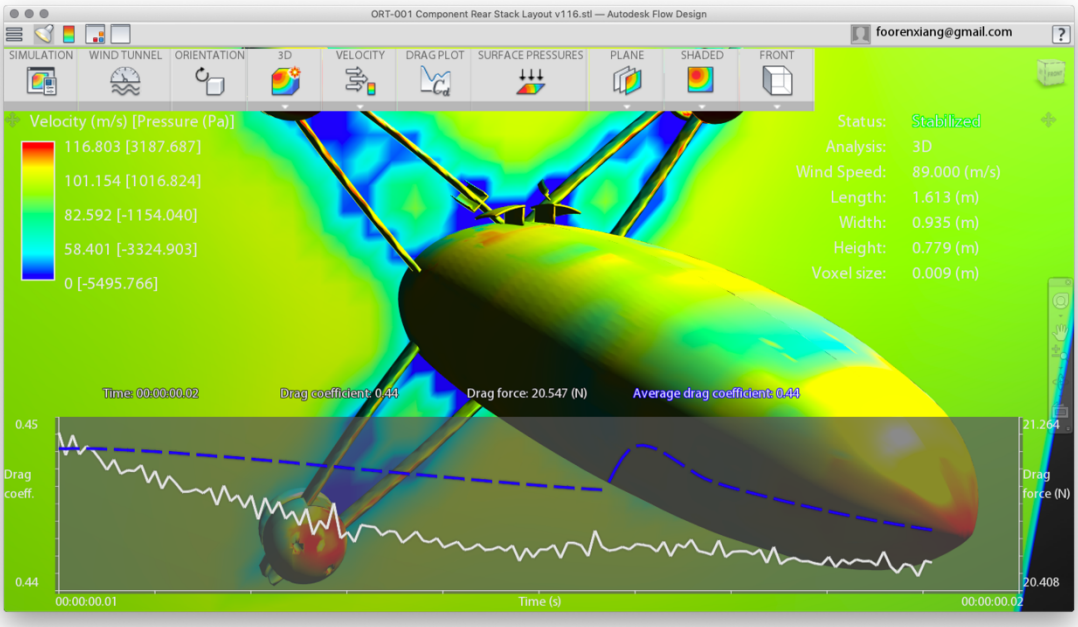


Fig 4.26 Turning vanes in front of arms

Winglets before the arms to redirect air downwards to the body. Re-energises the boundary layer by mixing in high velocity air, while avoiding turbulence generated by the arms

With the bodywork design complete, additional processing was required to prepare the model for fabrication:

- Pillars were extruded from mounting holes of the frame into the bodywork
- Holes were cut into the bodywork and pillars to form wells. Screws used to attach the bodywork to the frame would sit in these wells
- The bodywork was sliced into 3 separate pieces for modularity to form separate covers for the electronics and LiPo battery. Reducing the volume contained by piece also reduces replacement costs in the event of a broken section.
- Lips were added to each bodywork piece to aid alignment and increase aerodynamic seal
- Fillets were added to steep corners to increase strength and SLS printability
- A port was cut in the nose for the camera

The final design achieved a good centre of mass distribution, which was in the same plane as the motors. This would result in stable flight behaviour, as each motor would require a similar amount of torque to manage the weight of the drone.

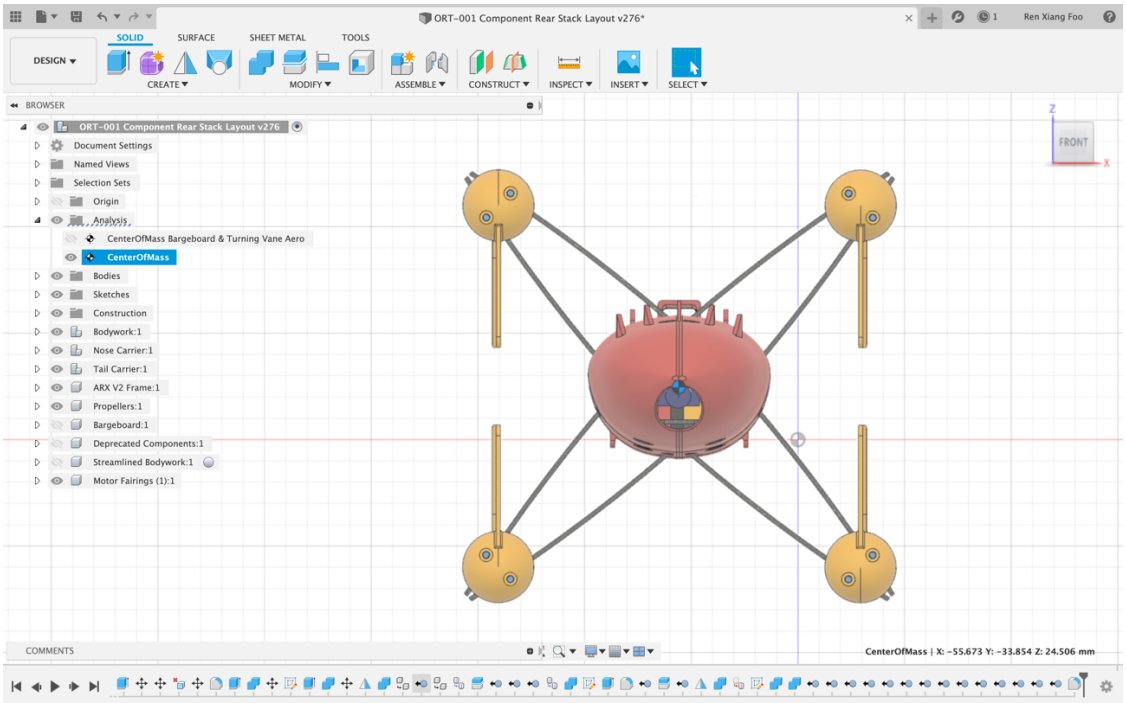


Fig 4.27 Center of mass, front profile view

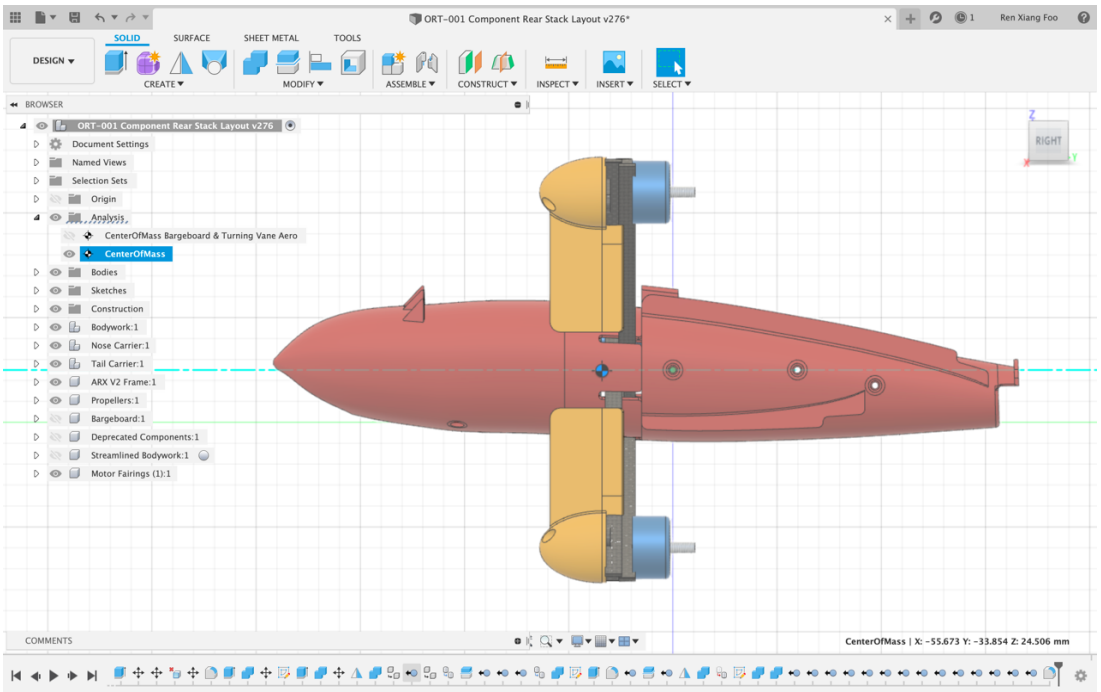


Fig 4.28 Center of mass, side profile view

The final CAD model:

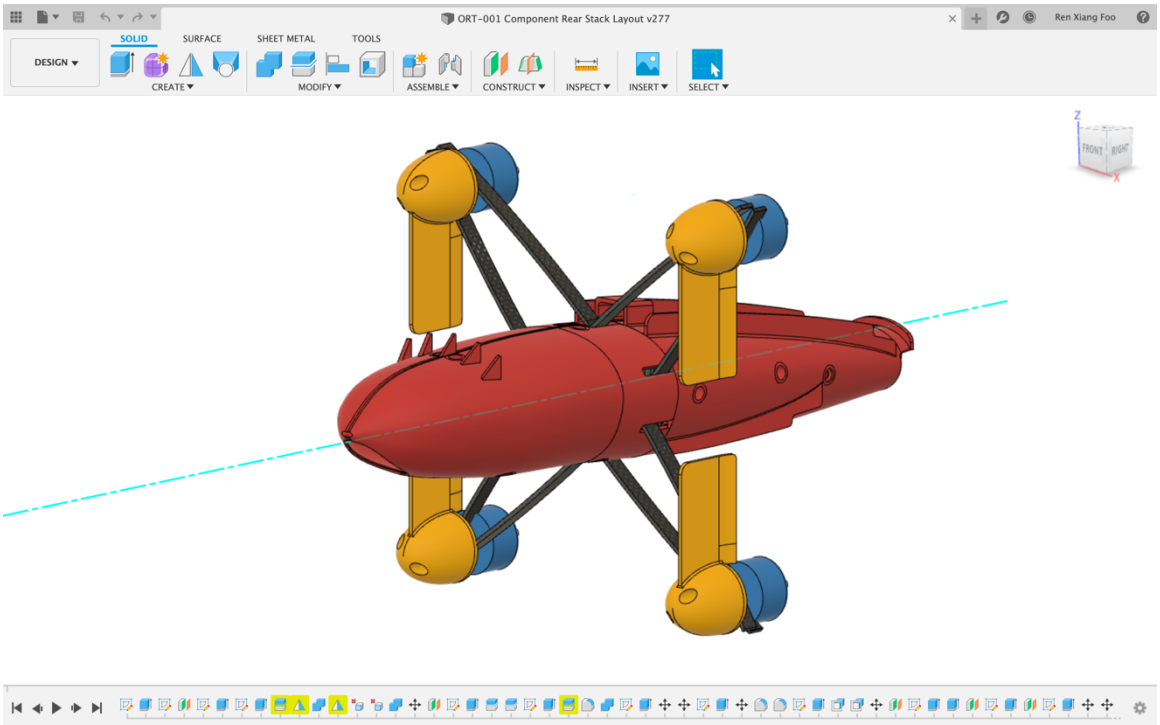


Fig 4.29 Final mUAV design, side view

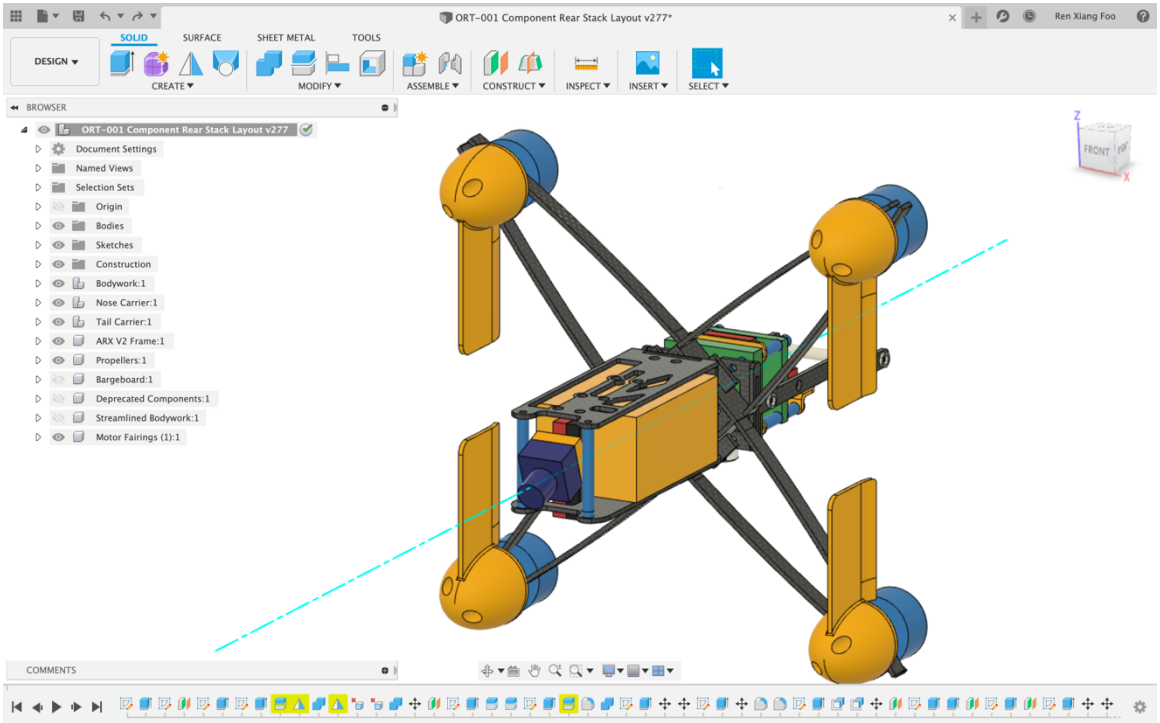


Fig 4.30 Final mUAV design, without bodywork

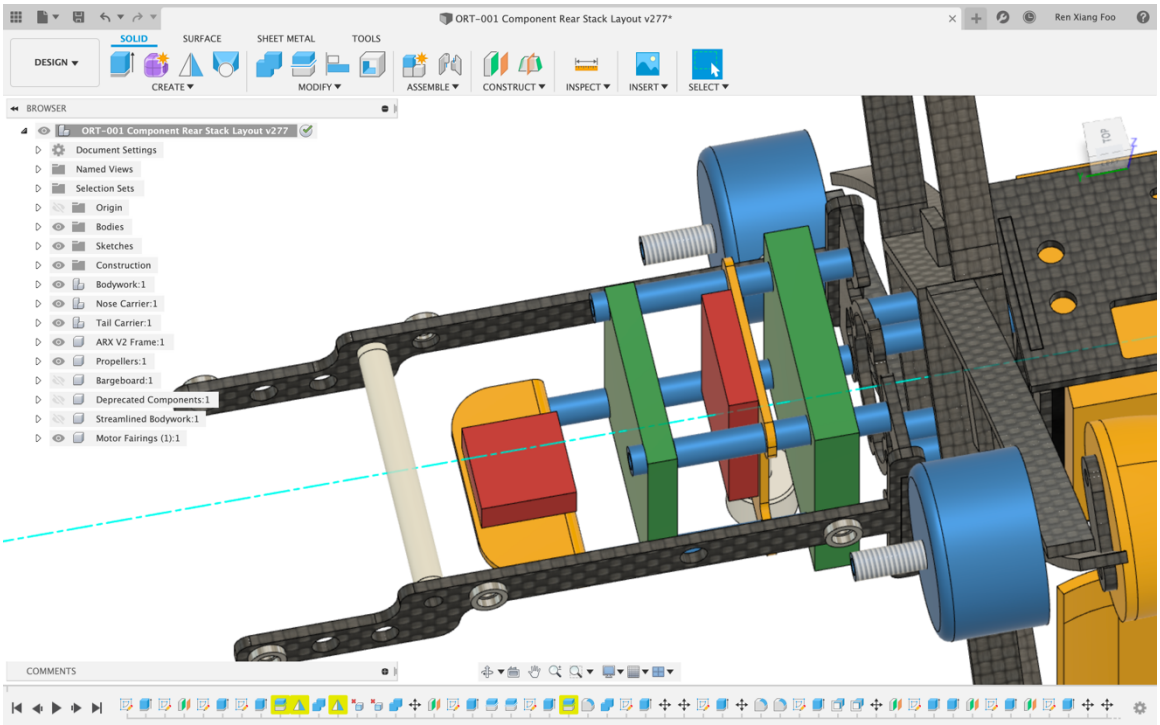


Fig 4.31 View of electronics

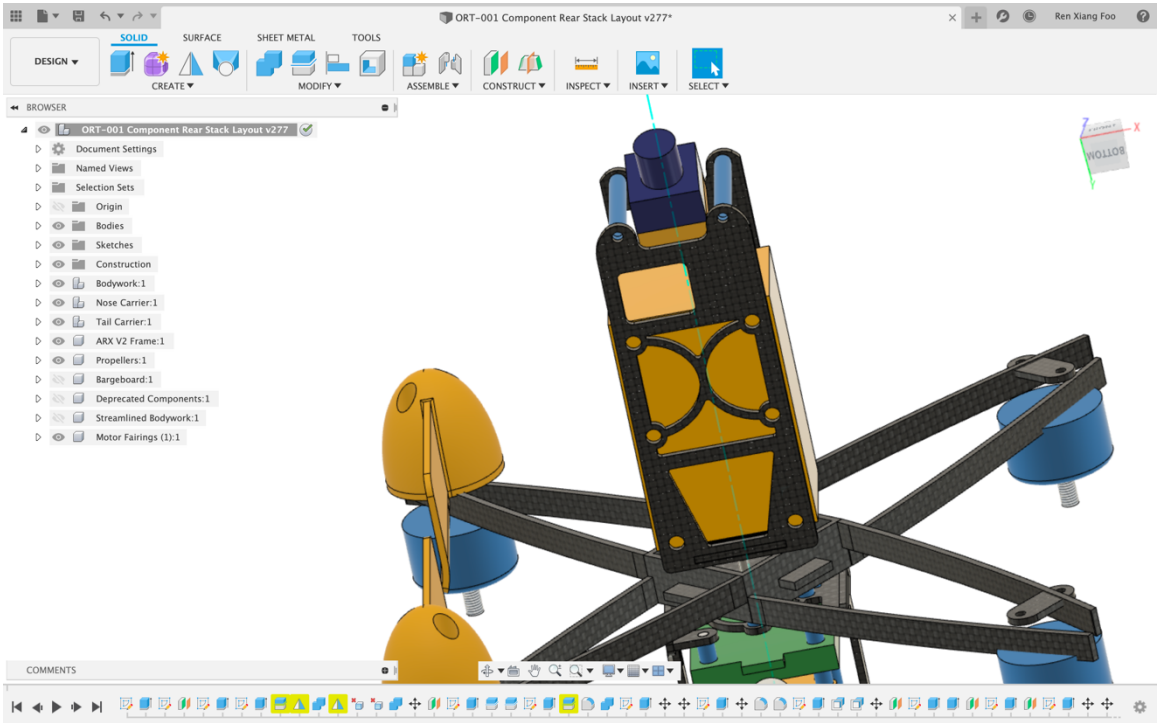


Fig 4.32 Bottom view of LiPo carrier

In total, over 650 CAD design revisions were made in the Fusion 360 project at the point of manufacture.

4.3 Manufacturing Process

Three manufacturing processes with different materials were used. The 3D fabrication was performed by a 3D printing house in Singapore, while CNC fabrication was initially done in Shenzhen, but subsequently changed to Taiwan.

Fused Deposition Modelling 3D printing (FDM) was used to produce the arm fairings, electronics mounting and launchpad. FDM provides fast and cheap manufacturing, but has limitations when printing complex geometry with overhangs. The parts cost $2-10 to print, depending on the volume.

Selective Laser Sintering 3D printing (SLS) using powdered Nylon 12 was used to form the bodywork. The design was not printable by the cheaper FDM process, due to wide and steep overhangs and precision required for the aerodynamic devices. SLS was chosen as the power material itself inherently forms a stable material to support printing of overhangs. It also has a high resolution of up to 0.2mm, due to the use of a laser, which are required for printing small details like fillets, precisely sized holes and sharp aerodynamic devices. It is also highly impact resistant with a degree of elasticity, that serves as impact protection due to its shock-absorbent properties. The SLS process was expensive, with the bodywork costing ~S$180 to print. However, it is still cost effective due to the longevity of the bodywork.

The frame pieces were CNC machined from 2mm thick carbon fiber plates with a hole tolerance of 0.25mm. The CNC fabrication process caused some delays in the project. A test run from the initial manufacturers that advertised low manufacturing costs was ordered.

The fabricated parts failed to meet the manufacturing tolerances required. After negotiating with the manufacturer, it was confirmed that their machines could not produce at the level of precision required, and a refund was obtained. The carbon fiber material provided was too brittle and cut with low precision cutting tips. This made the frame unusable. During installation of captive nuts (used to screw the bodywork onto the frame), the teeth of the nuts would dig into the material and cause it to strip, failing to anchor the nuts. In addition, the imprecise dimensions of the holes and pegs caused press fit pieces to not fit. The trial and error phase involved a long lead-time, as each production cycle took 1.5 weeks, due to production and shipping time. To solve the issue, a more expensive manufacturer with a good track record was tried. In addition, captive nuts that were compatible to the precision level provided by their manufacturing process were procured. However, they had a longer lead time of 3 weeks. This finally solved the problem and the frame was assembled. In total, about 8 weeks was required to resolve the issue.

Here are some photos of the assembled mUAV:

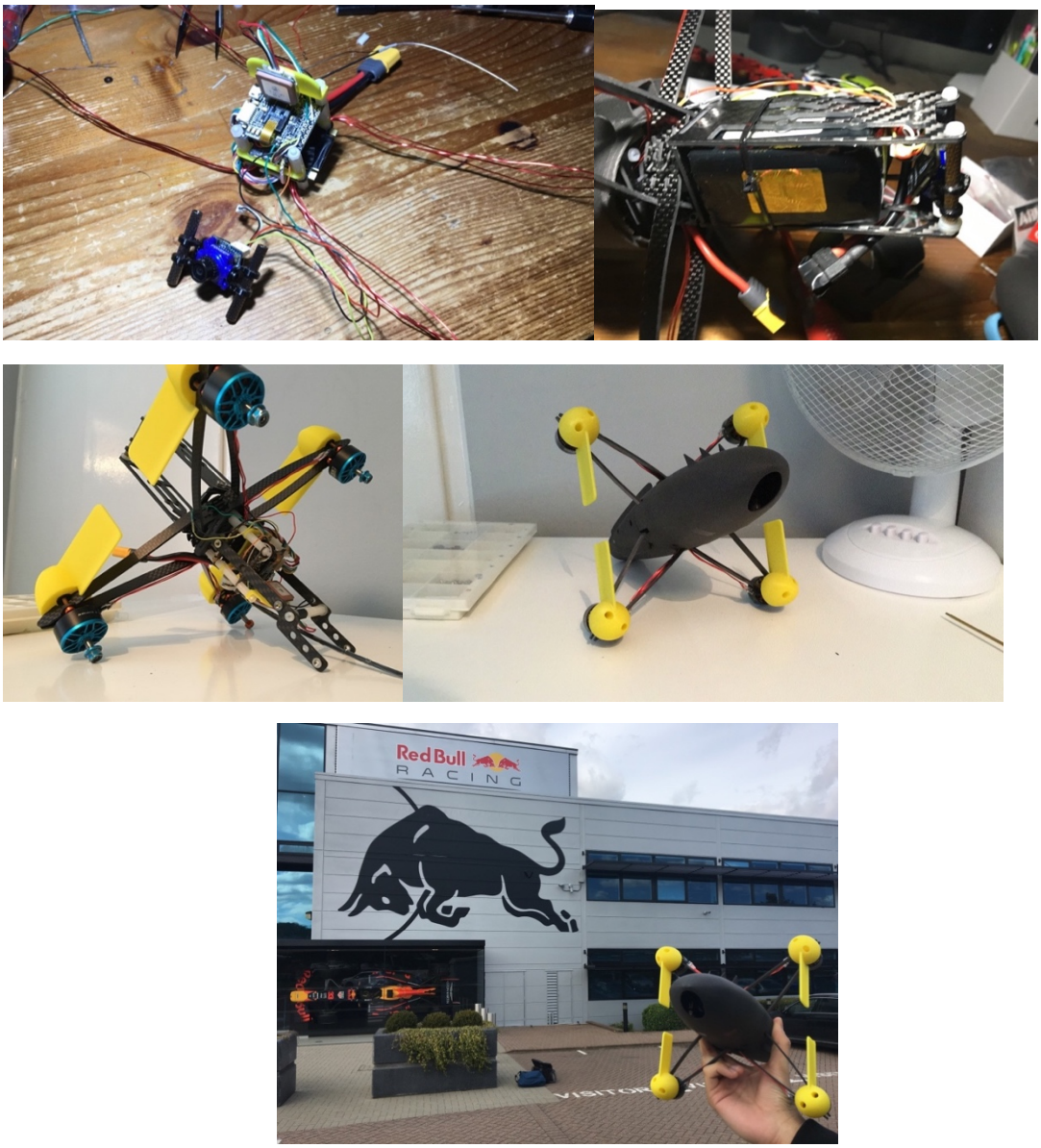


Fig 4.33 mUAV at Red Bull Racing Formula One Team HQ

A pitch was made to Red Bull Racing Formula One Team, presenting the still under development mUAV, as a potential high speed aerial photography platform for their race cars in promotional events. The goal of the pitch was to gain insight and feedback on what additional requirements that were to be considered, while also networking with them.

Chapter 5
UAV Rolling Launch Control (URLC)

5.1 Objective

The objective of proposing and developing UAV Rolling Launch Control (URLC) is to allow the autopilot system to outperform a human pilot in terms of control precision and reduce latency when adjusting to the everchanging mUAV powertrain conditions.

Fundamental Modelling Concerns when implementing UAV Rolling Launch Control:

- How to model pilot's & co-pilots' intuition when controlling the UAV?
- Must sufficiently account for unpredictable but significant factors, E.g.:
    - Changes in fluid (air) properties that can affect powertrain performance (Weather-based)
    - Varying intuition/condition of pilot/co-pilots that may bias/skew certain flight log samples
- Machine Learning model should model/surpass such intuition
- Pilot Intuition: What does the pilot observe/base his decision on when flying

The pilot has to multitask when flying:

- Ensure the UAV remains stable/correct angle of attack
- Get a feel of how fast the drone is accelerating/maintaining speed
- Know when he has to decelerate (run out of space to fly/LiPo has hit peak power delivery limit)
- Watch out for hazards in the area/potential malfunctions with the UAV

Due to the high workload from controlling a very fast machine that allows for little reaction time, he primarily **observes** the following

- Instantaneous GPS velocity
- Rate of change of GPS velocity (estimation/"gut feeling")
- G-force
- LiPo Voltage
- Voltage sag
- How straight and level is the UAV (angle of attack)
- Distance left to end of speed run

The pilot also **feels** the following (can be modelled as innate pilot intuition):

- Responsiveness of drone
- How much power can be delivered by the LiPo during the speed run
- How fast & how (generally) does the drone change velocity
- How long can top end speed be controlled
- How stable is the UAV? Will the tail end become unstable due to uneven aerodynamic forces?
- Is the UAV drifting due to crosswinds?
- Lifting body effect/aerodynamic drag on UAV when lifting and coasting
- Does the propeller noise sound erratic (abnormal efficiency losses/wear on propellers and motors)?
- Is the drone climbing or sinking?

The relationship of GPS speed and throttle input could also be modelled in the following manner:

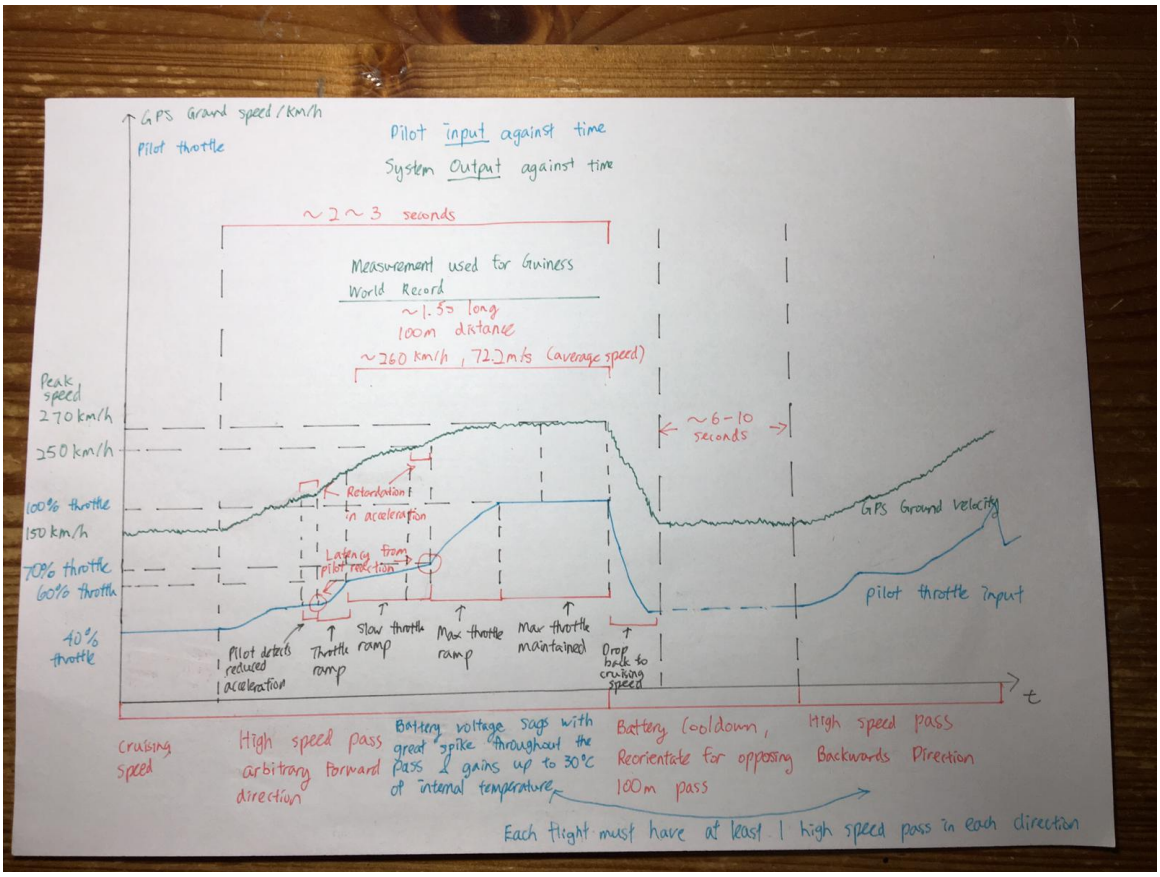


Fig 5.1 Speed and throttle behaviour during mUAV high speed pass

With the factors of the problem to be modelled determined, design of the URLC machine learning pipeline could commence.

5.2 URLC Design

The design phases of URLC:

1. Flight Data Collection
2. Data Pre-processing (Data clean-up and Exploratory Data Analysis)
3. Database Schema Design
4. Automation of Data Ingestion and Pre-processing
5. Machine Learning Pipeline Prototyping

6. Evaluation and selection of individual machine learning models in cascaded pipeline
7. Development of application that simulates a real-time telemetry stream evaluate real-time prediction performance replay
8. Migrating application to cloud

Flight Data Collection

The Blackbox logs collected during field testing were used. The DVR recordings were not directly used in the implementation of URLC, but remained very useful during data pre-processing to gain an intuition of the quantitative data collected in the Blackbox logs.

Data Pre-processing: Data Clean-up

Data of each flight are saved as 2 separate files (GPS and PID logs) by the Betaflight controller, due to the high sampling rate of the PID controller of 1KHz. Including GPS data within each row of PID data would result in a large CSV file with a lot of redundant data. The average size of the PID data CSV file containing data for ~2 minutes of flight time has a file size of ~5MB, while the GPS CSV file is ~1KB.

Before the data can be used, the PID and GPS data must be merged together and values that are invalid (missing or corrupted values) have to be dropped. A filter based on minimum speed of 60km/h is then applied to exclude flight data recorded during landing and hover. Derivative features were calculated from existing columns:

- Time delta between samples, based on PID loop time
- Acceleration, derived from time delta between samples and GPS velocity measured of each sample
- Unit conversions:
    - m/s → km/h
    - acceleration m/s^2 → G-force (g)

Manual data cleaning was applied on the Blackbox logs, prior to the completion of the automated data ingestion and pre-processing development of the URLC pipeline.

Data Pre-processing: Exploratory Data Analysis (EDA)

Exploratory Data Analysis was then performed on the cleaned up data using the Speed Run Analyser Excel Sheet and TIBCO Spotfire Analyst. This allows certain trends and correlation between features in the data to be identified. The analysis was done on flight data aggregated over 13 flights. These logs were chosen from flights performed on a single day with an identically configured mUAV, with minimal missing or corrupted data.

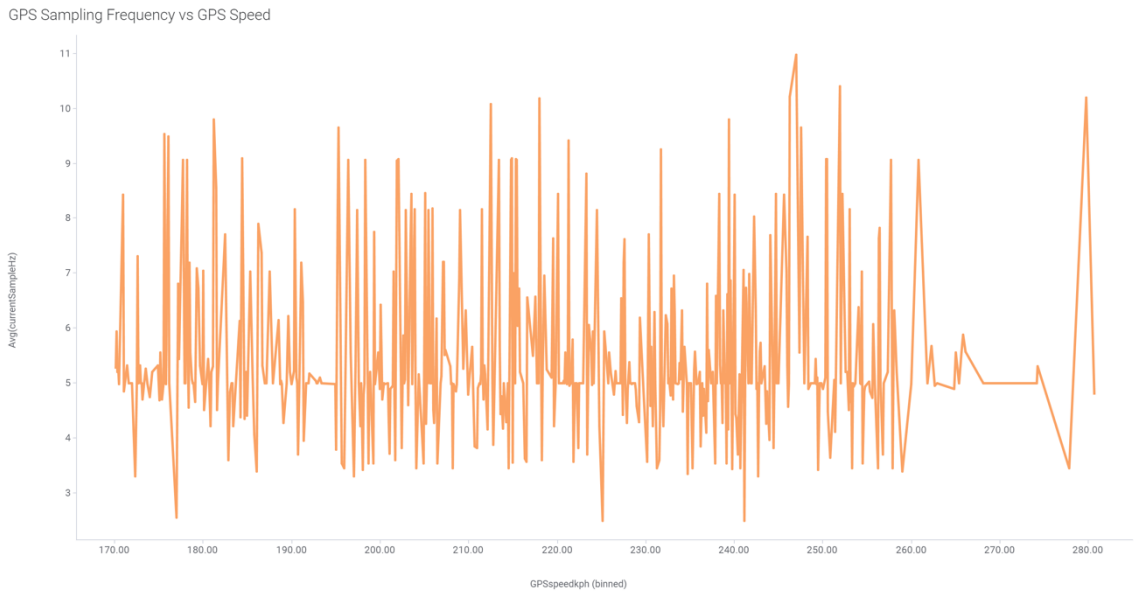


Fig 5.2 GPS Sampling Frequency vs GPS speed

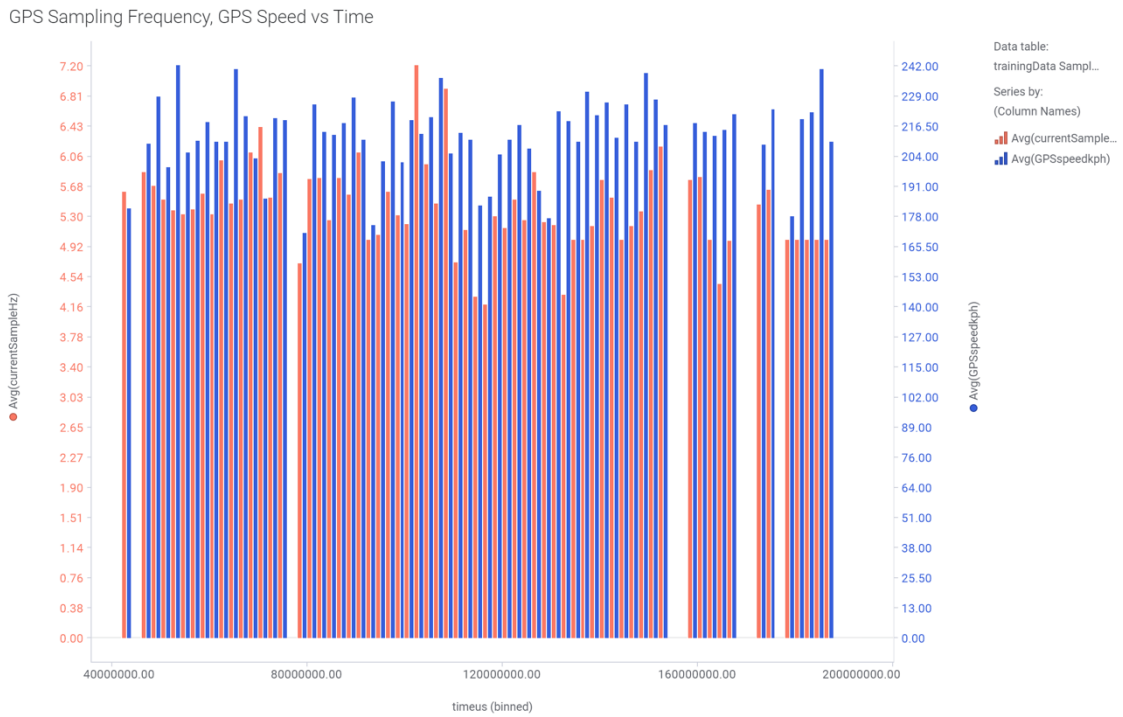


Fig 5.3 GPS sampling frequency, GPS speed vs time

Based on GPS sampling rate at different speeds, it was confirmed the GPS reception remained stable at high speeds. This would provide an accurate speed reading.

Plotting GPS sampling frequency over timespan of flight indicated that GPS reception was also stable.

Based on the stability of GPS sampling rate data due to good satellite lock, it was deemed that the GPS ground speed captured by the mUAV has sufficient accuracy for use.

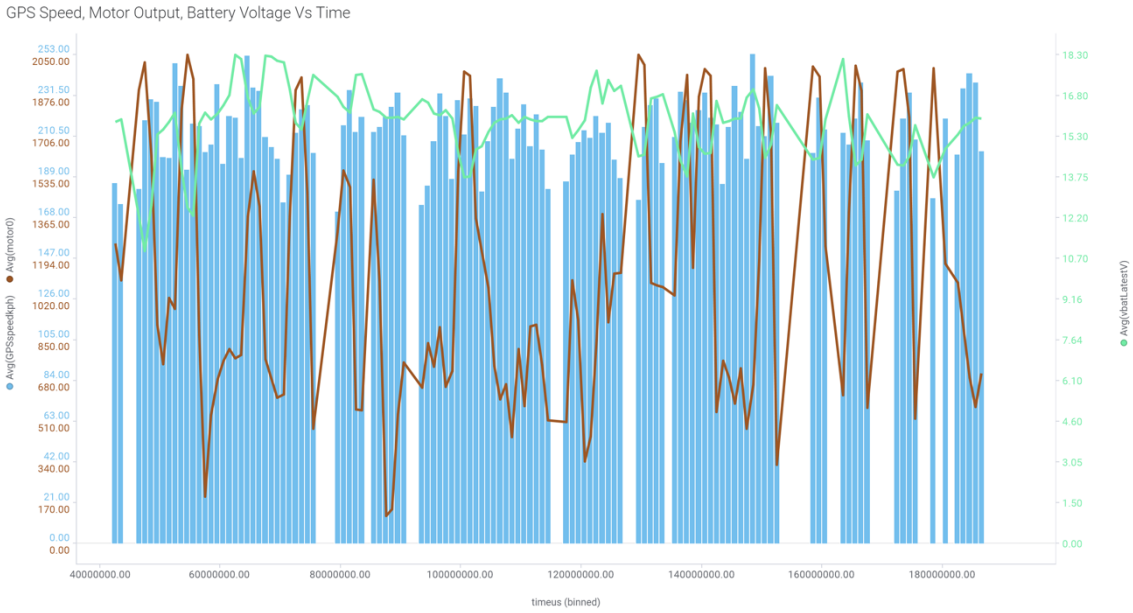


Fig 5.4 GPS Speed, Motor Output, LiPo Voltage vs Time

Observing the trend of motor RPM, LiPo voltage and GPS speed over time, it was confirmed the flight data captured matches the observed behaviour of the mUAV during flight.

The GPS speed does not immediately follow the trend of motor RPM (which is controlled by transformed pilot input of the PID controller), due to the momentum of mUAV preventing an instantaneous change in velocity.

The positive acceleration in GPS speed (positive gradient) is due to the high thrust to weight ratio of the mUAV of ~13:1 (~8kg of thrust : 0.6kg weight). Correspondingly, the large deceleration when thrust is lowered, indicates high aerodynamic drag.

The negative gradient of LiPo voltage indicates the depletion of available energy in the LiPo as the mUAV was flown, with the sinusoidal trend indicates increase of LiPo internal resistance (which is a function of LiPo voltage) as the LiPo starts to overheat from sustained high power delivery during high speed passes, with internal resistance decreasing again during cooldown as throttle is lowered.

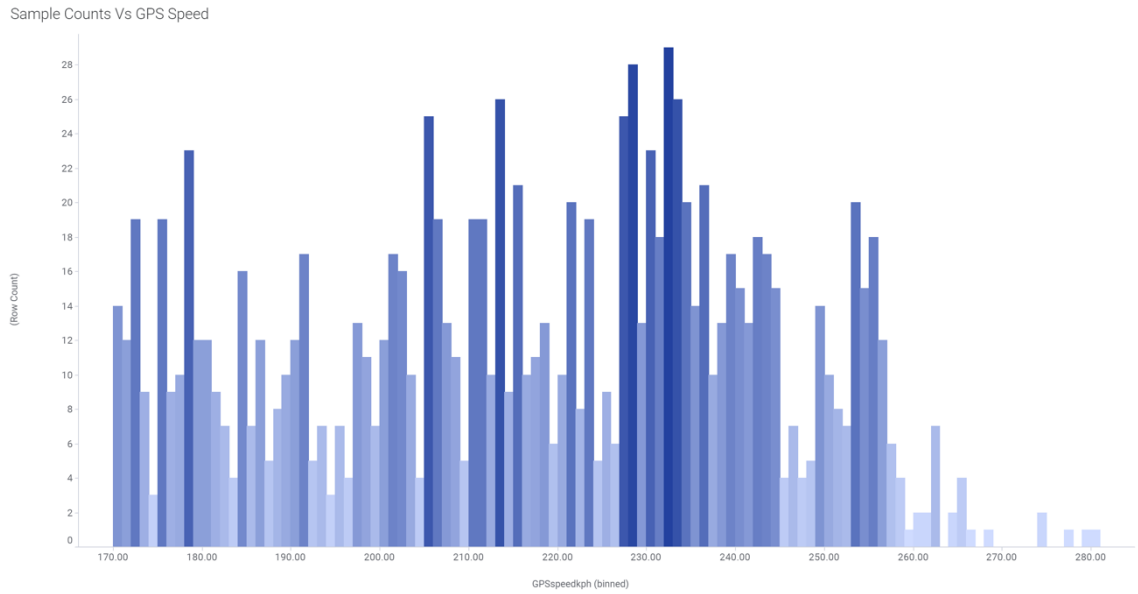


Fig 5.5 Sample counts vs GPS speed

Plotting sample count over GPS speed provides a distribution of average instantaneous GPS speed of the mUAV. This is useful for filtering values to be used for modelling the mUAV kinematics behaviour. In this visualisation, the values were filtered to 170km/h. Multiple peaks are observed. They can be attributed cruising speed, which averages around 200km/h, and speeds recorded as the mUAV was accelerating up to top sustained speed during a speed run. Small sample counts at the right of the graph, with the highest GPS speed values, indicate the peak instantaneous speed reached by the mUAV.

MATLAB Regression Learner was also used to quickly prototype and test which regression machine learning models were suitable in modelling the kinematics of the mUAV, in order to predict GPS speed and LiPo voltage values. Feature engineering was also performed, to identify and only select features that were useful for prediction. This will speed up machine learning prediction speed and accuracy, as redundant and/or misleading data are excluded from consideration. The impact of unsupervised dimensionality reduction using Principal Component Analysis was also tested, on top of feature engineering, with sufficient principal components kept explaining 95% of variance of data in the selected features.

The following classes of models were tested:

- Linear Regression
- Support Vector Machines (SVM)
- Regression Trees
- Regression Forest
- Gaussian Process Regression

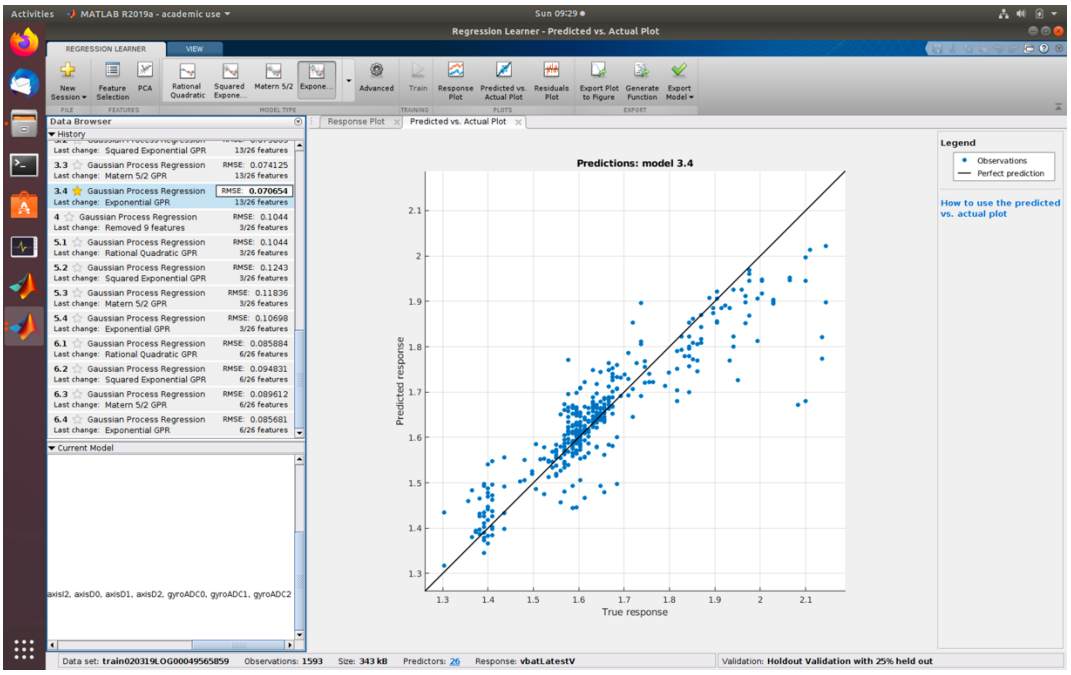


Fig 5.6 Regression prediction of LiPo voltage

LiPo Voltage prediction using Gaussian Process Regression with GPS, throttle and powertrain data features RMSE: 0.7V. It was noted outliers existed around the >20V and <15V range. The data for >20 voltage range is not useful for URLC deployment, as it indicates LiPo voltage during take-off, that promptly drops off and remains constant at ~18.5V before the first speed run occurs. The <15V range indicates complete depletion of the LiPo battery, which would occur after all speed runs are completed. These conditions will not be present during deployment of URLC

This solution would be to add an additional filter to drop data in these ranges from the training set during data pre-processing, to increase accuracy of the LiPo and GPS speed prediction models in the relevant range required for URLC.

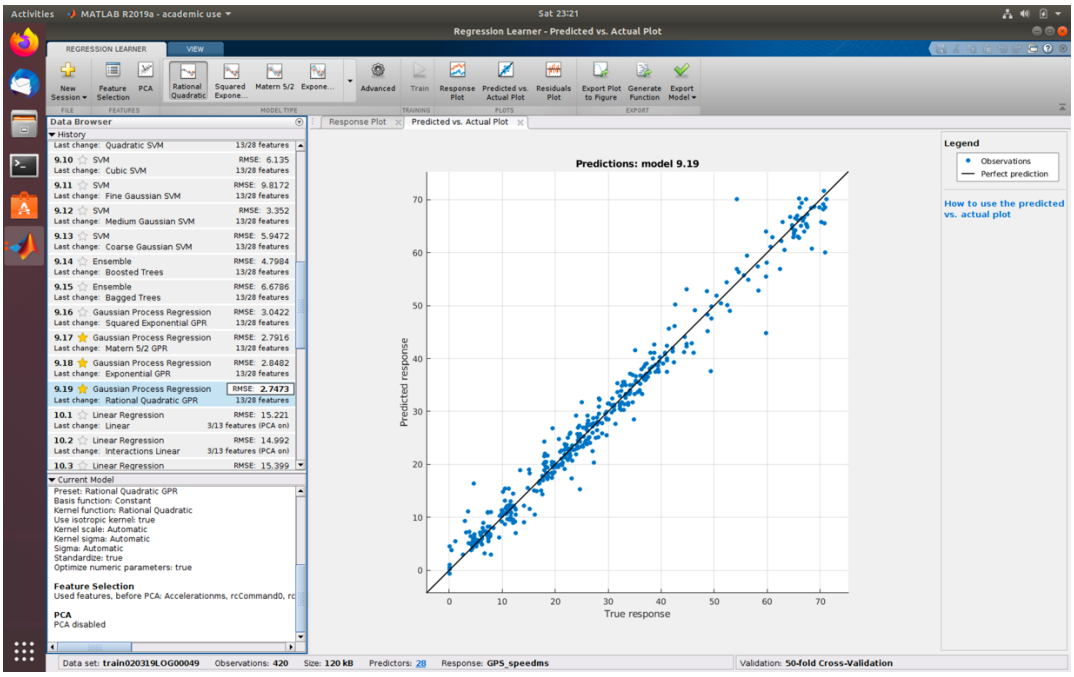


Fig 5.7 Regression prediction of GPS speed

GPS speed prediction using Gaussian Process Regression with LiPo Voltage, throttle and powertrain data features. RMSE: 2.74km/h

From the prediction response graph and low RMSE, the prediction was quite accurate. Having gained an insight of the data trends and validating the fact that in general, regression models can model the GPS speed and LiPo voltage data, URLC can be built upon these findings.

In addition, deep feature synthesis was tested and used to generate products of original features, with the aim of investigating if any new engineered features would increase the accuracy of the models. However, this was not actively used, due to the time required to generate the samples, and the file size of the final generated data. For example, 5 minutes of flight data with 1500 samples and 33 columns and initial file size of 20MB, would require 40 minutes to perform deep feature synthesis. The final synthesized data will have >70,000 features and 900MB. It was unfeasible to perform EDA with the resulting data, as Excel and MATLAB could not load the data due to the high feature count and file size.

This could be further explored by tuning it to reduce the number of features to be generated.

Database Design

With the application being time-series data driven, a core component of the URLC software is a database.

A database called kdb+ from Kx Systems was used, due to the following features:

- Fast performance when dealing with live data streams and large time-series datasets.
- It has a built in programming language called q that provides good scripting capabilities with close integration with BASH on Unix systems.
- Built-in Python interpreter, useful for running Python-scripted machine learning models that can directly exchange variables between the q and Python memory space, reducing data exchange overhead
- Has inter-process communication. This allows instances of kdb+ processes, hosted on different machines and running different specific applications, to be easily networked together. This functionality is useful for deployment of the application modules to separate cloud virtual machines.

  It is performance optimised due to being columnar data store and storing its real-time database in RAM. Using columnar store of data tables allows data to be read faster, as each column is processed in a single vectorised CPU instruction. Storing the real-time database in RAM provides faster I/O access in memory than on disk, when performing create, read, update, delete (CRUD) operations data typical of SQL queries. It is typically used in industries that require low latency processing of time-series data from large data sources. This includes financial data, automotive, oil & gas and digital marketing.

Building upon the powerful scripting capabilities of q/kdb+, the URLC application was designed with q processes serving as the main process that controls and manages key functionality of each module for consistency.

Due to the linear relation of all features in the flight data, a simple database table design was required. This same structure exists for the historical database and real-time database in the system. The historical database is used to archive older data entries that are no longer regularly accessed, by saving them to disk and removing it from the real-time database. This reduces memory use.

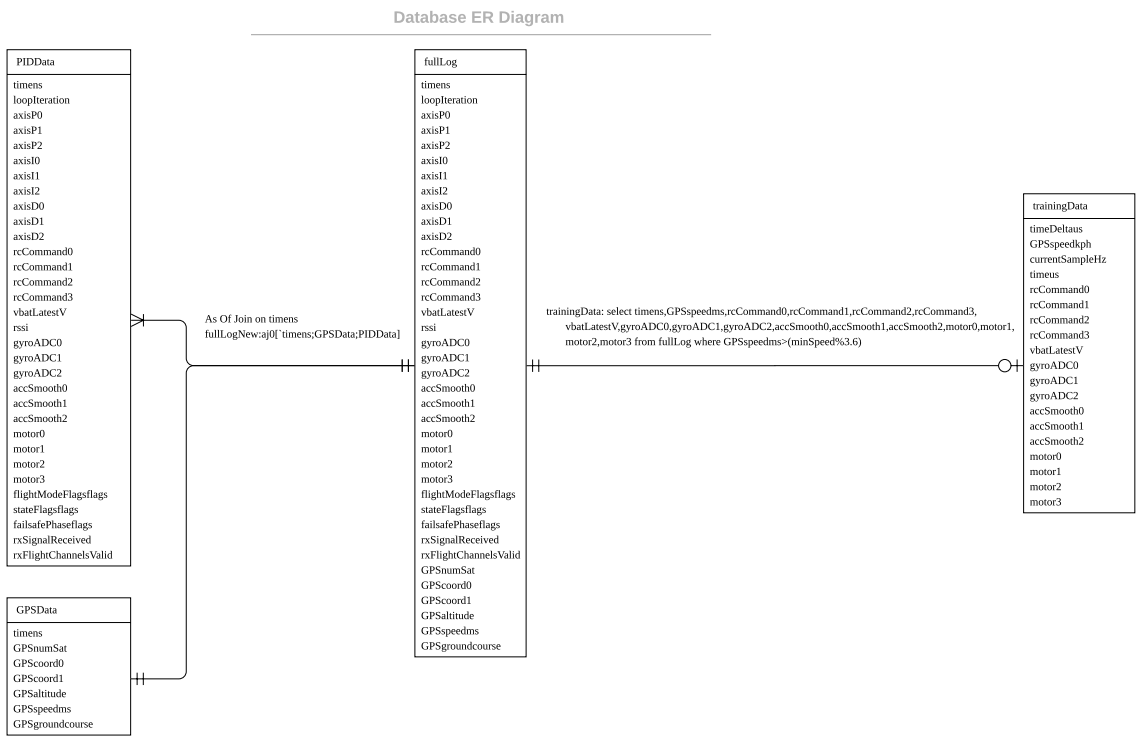


Fig 5.8 Entity-Relationship Diagram of Database

Automation of Data Ingestion and Pre-processing

To reduce the manual work required to prepare the historical flight data of the mUAV from Blackbox recordings for use in data analysis and machine learning, a data pre-processing module was developed. It consists of:

- Web upload interface implemented in PHP and JavaScript.
    - JavaScript to handle the user interface (UI) elements

  - JavaScript also connects to the kdb server via q WebSocket
  - WebSocket connection prompts user for credentials in order to view and upload data logs
  - An upload form for selecting GPS and PID csv logs to be uploaded
  - PHP creates a manifest of files uploaded and records the number of columns in each file, saved as a CSV file
- Backend q script, that is triggered to perform q-SQL queries to clean the data and populate the real-time database tables

The process is captured by the flow chart on the next page:

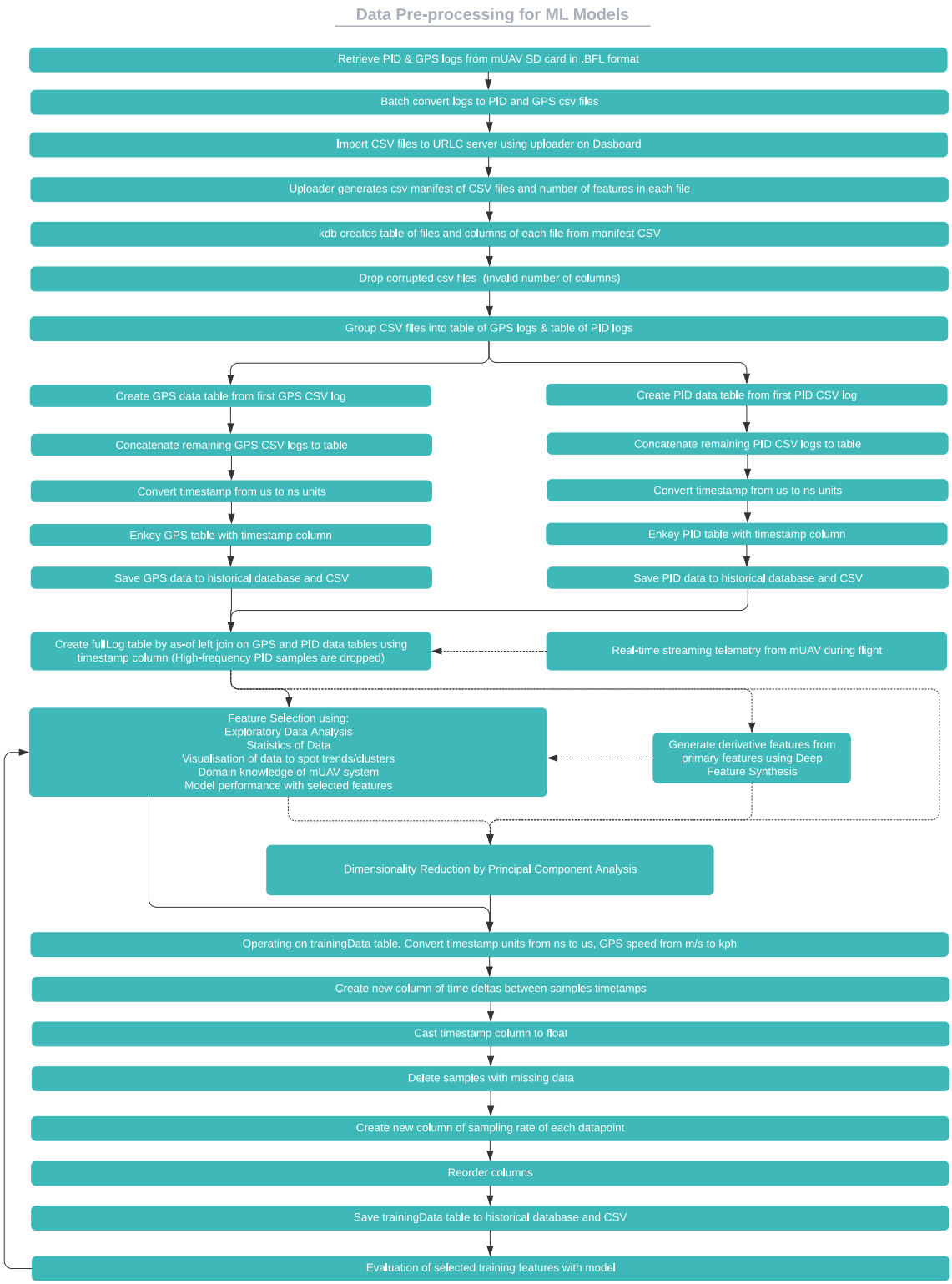


Fig 5.9 Data Pre-processing for ML models

Machine Learning Pipeline Prototyping

Based on the findings during Exploratory Data Analysis, it was determined that regression models could accurately predict GPS speed and LiPo voltage predictions. In order to use that prediction capability to generate the optimal target throttle control output from real-time telemetry data, the following training pipeline on the next page was proposed:

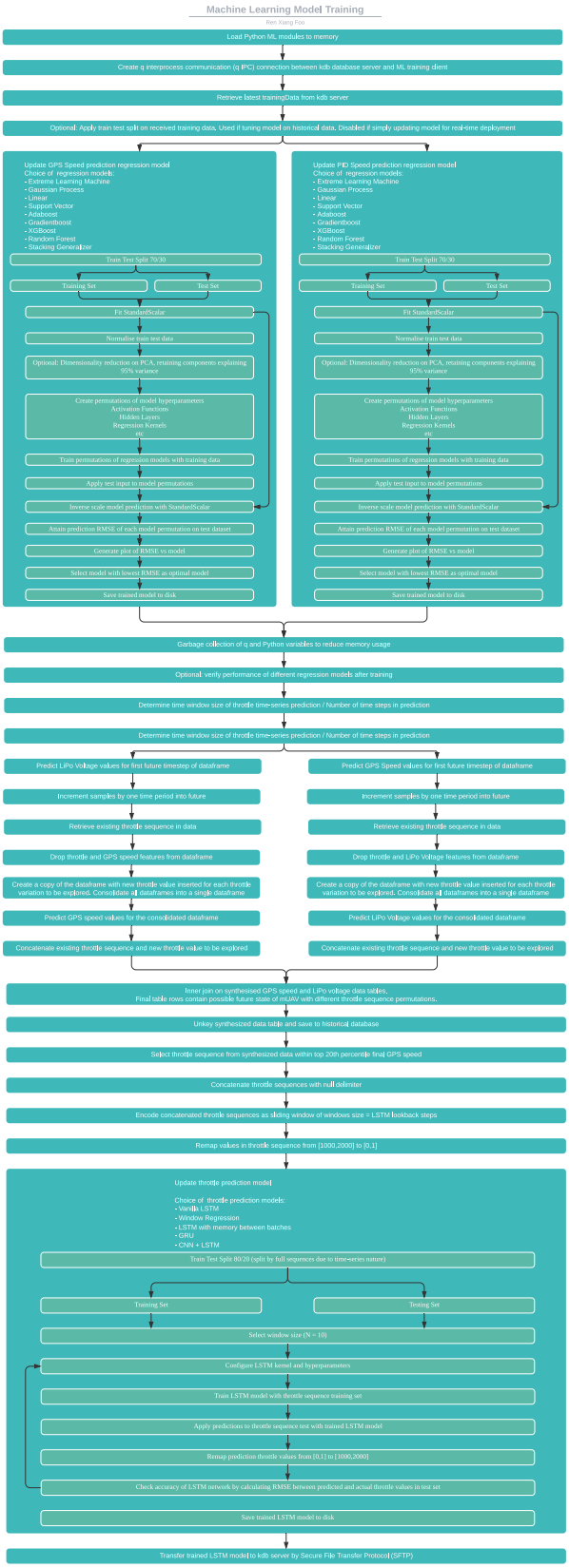


Fig 5.10 Machine Learning Model Training Pipeline

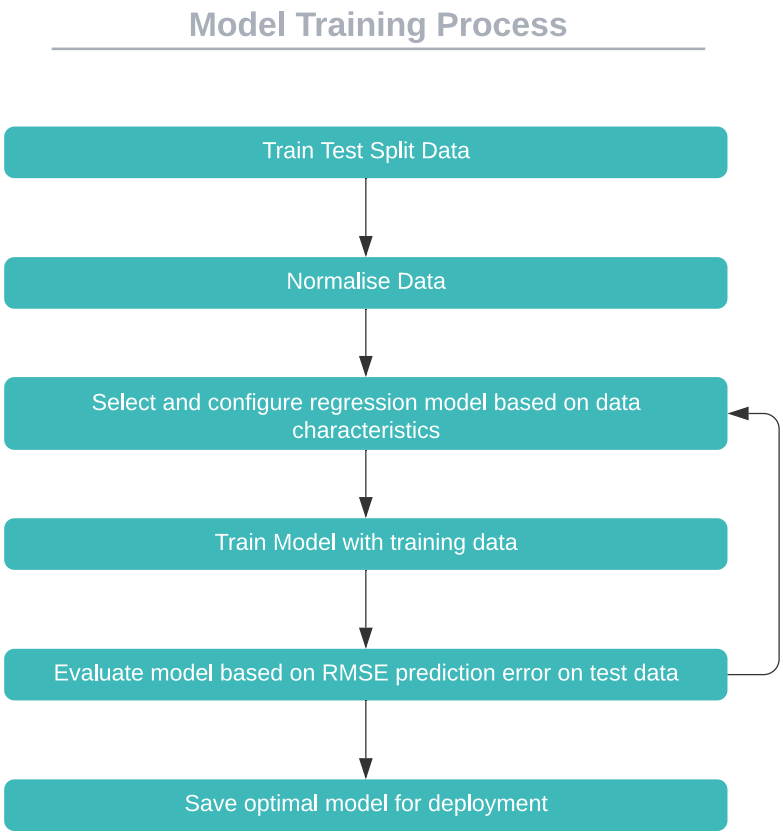


Fig 5.11 A simplified version of the model training cycle

Unsupervised dimensionality reduction using Principal Component Analysis was performed on the dataset of historical flight data, as the training data had 18 features. Dimensionality reduction speeds up machine learning training and prediction as lesser features have to be transformed by the model. In the URLC implementation, principal components explaining up to 95% of the data variance was kept. Accuracy of the model may also increase as a result, as noise in the data may be captured with the least significant principal components and dropped during dimensionality reduction. This was clearly observable when performing GPS speed prediction using Support Vector Regression with

and without PCA, as the calculations for the decision hyperplane equations are sensitive to outliers that may be caused by noise.

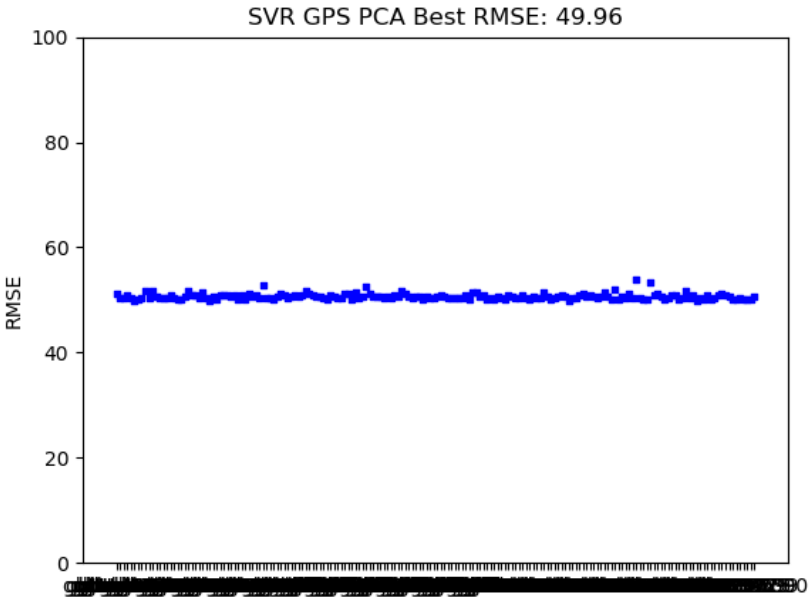


Fig 5.12 Support Vector Regression with PCA on input data

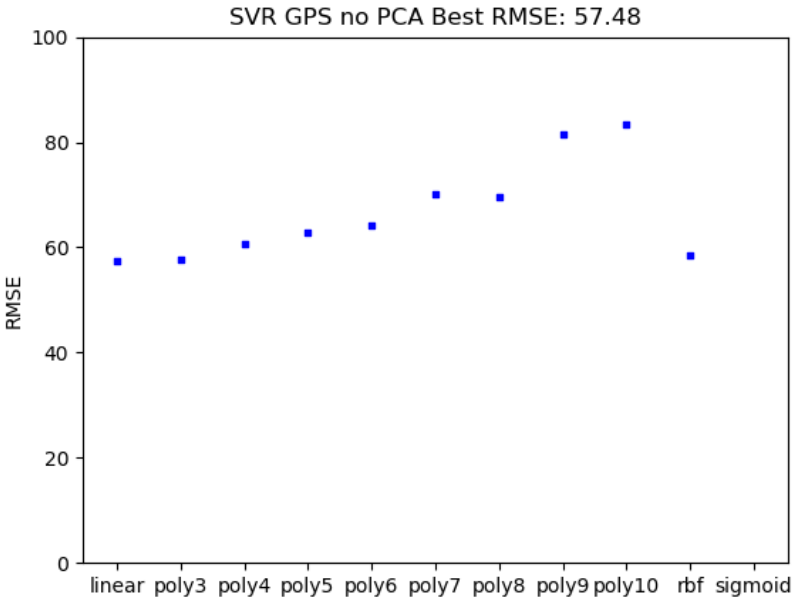


Fig 5.13 Support Vector Regression with PCA on input data

A regressor is used to model the GPS speed of the mUAV, while another regressor is used to LiPo voltage of the mUAV. Both models are fed with the same input features, with exception of the field it is predicting:

1) GPS Speed
2) LiPo Voltage

Various models were tested, with Gaussian Process Regression and Extreme Learning Machine performing with the best accuracy. Gaussian Process Regression (GPR) was observed to be more accurate with small training datasets (~300 samples), in comparison to Extreme Learning Machine (ELM). However, the accuracy of ELM and GPR starts to converge and becomes indistinguishable when the training data includes a larger number of samples (~1.6K samples). In the development environment, ELM models trained 44x as fast as GPR and uses a constant 3GB of RAM. Meanwhile, GPR RAM and CPU usage increases as training set size increases. The final model selected for both GPS speed and LiPo Voltage prediction was ELM, due to its sufficient accuracy and training/prediction performance.

The follow RMSE errors of various models with different hyperparameters are as shown:

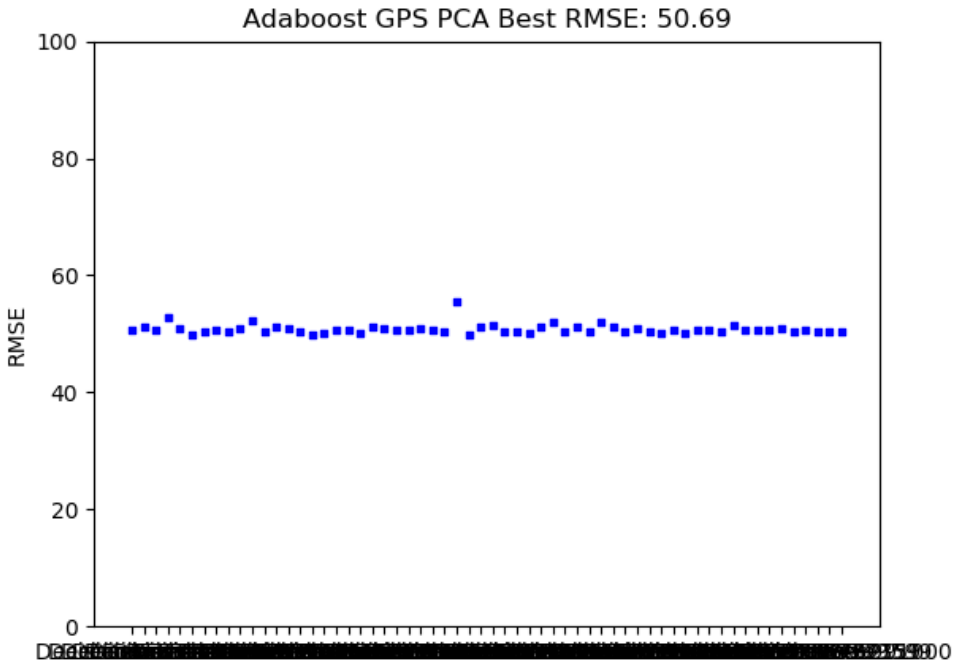


Fig 5.14 Adaboost regression for GPS speed prediction

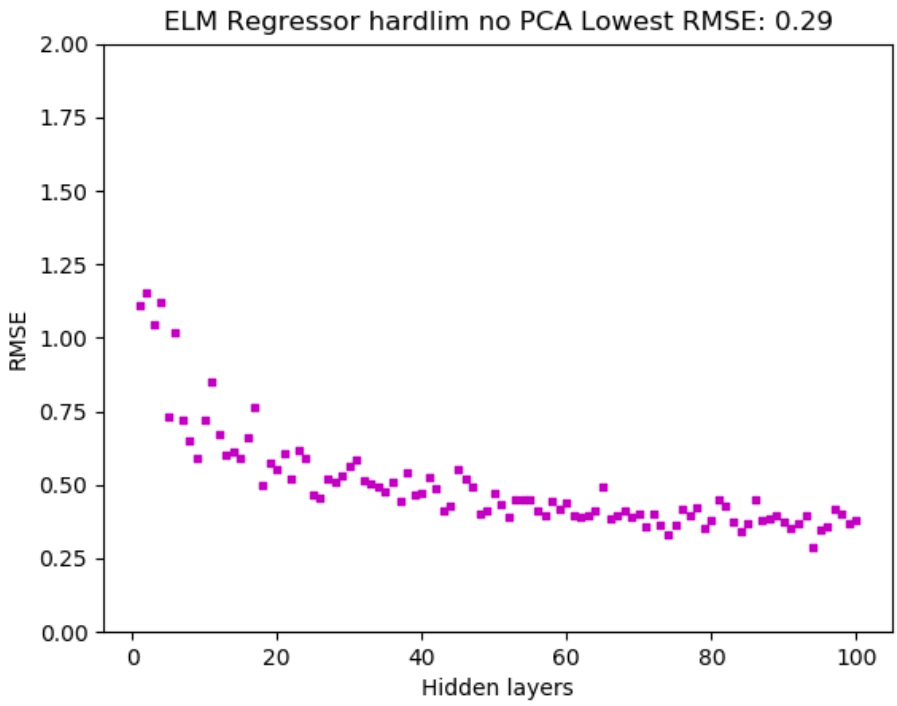


Fig 5.15 ELM regressor for LiPo Voltage prediction

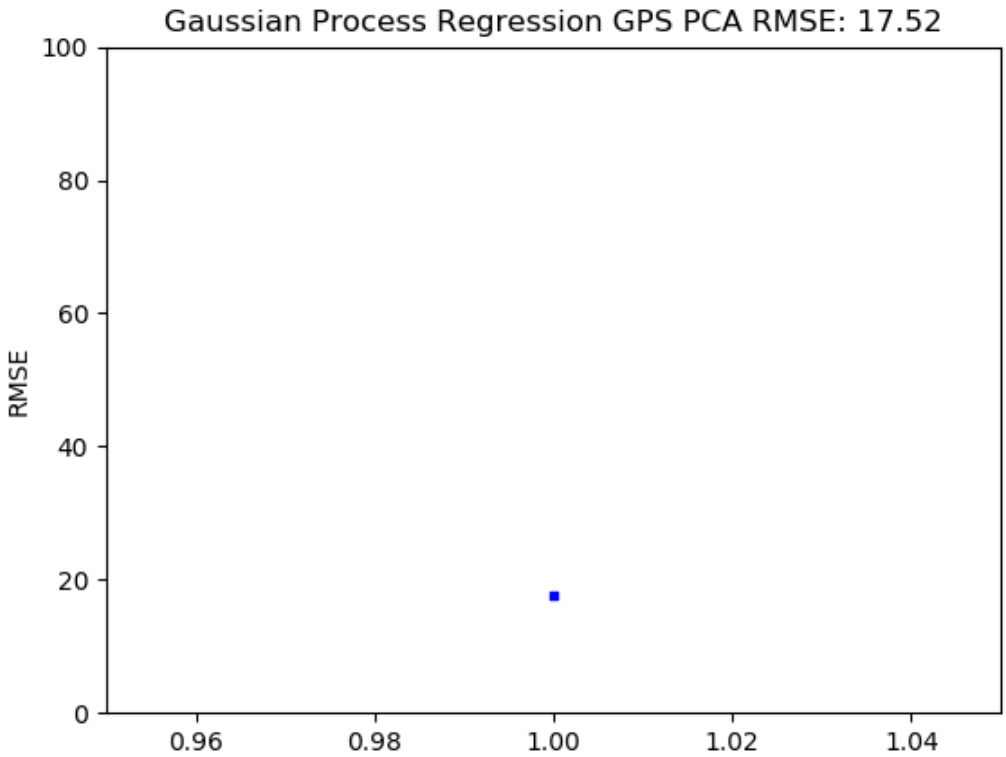


Fig 5.16 Accuracy of Gaussian Process Regression for GPS speed prediction on small dataset (~300 samples)

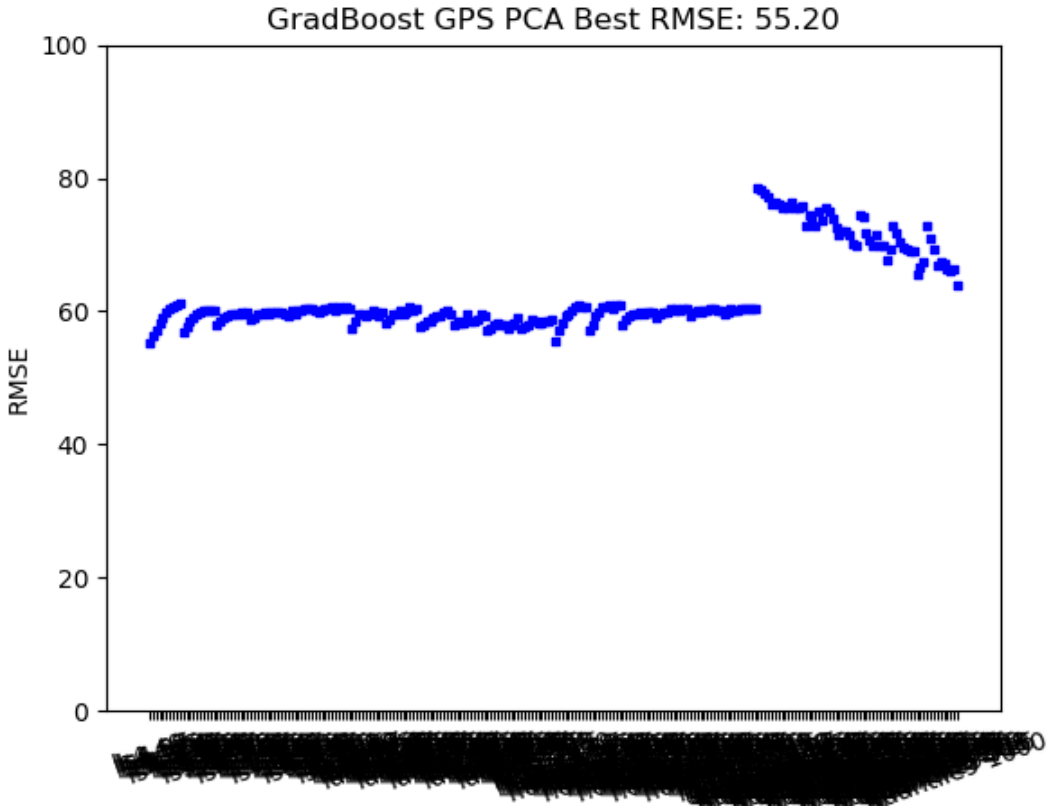


Fig 5.17 Gradient Boosting GPS speed prediction on small dataset (~300 samples)

These models are used in a recurrent cascade, iterating until enough samples are generated for the LSTM network lookback requirement.

To synthesize the throttle data, an assumption for the gyroscopic property of the mUAV is required:

- Due to the short period of each timestep between throttle changes, alongside with the stability of the PID controller, the mUAV will not experience significant rotation of its fuselage when doing a high speed pass. Hence, the gyroscopic readings of the data will not change significantly across several timesteps.

As such, the data of the IMU can be frozen and fed between the cascaded and recurring GPS speed and LiPo voltage prediction models.

An optimisation trick was devised to perform a quick pseudo search on the synthesized throttle data. This data has unsorted array structure, which is computationally expensive to search, due to a $M*N^T$ node growth (M is the number of real time-series samples, N is number of child per real/synthesized sample, representing throttle resolution, and T is the size of the future prediction steps window). The final unsorted array will be difficult to search, as M values can be quite high (M = 18K real samples for just 6 hour of flight data, with N = 10 for throttle resolution of 10%, and T = 10 for 2 seconds of lookahead data, totalling $1.8*10^{14}$ nodes) in a deployed system. The trick exploits the selection metric property of optimal throttle sequences being based on final GPS speed and LiPo voltage, with the length of throttle sequences being expanded on iteratively. Each throttle generation iteration encodes the previous throttle values that have occurred to reach that state. This places the values into a simple kdb table. When the complete throttle sequences have been generated, the table is spliced using a q-SQL query, with the new table containing only complete throttle sequences and relevant columns for LSTM training. The selected throttle sequences are then sorted by final GPS speed in descending order, with the first 20% results retrieved. They contain the top 20% final GPS speeds. This is a trivial task when performed using kdb, taking less than 1ms to complete for 41,000 rows.

Before the throttle sequences can be fed to the LSTM network, that have to be re-encoded in a time shifted window. This is done using a q function as a vectorised operation for efficiency, due to the large size of the data. The final throttle sequences are used to train the Long-Short Term Memory network, a variant of the Recurrent Neural Network that selectively forgets past data depending on current input state. The optimal flight behaviour and control characteristics of the mUAV is effectively encoded into the LSTM network when trained with synthesized data.

The trained LSTM network can predict optimal future throttle output sequences using just the current throttle input sequences at the end of the machine learning pipeline. The throttle

prediction LSTM performed relatively well when tested using synthesized and real historical flight data:

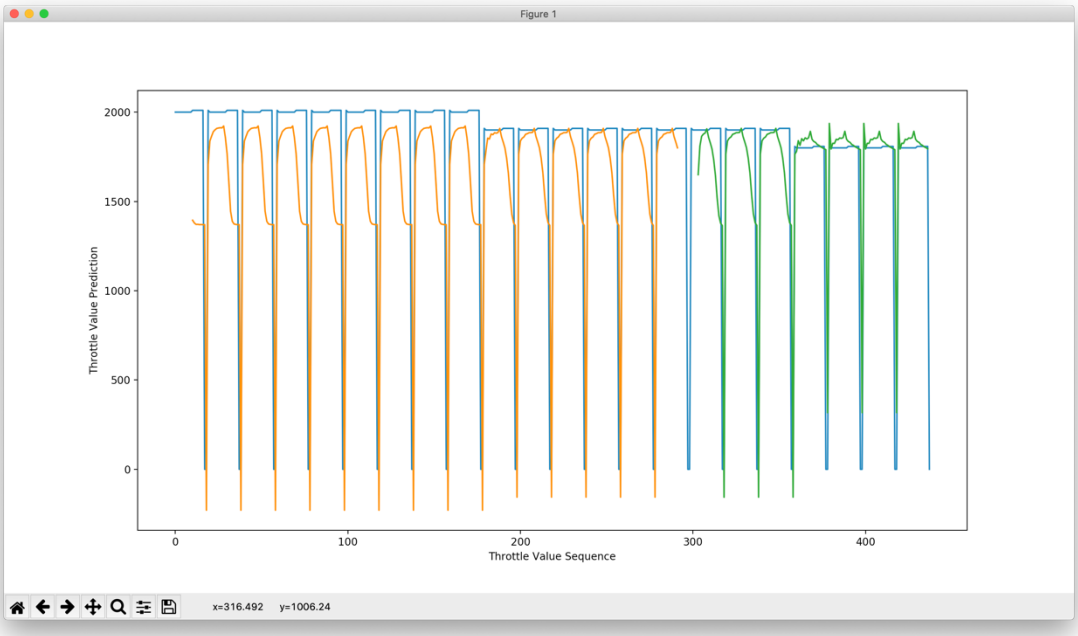


Fig 5.18 LSTM performance on real flight data input. Orange denotes training data, green denotes test data, blue denotes LSTM prediction.

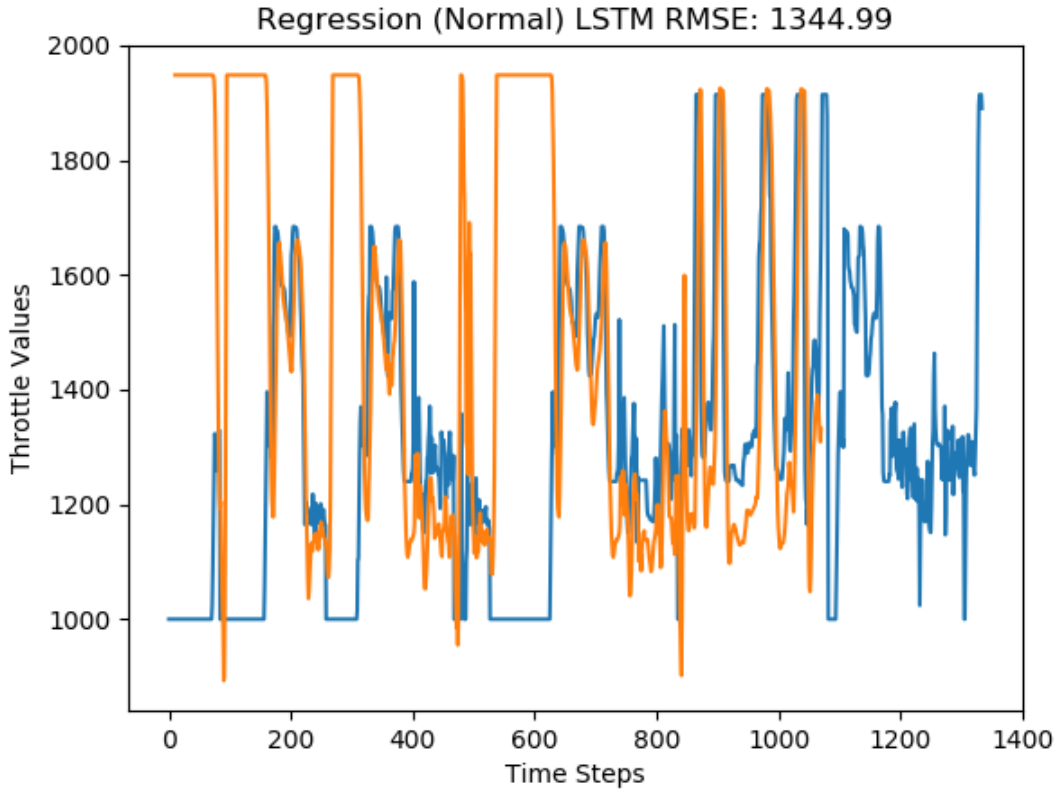


Fig 5.19 LSTM performance on synthesized data. Blue denotes test data; Orange denotes LSTM prediction.

The standard scalar initially used for the LSTM model was not suitable, producing outputs that had a drastic negative offset.

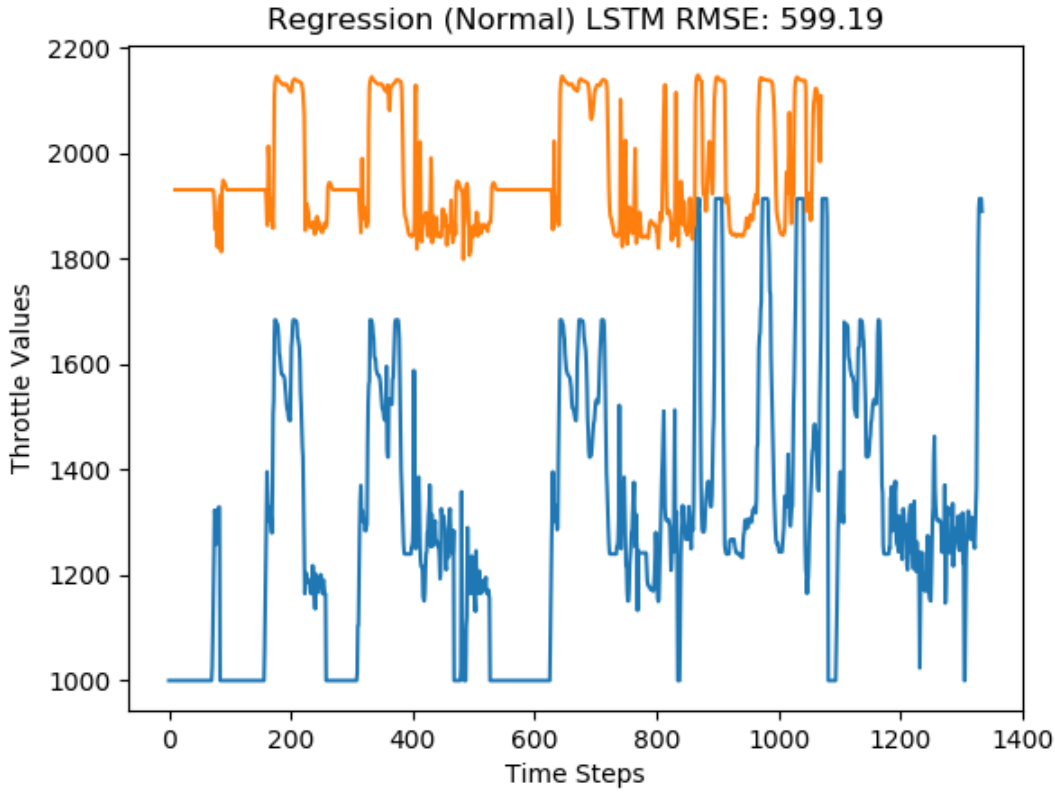


Fig 5.20 LSTM performance with standard scalar on input sequence

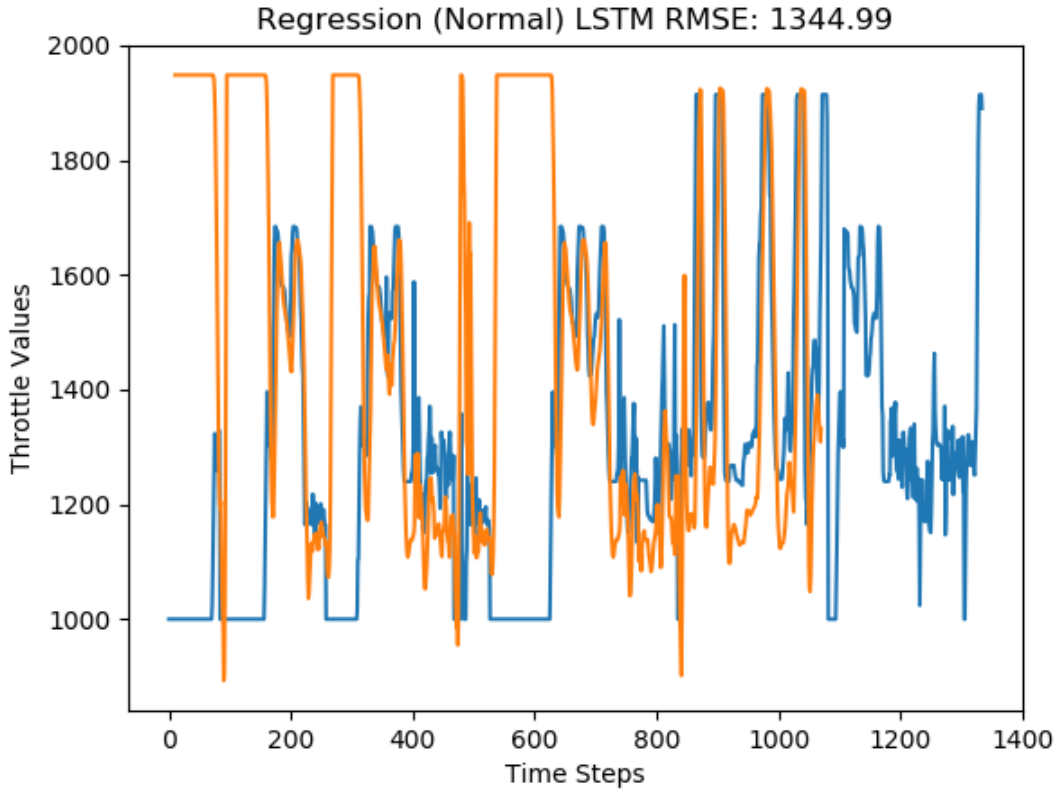


Fig 5.21 A direct scalar based on the throttle encoding function was written, providing an accurate output scale.

The design of this machine learning pipeline has an inherent dimensionality reduction effect built in; the single model to be deployed for real-time throttle prediction from real-time telemetry stream, only has to consider the throttle sequence from 2 seconds in the past to the present, in order to predict the optimal throttle values for 2 seconds into the future. The other telemetry fields, such as IMU, powertrain and GPS data need not be considered, as the LSTM network can infer the kinematic behaviour of the mUAV and the correct throttle response required. This will speed up prediction time, as less computation is required.

Limitations:

As URLC was already under development while the new drone was not yet fully operational, test flights were performed with a sufficiently fast mUAV that could easily achieve speeds of just under 300kph. The old and new mUAV designs use very similar electronics, with the same UAV operating system, Betaflight, while sharing similar flight characteristics. This ensures the machine learning pipeline is cross-compatible. To update the URLC software for use with the new mUAV, it just needs to be retrained with new flight logs from the new mUAV design, without modification to the current machine learning software.

With the training pipeline complete, the URLC was deployed as a partially cloud-based Web Application, at http://renxiang.cloud.

The application contains the following modules, networked using q IPC:

- Kdb+ server storing historical and real-time mUAV data
- Machine Learning models update module, for batch retraining
- Throttle prediction module for real-time telemetry stream
- Real-time telemetry streaming module, that receives Mavlink telemetry data sent by the mUAV to the ground station and publishes it to the database
- Control Uplink module, that outputs the throttle output predictions from the kdb+ server to the mUAV
- Flight log playback module that publishes a simulated real-time telemetry stream, for testing the LSTM throttle prediction model
- Connection to data analytics dashboards via q IPC and ODBC (Kx Developer, Tableau, TIBCO Spotfire Analyst)

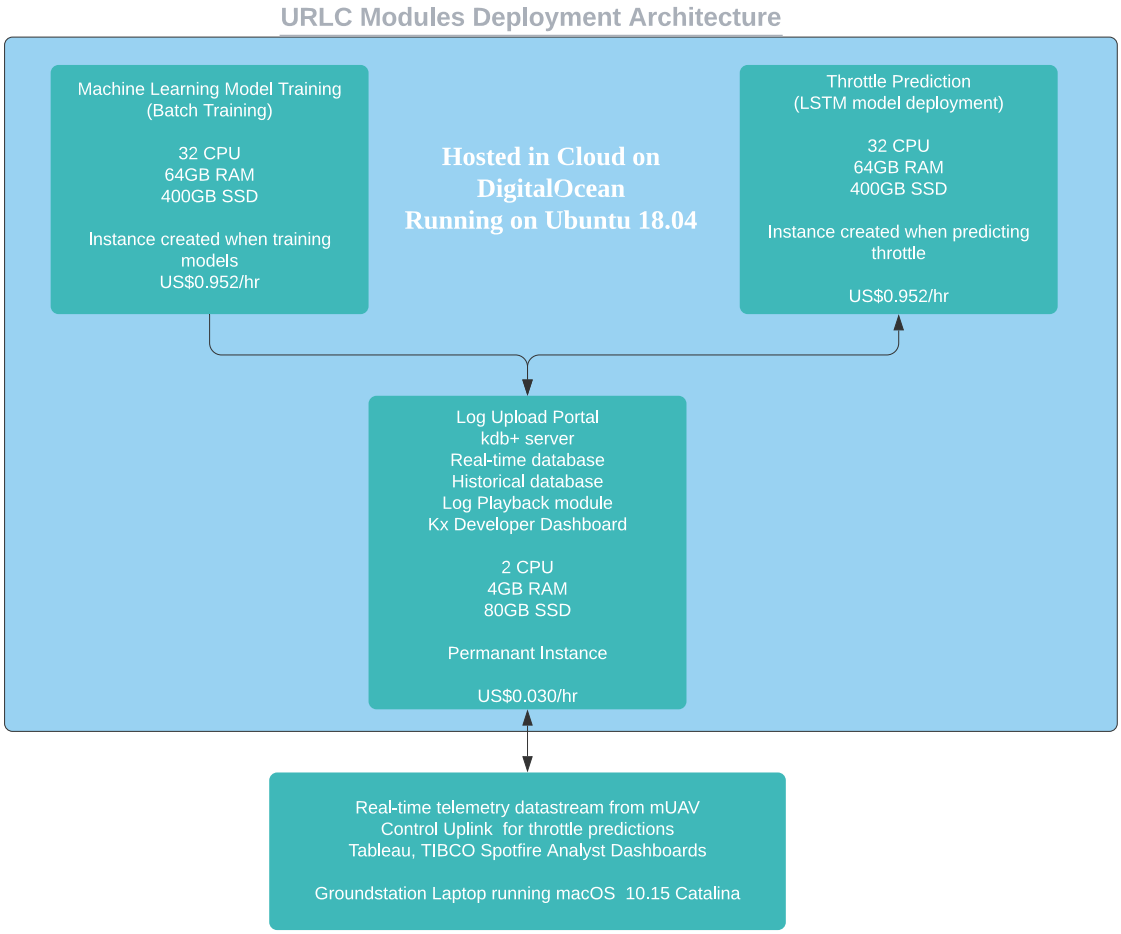


Fig 5.21 Topology of networked machines running URLC as a Web Application

DigitalOcean's Infrastructure-As-A-Service (IAAS) was used. Virtual machines running the Ubuntu 18.04 server images were deployed for all cloud-based modules, with varying machine costs and computational resources.

An inexpensive instance that has with sufficient computational power, served as an always-up, permanent instance for web hosting of the upload portal, kdb+ database and Kx Developer dashboard. It also stores the trained LSTM model, uploaded to it via SFTP from the training module. The kdb+ on-demand 64-bit academic license which supports the EmbedPy Python interpreter integration is used, if Python scripts and Kx Developer are to

be run from within kdb+. Otherwise, the non-commercial, 32-bit personal license with a 4GB memory limitation may be used.

Due to the high computational power required for training and running the machine learning models, they are each deployed to separate, high-performance virtual machines that are created on-demand when required during mUAV operation. Operating these servers on-demand is required due to high operation costs of the fast virtual machines. To minimise deployment time and configuration repeatability of these on-demand instances, a server was configured with the Git repository of the machine learning code, with trained models stored on the kdb+ server permanent instance and saved as a customised server image. This image would be deployed for all future on-demand instances of the application, with a Git pull performed to update the archived code repository to the latest deployable version.

The ground station serves as the hardware interface for the URLC autopilot system backend with the mUAV aircraft. It uses a Mavlink USB Serial radio to receive the real-time telemetry data from the mUAV, that interfaces through a local Python script with q IPC connection to the kdb+ cloud server, to publish the received data with low latency. Simultaneously, a separate local Python script creates another q IPC connection to the kdb+ server to retrieve the latest throttle predictions. The Python script has a USB Serial connection that encodes the throttle value into a Pulse Position Modulation (PPM) signal generator, that is then transmitted to the mUAV via a co-pilot interface on the mUAV remote controller. An ODBC database connection from kdb+ to Tableau and Spotfire Analyst allows real-time telemetry data to be directly visualised as dashboards for the navigator to utilise.

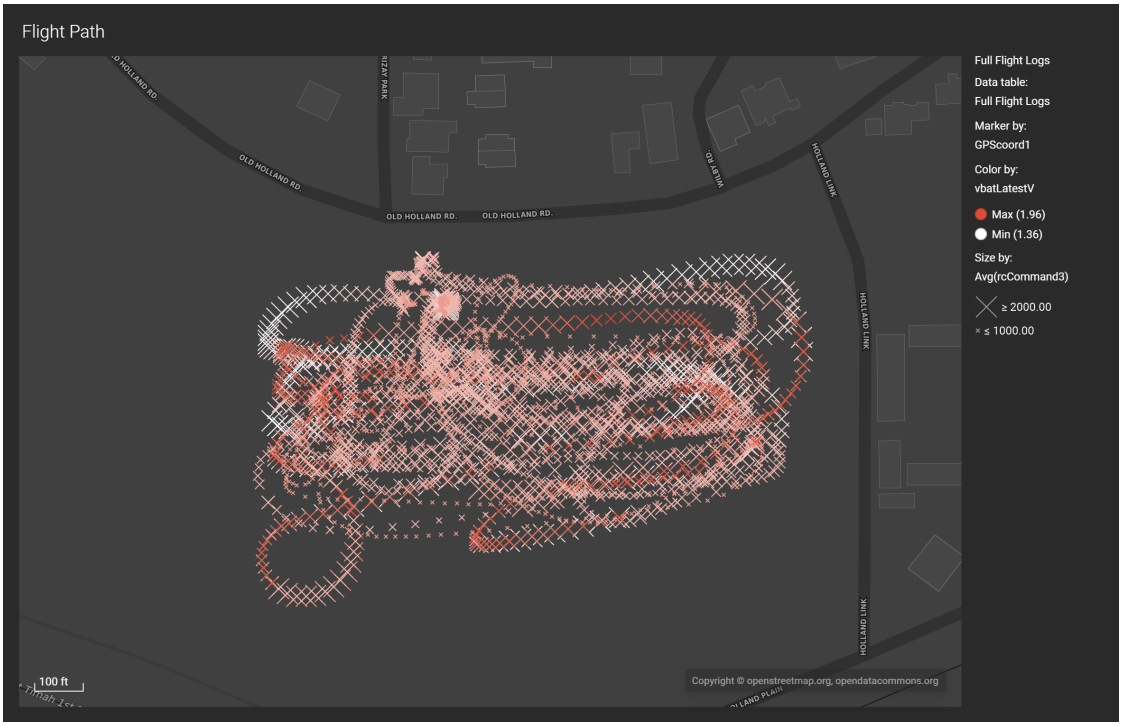


Fig 5.22 Flight path with pilot input and LiPo voltage in Spotfire Analyst

Data Relationships (Linear Regression)

Y (numerical)	X (numerical)	p-value	FStat	RSq	R	Df	n
vbatLatestV	rcCommand3	2.06E-111	545.03	0.15	-0.38	3186	3188
vbatLatestV	motor2	4.93E-105	510.85	0.14	-0.37	3186	3188
GPSspeedms	timens	7.83E-075	353.30	0.10	-0.32	3186	3188
vbatLatestV	accSmooth0	1.22E-054	251.83	0.07	0.27	3186	3188
vbatLatestV	rcCommand1	4.70E-040	180.43	0.05	0.23	3186	3188
GPSspeedms	GPScoord0	3.22E-037	166.74	0.05	-0.22	3186	3188
vbatLatestV	gyroADC1	1.40E-036	163.66	0.05	0.22	3186	3188
GPSspeedms	GPScoord1	4.05E-036	161.45	0.05	0.22	3186	3188
vbatLatestV	axisP0	2.29E-035	157.83	0.05	-0.22	3186	3188
vbatLatestV	axisD0	2.74E-035	157.46	0.05	-0.22	3186	3188
vbatLatestV	axisP2	4.07E-035	156.63	0.05	0.22	3186	3188
vbatLatestV	axisP1	4.32E-035	156.51	0.05	-0.22	3186	3188
GPSspeedms	motor2	2.21E-032	143.53	0.04	0.21	3186	3188
vbatLatestV	gyroADC0	5.27E-032	141.73	0.04	0.21	3186	3188
vbatLatestV	timens	9.70E-032	140.46	0.04	-0.21	3186	3188
vbatLatestV	gyroADC2	2.36E-029	129.09	0.04	-0.20	3186	3188

Data table:
Data Relationships

Fig 5.23 Regression model of data from telemetry in Spotfire Analyst

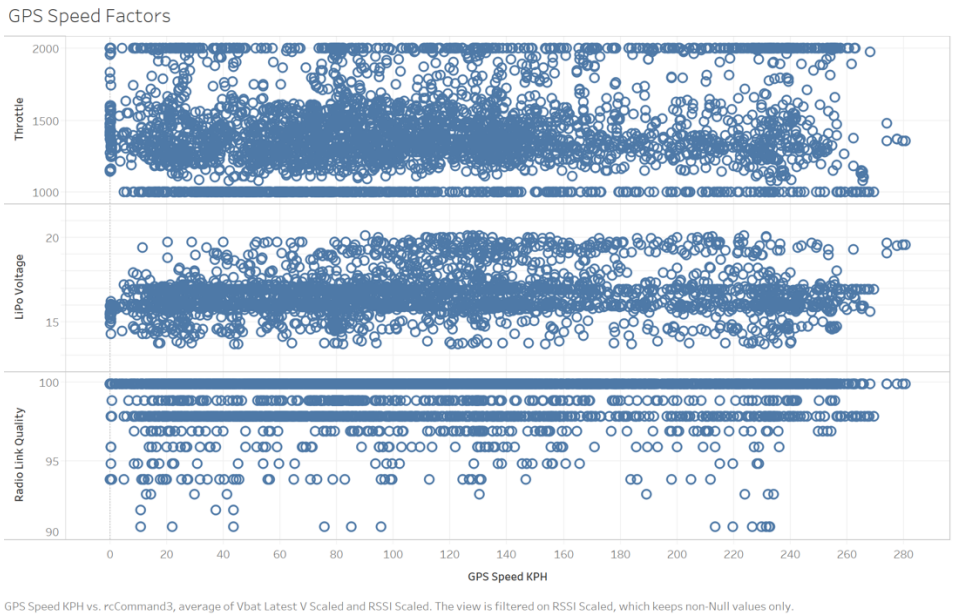


Fig 5.24 Real-time data distribution mapped to GPS speed ranges in Tableau

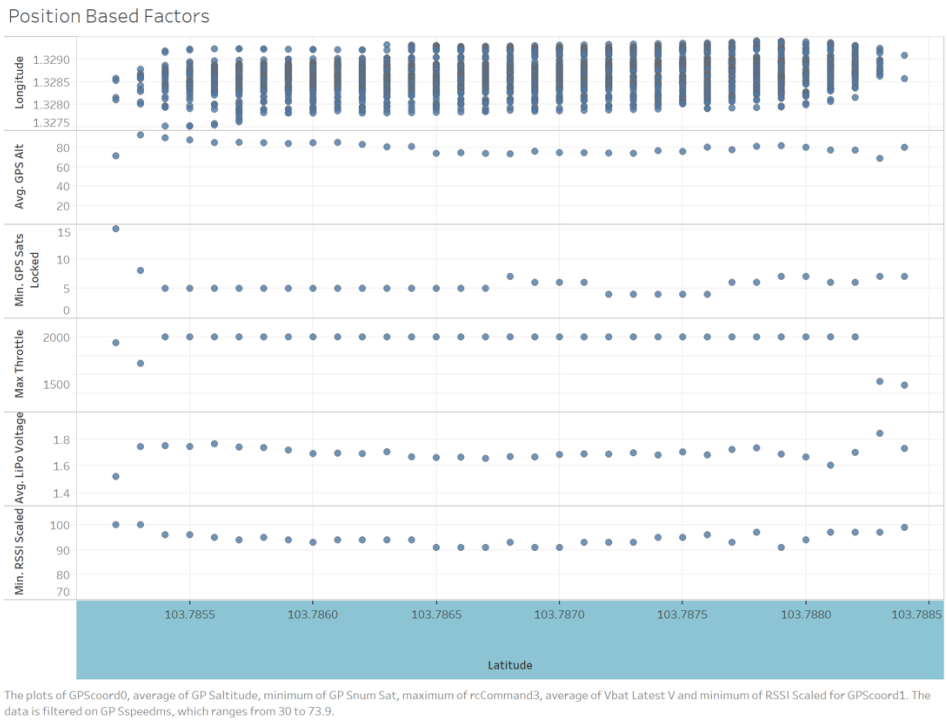


Fig 5.25 Real-time data distribution mapped to GPS Latitude in Tableau

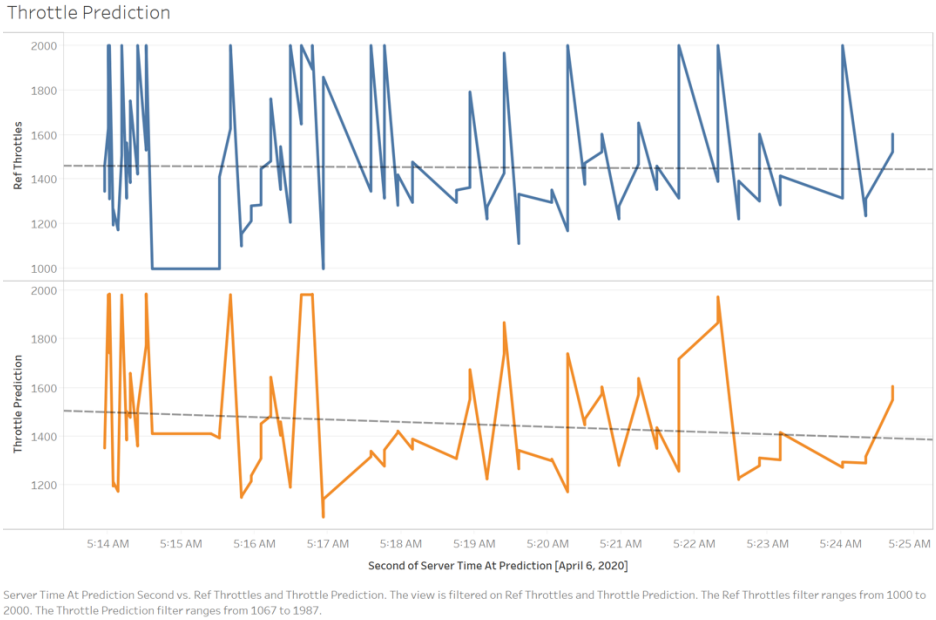


Fig 5.26 Real-time URLC Throttle Prediction Performance evaluation in Tableau

The overall URLC application topology is captured by the following flowchart:

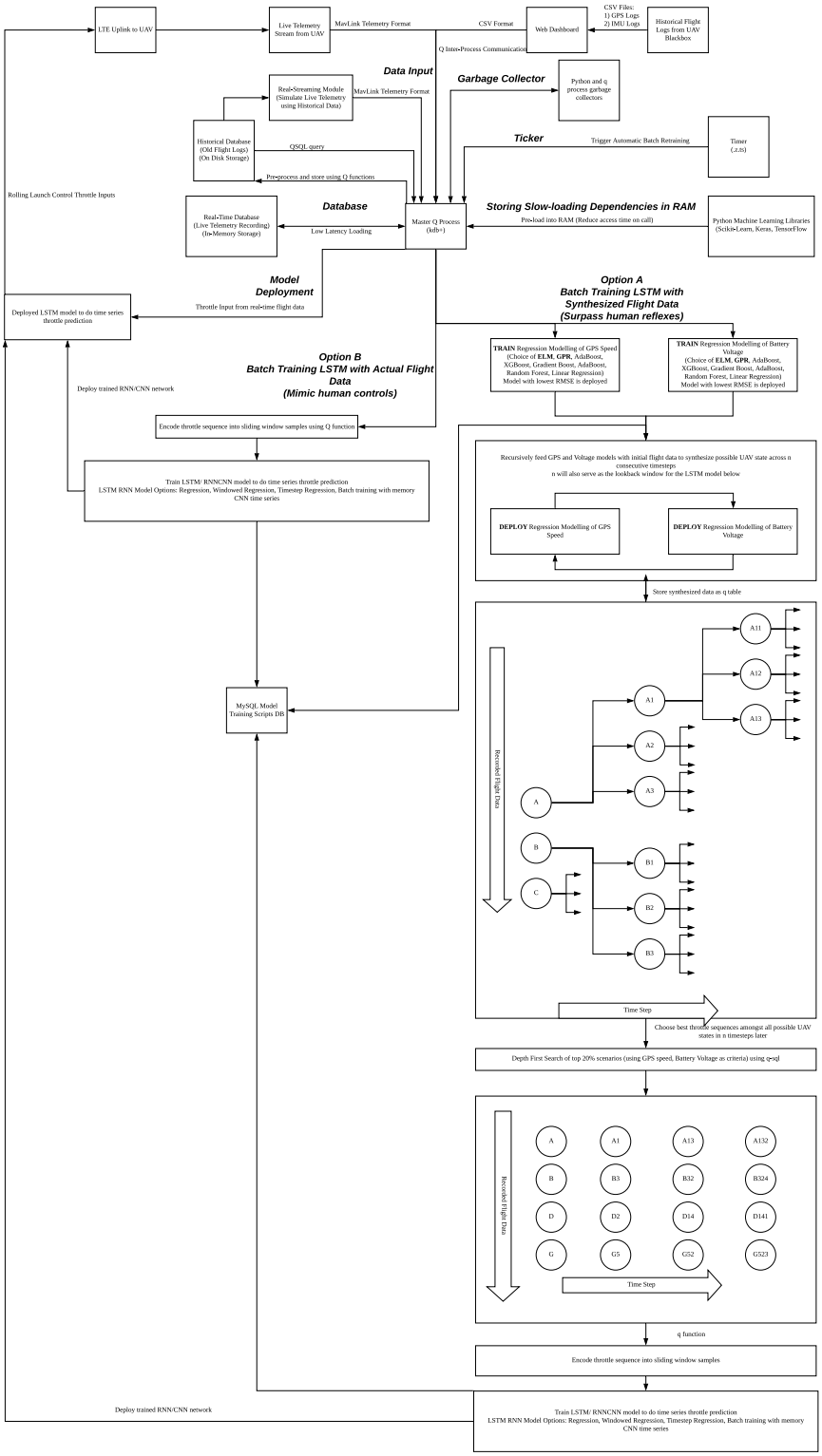


Fig 5.27 URLC Application Topology

Chapter 6

Prototype mUAV Field Testing

6.1 Test Findings

Fig 6.1 Test Prototype mUAV at Old Holland Road Flying Field

Fig 6.2 mUAV on launchpad. Zhong Bingchen (Navigator/Spotter) logging flight conditions

Fig 6.3 Right: Foo Ren Xiang (FPV Pilot) and Pan Zexu (LOS Pilot) preparing for landing

Firmware

In general, the firmware and configuration used on the previous test mUAV design could be ported to the new mUAV design, with some key changes required.

The frame structure of the mUAV is distinctly different from the usual racing drone configuration. This required some key changes when configuring the Betaflight firmware. PID control values had to be reduced, as the efficient aerodynamics and centered mass distribution of the UAV require less correction from the PID controller to control its angular momentum.

In addition, the low pass filter for the gyroscope had to be reduced from the default 200Hz to 50Hz. This was due to some high amplitude; low frequency vibrations being transmitted from the arms. The vibrations were caused by a combination of two unwanted factors: the motors with larger stators to provide higher power, were heavier. The low drag arms had carbon plates that were aligned perpendicular to the vibration axis of the heavy motors, resulting in significant flex when the motors had angular acceleration, from changes in motor RPM/throttle.

Before the change to the low pass filter was discovered, the issue manifested as catastrophic failure of the powertrain. A plausible explanation was due to vibrations recorded by the gyroscope data going past the low pass filter, the PID controller was actively trying to correct for angular rate errors from the vibration, by increasing the correction output. This was amplified by the P gain, causing the motor RPM to increase more per correction. This manifested as an increase in angular acceleration error, that was corrected by the D gain of the PID controller, which caused the motor RPM corrections to be performed faster, increasing the vibration frequency. This caused an increasing, oscillating feedback loop to the PID controller. As a result, small errors in angular rate

resulted in a strong oscillation of motor RPM. A high DC component of the correction signal caused the low resistance motor coils to quickly heat up and vapourise the enamel wire insulation, shorting out the ESC terminals. This resulted in the ESC catching fire, requiring the ESC and all motors to be changed. The solution was found by performing a thorough check on the electronics wiring and PID configuration. The filter configuration was a usual cause of unexpected errors, and adjustments were made to it until the PID controller remained stable while the mUAV was shaken with forces of magnitude expected during flight. A current-limiting resettable PTC fuse was placed between the LiPo and the rest of the powertrain to avoid damage during diagnostics.

Subsequently, the gyroscope low pass filter PID controller settings were finetuned by performing hovers and hard continuous accelerations in line of sight test flights. Due to flex still present in the arms, a perfectly stable tune was not achieved yet for top speed flight, but was stable enough for moderate speed flight.

Test Flight/Unplanned Crash Structure Testing

A proper take-off and test flight using FPV was conducted. The untuned mUAV reached a top speed of ~170km/h with 62% throttle input, with a maximum acceleration of 3.5G. Higher throttle speeds could not be achieved due to PID controller oscillations.

The FPV camera temporarily failed during flight, due to electronics overheating from insufficient airflow. The ESC reached 120C and the flight controller reached 79C. This caused the camera and video transmitter to enter thermal shut down for a few seconds during a speed run at 178km/h. With safety as priority, the field was confirmed to be clear of people and the split second decision to immediately disarm/stop the mUAV was made.

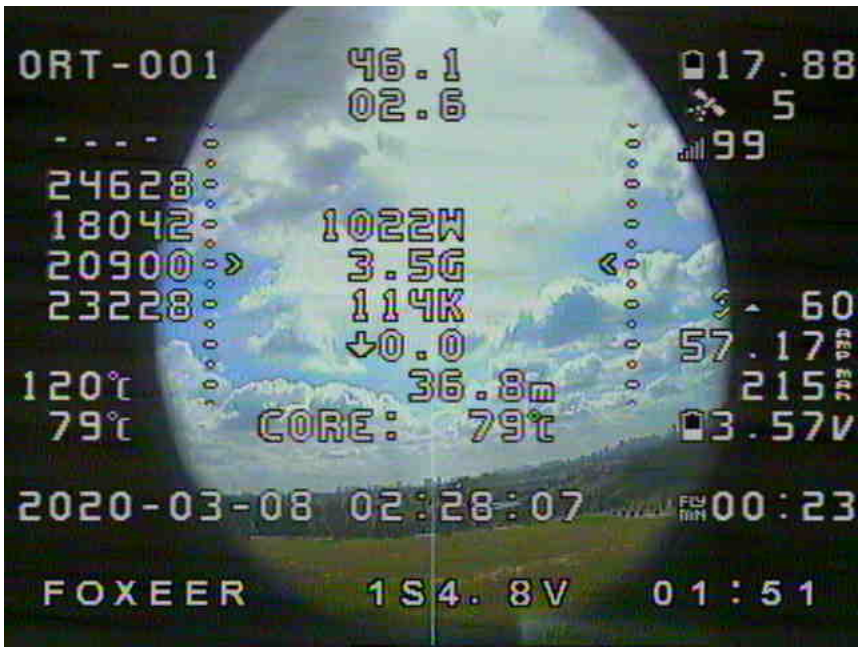


Fig 6.4 Max acceleration of 3.5G

Fig 6.5 Max speed of 178km/h

A max deceleration of 14.1G due to aerodynamic drag of the mUAV entering a spin was recorded, with its long and wide fuselage acting as an airbrake. The angle and position of mUAV could be determined from GPS and IMU data displayed on OSD despite a missing

camera view. This inevitably became a crash test of the frame structure, designed for such a scenario.

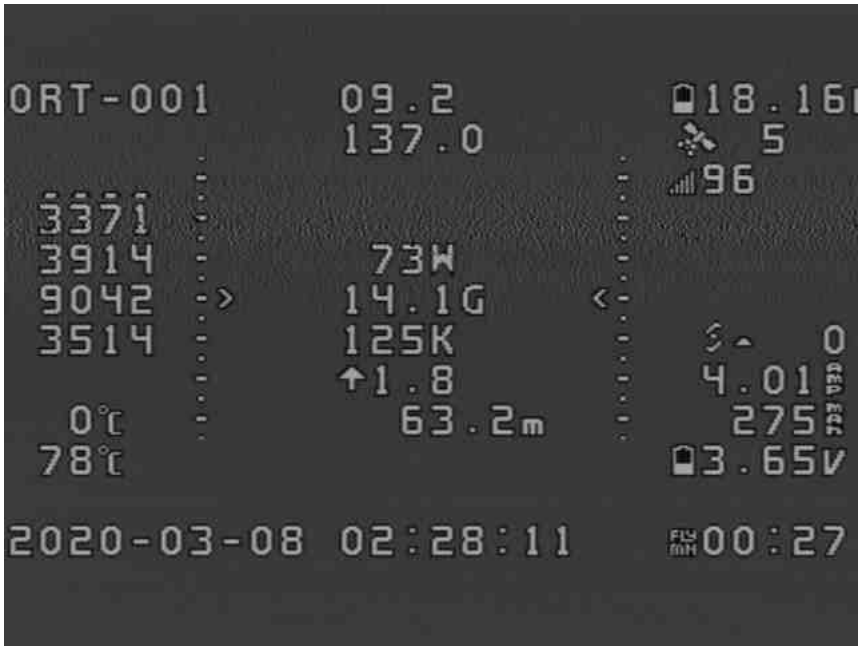


Fig 6.6 Max deceleration of 14.1G from aerodynamic drag

The following minor damage was observed:

- The motors were detached from the mounts from the impact of the crash, but the wiring remained intact. The aerodynamic motor fairings were shattered, as part of the motor mount
- The bodywork around the arms was shattered from the arms expanding. A cracked formed on the underside of the nose cone from impact
- The LiPo carrier section of the frame became detached, as the sacrificial nylon bolts holding it snapped. The sacrificial camera signal cable was also torn before at its strain relief knots. The LiPo battery was fully enclosed by the LiPo carrier, which acted as a roll cage and absorbed most of the impact, preventing it from being smashed, which could have caused a fire. However, the LiPo was still slightly warped due to its own mass.

- The tail bodywork and electronics within in the tail of the drone was unscathed despite impact with the ground, evident from mud stains on the bodywork. The right tail strut absorbed the impact and fractured.
- In general, the sacrificial components of the frame, designed to be cheap and easy to replace, took the impact as designed and protected the expensive components during the crash

Fig 6.7 mUAV after crash

Fig 6.8 Crack at bottom of nose cone

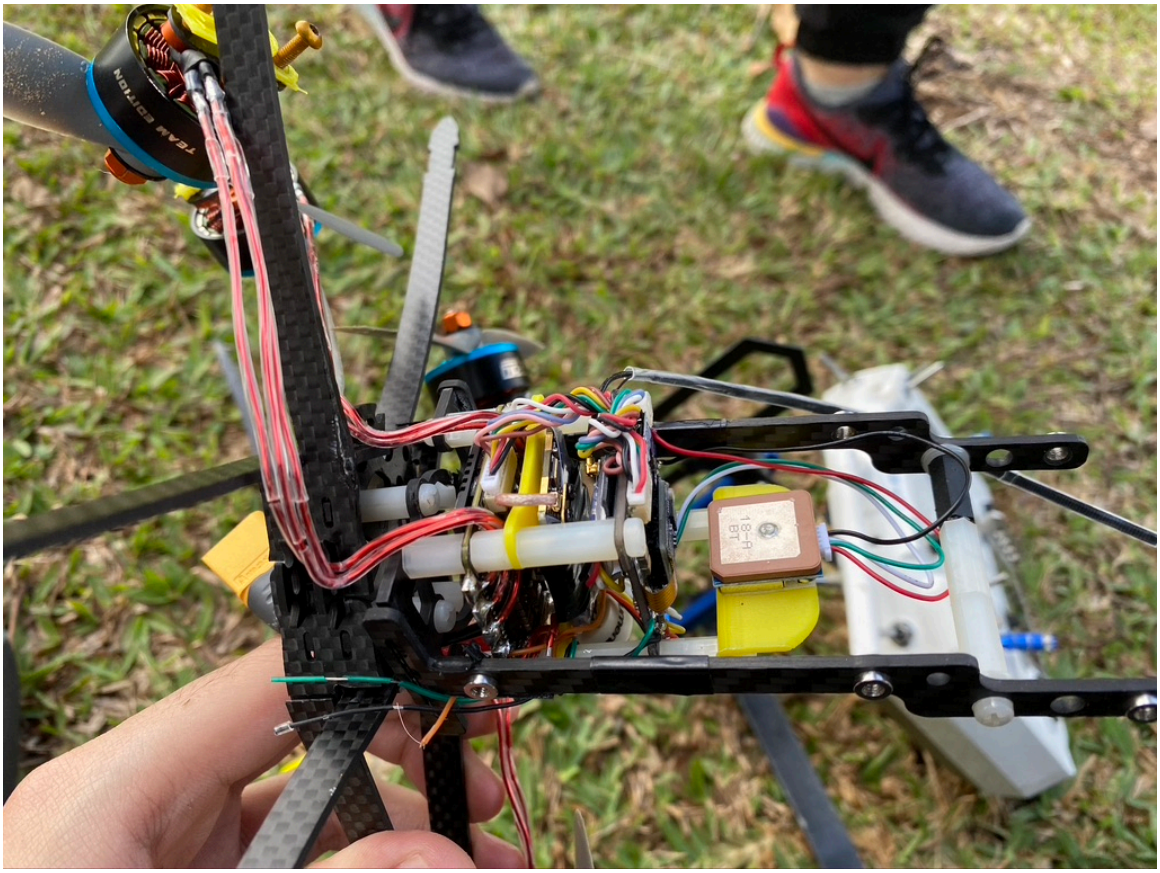


Fig 6.9 Tail electronics unscathed. Right tail strut fractured.

Fig 6.10 No cracks on tail bodywork despite impact with ground

6.2 Design Improvements

Additional bracing structure and a stronger motor mount clamp can be installed to increase stiffness and rigidity of the arms. This should allow the heavy and powerful motors to be used while reducing low frequency oscillations being transmitted to the IMU. Additional damping, acting as a mechanical low pass filter, can also be added to the IMU.

With the crash being caused by electronics overheating, the film placed over the nose viewport for the FPV can be removed, allowing it to act as an intake for air cooling of the electronics in the tail. The gaps present at the arm mounting point will serve as the outlet for the air with minimal disruption to the aerodynamics of the mUAV.

A small, dedicated buck converter can be added for powering the FPV camera. This should reduce the load of the buck converter currently used, which is powering both the video transmitter and FPV camera, resulting in lower heat build-up.

Conclusions and Future Work

7.1 Conclusions

The conceptualisation and development of the prototype mUAV hardware and URLC application resulted in promising results in simulations. This indicates the concept is workable with good reason for further development. Due to the depth of development required, this project concludes with the initial prototype aircraft and an alpha testing version of URLC deployed.

With some further work, a working mUAV system optimised for sustained top speed is realisable.

7.2 Recommendation in Future Work

Aircraft and hardware Design:

- A cloud-based CFD solver that is properly tuned for results with good real-world correlation will result in more precise designs
- A redesign of the frame's arms with increased stiffness will result in a more stable flight behaviour
- Improved electronics cooling solutions with minimal aerodynamic impact should be investigate
- Determining a stable and precise PID controller tune

URLC:

- Optimise memory usage of machine learning training pipeline
- Update application to support multiple models trained for specific multiple weather conditions that can be selectively deployed
- Implement k-fold cross validation testing in regression model training to further improve accuracy
- Implement transfer learning with LSTM to reduce training time during batch retraining, for lower latency model updates
- Implement usage of plaidML backend for Keras, to train LSTM using OpenGL that supports integrated graphics and AMD GPU, instead of just Nvidia CUDA GPUs.
- Migrate URLC to Docker Compose for greater application mobility

Reflection on Learning Outcome Attainment

Engineering knowledge

Problem Analysis/ Investigation

- The problem to be solved by the project is relatively novel and required in depth analysis and investigation. This led to a suitable and new model to be created as its solution

Design/development of Solutions

- A hardware and software solution was planned, designed and implemented from scratch. In the process, a lot of new technical skills were picked up.

Modern Tool Usage

- Mechanical CAD software was heavily used for design and analysis in aerodynamic/mechanical Simulation.
- Gained familiarity in creating a machine learning pipeline based on a real-time time-series database
- Picked up and became proficient with Python and q/kdb+ programming languages

Individual and Teamwork

- *Individual work: Learned to model the problem to be investigated, gain an intuition on possible solutions and conduct research to develop the solution*
- *Teamwork: Worked with friends who are fellow mUAV hobbyists to conduct development, parts procurement, field testing and debugging*

Communication

- *Consulted with FYP supervisor on problem formulation and requirements capture*
- *Discussed various mechanical and aerodynamic design concepts with friends to finetune the design*

- Had meetings with manufacturers to discuss fabrication limitations of the CNC and SLS 3D printing machines in order to modify design for manufacture
- Attended soft computing conference to present problem discovered and preliminary novel approach to solving the problem. Discussed ideas with peers to gain more insight on problem formulation and alternate perspectives.
- Had frequent discussions with FYP supervisor during modelling and formulation of machine learning solution and pipeline to identify flaws and areas of improvement in the concept

Project Management and Finance

- Manufacturing, purchase of components, mUAV assembly and field testing required careful budget and purchase/production lead time planning, manpower coordination
- Due to the indeterminate nature of machine learning development, work on the machine learning pipeline was started early and development priorities were constantly evaluated to ensure

References

[1] F1, “The long read: Chasing 400km/h in the world's fastest F1 car,” The official home of Formula 1®, 23-Jul-2018. [Online]. Available: https://www.formula1.com/en/latest/article.the-long-read-chasing-400km-h-in-the-worlds-fastest-f1-car.4KJFHbfjaU8GgqUEgsKQeU.html. [Accessed: 23-Sep-2019].

[2] K. Stephenson, “The Drone Racing League sets quadcopter speed record,” Guinness World Records, 20-Jul-2017. [Online]. Available: https://www.guinnessworldrecords.com/news/2017/7/the-drone-racing-league-builds-the-worlds-fastest-racing-drone-482701. [Accessed: 23-Sep-2019].

[3] KIDERA, K., Yoichiro, Y. and ITO, T. (2009). Development of Traction Control Systems for Formula One. Honda R&D Technical Review F1 Special (The third Era Activities), 1(1), pp.190-200.

[4] OGAWA, A., YANO, S., MASHIO, S., TAKIGUCHI, T., NAKAMURA, S. and SHINGAI, M. (2009). Development Methodologies for Formula One Aerodynamics. Honda R&D Technical Review F1 Special (The third Era Activities), 1(1), pp.142-151.

[5] SHIBUE, H., SUZUKI, T., URAKI, M., IDA, A., TSUBOUCHI, A. and KITAKI, Y. (2009). Measurement and Analysis Techniques of Formula One Chassis Development. Honda R&D Technical Review F1 Special (The third Era Activities), 1(1), pp. 179-187.’

[6] “Fastest ground speed by a battery-powered remote-controlled (RC) quadcopter.” [Online]. Available: https://www.guinnessworldrecords.com/world-records/464800-fastest-ground-speed-by-a-battery-powered-remote-controlled-quadcopter. [Accessed: 01- Jun- 2019].

[7] J. Cook, "FPV Multirotor Speed Record - In Pursuit", GetFPV Learn. [Online]. Available: https://www.getfpv.com/learn/fpv-news/fpv-multirotor-speed-record/. [Accessed: 01- Jun- 2019].

[8] "THE DRL RACERX", Drone Racing League. [Online]. Available: https://thedroneracingleague.com/racerx/. [Accessed: 01- Jun- 2019].

[9] N. Chandran, "FAA's new drone laws go into effect Monday, allowing US companies to innovate", CNBC, 2020. [Online]. Available: https://www.cnbc.com/2016/08/29/faas-new-drone-laws-go-into-effect-monday-allowing-us-companies-to-innovate.html. [Accessed: 01- Jun- 2019].

[10]"Vodafone XBlades Racing Drone Sets a New Drone Speed World Record at Goodwood Festival of Speed", GlobalMediaCentre.com. [Online]. Available: https://www.globalmediacentre.com/news/vodafone-xblades-racing-drone-sets-new-drone-speed-world-record-goodwood-festival-speed. [Accessed: 01- Jun- 2019].

[11]"Flow Separation - an overview | ScienceDirect Topics", Sciencedirect.com. [Online]. Available: https://www.sciencedirect.com/topics/engineering/flow-separation. [Accessed: 01- Jun- 2019].

[12]"The Drag Equation", Grc.nasa.gov. [Online]. Available: https://www.grc.nasa.gov/www/k-12/airplane/drageq.html. [Accessed: 01- Jun- 2019].

[13]"Aerospaceweb.org | Ask Us - Streamlining and Aerodynamics", Aerospaceweb.org. [Online]. Available: http://www.aerospaceweb.org/question/aerodynamics/q0094b.shtml. [Accessed: 01- Jun- 2019].

[14] M. Kumar and D. Anoop , “Effect of Vortex generators on Aerodynamics of a Car: CFD Analysis,” International journal of innovations in engineering and technology, 2013.

[15]"What is a Car Spoiler and What Does It Do?", Medium, 2018. [Online]. Available: https://medium.com/@hardingsteel/what-is-a-car-spoiler-and-what-does-it-do-8fd1ae989d9c. [Accessed: 01- Jun- 2019].

[16]W. Dong, G. Gu, X. Zhu and H. Ding, "Modeling and Control of a Quadrotor UAV with Aerodynamic Concepts", World Academy of Science, Engineering and Technology, International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering, vol. 7, pp. 901-906, 2013.

[17] T. Puls, M. Kemper, R. Küke and A. Hein, "GPS-based position control and waypoint navigation system for quadrocopters," 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, 2009, pp. 3374-3379, doi: 10.1109/IROS.2009.5354646.

[18] S. Koen, "Architecting a Machine Learning Pipeline", Medium, 2019. [Online]. Available: https://towardsdatascience.com/architecting-a-machine-learning-pipeline-a847f094d1c7. [Accessed: 01- Jun- 2019].

[19] J. Brownlee, "Automate Machine Learning Workflows with Pipelines in Python and scikit-learn", Machine Learning Mastery, 2016. [Online]. Available: https://machinelearningmastery.com/automate-machine-learning-workflows-pipelines-python-scikit-learn/. [Accessed: 01- Jun- 2019].

[20] M. Rajaratne, "Data Pre Processing Techniques You Should Know", Medium, 2018. [Online]. Available: https://towardsdatascience.com/data-pre-processing-techniques-you-should-know-8954662716d6. [Accessed: 01- Jun- 2019].

[21]P. Patil, "What is Exploratory Data Analysis?", Medium, 2018. [Online]. Available: https://towardsdatascience.com/exploratory-data-analysis-8fc1cb20fd15. [Accessed: 01- Jun- 2019].

[22]L. Smith, A tutorial on Principal Components Analysis. 2002.

[23] "PCA Is Not Feature Selection", Medium, 2020. [Online]. Available: https://towardsdatascience.com/pca-is-not-feature-selection-3344fb764ae6. [Accessed: 01- Jan- 2020].

[24]E. Rençberoğlu, "Fundamental Techniques of Feature Engineering for Machine Learning", Medium, 2019. [Online]. Available: https://towardsdatascience.com/feature-engineering-for-machine-learning-3a5e293a5114. [Accessed: 01- Jun- 2019].

[25] P. Ippolito, "Feature Engineering Techniques", Medium, 2019. [Online]. Available: https://towardsdatascience.com/feature-engineering-techniques-9a57e4901545. [Accessed: 01- Jan- 2020].

[26]"Deep Feature Synthesis: How Automated Feature Engineering Works", Alteryx Innovation Labs, 2018. [Online]. Available: https://innovation.alteryx.com/deep-feature-synthesis/. [Accessed: 01- Jun- 2019].

[27]N. Chauhan, "A beginner's guide to Linear Regression in Python with Scikit-Learn", Medium, 2019. [Online]. Available: https://towardsdatascience.com/a-beginners-guide-to-linear-regression-in-python-with-scikit-learn-83a8f7ae2b4f. [Accessed: 01- Jun- 2019].

[28] I. Bhattacharyya, "Support Vector Regression Or SVR", Medium, 2018. [Online]. Available: https://medium.com/coinmonks/support-vector-regression-or-svr-8eb3acf6d0ff. [Accessed: 01- Jun- 2019].

[29]J. McMinis, "Intuition for Support Vector Regression and the Gaussian Process", Mcminis1.github.io, 2014. [Online]. Available: http://mcminis1.github.io/blog/2014/05/10/intuition-for-SVR/. [Accessed: 01- Jun- 2019].

[30]"Quick Start to Gaussian Process Regression", Medium, 2019. [Online]. Available: https://towardsdatascience.com/quick-start-to-gaussian-process-regression-36d838810319. [Accessed: 01- Jun- 2019].

[31] G.-B. Huang, Q.-Y. Zhu and C.-K. Siew, "Extreme Learning Machine: Theory and Applications", Neurocomputing, vol. 70, pp. 489-501, 2006

[32]J. Brownlee, "Crash Course in Recurrent Neural Networks for Deep Learning", Machine Learning Mastery, 2016. [Online]. Available: https://machinelearningmastery.com/crash-course-recurrent-neural-networks-deep-learning/. [Accessed: 01- Jan- 2020].

[33]J. Brownlee, "Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras", Machine Learning Mastery, 2016. [Online]. Available: https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/. [Accessed: 01- Jan- 2020].

[34]"The Drag Equation", Grc.nasa.gov, 2020. [Online]. Available: https://www.grc.nasa.gov/www/k-12/airplane/drageq.html. [Accessed: 01- Jun- 2019].

[35]2019. [Online]. Available: https://www.caas.gov.sg/public-passengers/unmanned-aircraft/ua-safety-guidelines.

[36]2019. [Online]. Available: https://www.caas.gov.sg/public-passengers/unmanned-aircraft/ua-regulations.

[37]R. Ryan Lademann, "GPS Speed Run Analyzer", Quadstar Drones, 2019. [Online]. Available: https://quadstardrones.com/2018/03/11/gps-speed-run-analyzer/.

Appendix

Readme of URLC GitHub Repository

High Speed UAV Rolling Launch Control and Flight Analysis System (FAS)

Using data analytics and machine learning to optimise UAV straight line speed

The high speed UAV this application interfaces with. This aerodynamic machine was designed with CFD and FEA techniques. Monocoque is manufactured with

carbon composites and bodywork is 3D-printed with SLS technology. Coupled with a 2.3kW powertrain to achieve >300kph speeds

Live machine learning prediction for high speed drone control using streaming flight telemetry

Using: Kx kdb+ in-memory time-series database for data processing, streaming and storage

Scikit-learn (Gaussian Process Regression, Linear Regression), ensemble learning (Adaboost, XGBoost) and Extreme Learning Machines regression models to model flight physics

Keras/tensorflow LSTM model for time-series throttle prediction, trained by synthesized data from regression models

Kx developer, Tableau and Tibco Spotfire dashboards are connected to kdb for additional data filtering, visualisation and analysis. Connectivity through q WebSocket and ODBC.

Application hosted in cloud on #DigitalOcean for high availability. CPU intensive ML training is executed on a temporary, high-spec instance during batch training. Database and model deployment are deployed on a separate cost-effective instance. Module communications implemented using q IPC.

Live Prediction on Streaming Telemetry (Video Demo): https://youtu.be/7xETmWG2jyI

Tableau Connected to cloud kdb server (Video Demo): https://youtu.be/-vmvTTZp8GE

Application written in Python and q/kdb. Test version hosted at http://renxiang.cloud/OHR400Dashboard

Rolling Launch Control Components:

kdb+ Server (FASServer32bitInit.q or FASServer64bitInit.q)

q server process that acts as the database (RDB and HDB). Historical and real-time flight telemetry and Rolling Launch Control data is streamed and retrievable from this process. Hosted at renxiang.cloud.
Interfaces using q IPC, ODBC and websockets. (Requires authentication).
Kx Developer is active if 64bit server is in use.
Requires 32bit or 64bit KDB (requires cloud permission)
Tableau (live data refresh) and TIBCO Spotfire Analyst (manual data refresh) have been successfully connected to this server instance using ODBC.
See screenshots in Git Repository for useful dashboard views.

- To run:

  ```
  #64 bit version
  ./FASServer.sh
  ```

  Alternatively

  ```
  #64 bit version
  rm nohupServer.out; nohup q FASServer64bitInit.q -U credentials.txt>
  nohupServer.out & tail -f nohupServer.out
  #32 bit version
  rm nohupServer.out; nohup ~/q/l32/q FASServer32bitInit.q -U
  credentials.txt> nohupServer.out & tail -f nohupServer.out
  ```

Log Upload Web Portal

A simple web interface (index.php) allows flight data (GPS and PID csv files generated by Betaflight blackbox tools) to be uploaded to the server.
Once the csv files are uploaded, a simple PHP script detects the number of features in each csv file. A manifest of the uploaded logs and its feature counts are stored into a new csv file called logsManifest.csv.
The web interface has a websocket connection to the kdb server (requires user authentication) that will then call the FASUpdate.q script to read logsManifest.csv file and enlist all the csv logs into kdb.
An as-of join is then conducted on the GPS and PID flight logs (required as GPS data is sampled ~ 5Hz and PID data sampled ~ 2KHz). With the joined data, qsql queries are used to generate a new table (trainingData) that contains features selected for the machine learning model training.

Launch Control ML Model Trainer (FASMLTrainingClientInit.q)

q client process that reads flight data from the kdb server to train ML pipeline implemented in it.
The ML pipeline consists of 3 cascaded models.
2 regression models that predict GPS speed and battery voltage respectively are first trained.
The physics of the UAV and its battery limitations are effectively modelled by the regression models after training.
By cross feeding the historical flight data between the two trained regression models, with an artificial time increment of 1 time period (0.2s, based on GPS speed refresh rate) between each cross feed step, an expanding combination of

possible future flight data is synthesized.
The synthesized data is arranged in a B-tree and the top 20% throttle sequences, based on final GPS speed, is selected using q-sql.
These throttle sequences are then transformed into a sliding window table in kdb and used to train the LSTM model.
The trained LSTM recognises the latest throttle input received by the UAV and generates the optimal throttle pattern for the next 2 seconds to maintain the highest GPS speed without overexerting the limits of the battery.
This trained LSTM model is used in the deployment of the Rolling Launch Control feature of the UAV.
When the LSTM model is done training, the script will save and transfer the trained model to the kdb server for storage over SFTP.

- To run:

```
rlwrap q FASMLTrainingClientInit.q
```

Telemetry Stream Client (FASMavlinkInput.py)

Python client that acts as the streaming telemetry interface between the UAV and kdb server.
It reads Mavlink telemetry data from the USB UAV telemetry radio attached to the local machine and streams it to the server using q IPC.
Requires qPython. Recommended to run on local machine with 3DR telemetry radio attached.

- To run:

```
python FASMavlinkInput.py
```

Launch Control ML Model Deployment (FASLaunchControlClientInit.q)

q client process that reads the latest 2 mins of flight data from the kdb server and generates Rolling Launch Control output for the UAV. The generated throttle sequence is streamed back to the server.
Under the hood, a trained keras LSTM model generates the output from the latest throttle data.
Requires 64 bit KDB and embedPy. Recommended to run on local machine with Tensorflow/CUDA.

- To run:

```
rm nohupLaunchControlClient.out; nohup q FASServerInit.q >
nohupLaunchControlClient.out & tail -f nohupLaunchControlClient.out
```

Launch Control Hardware Subscriber (FASLaunchControlOutput.py)

Python client that subscribes to throttle sequence generated by the launch control client on kdb server.
Requires 64 bit kdb and embedpy running on local machine, with USB interface to UAV control link.

- To run:

```
python FASLaunchControlOutput.py
```

Recorded Telemetry Playback Client (FASLogPlayback.q)

q client process that streams historical UAV flight logs and training data logs into the server to test client applications that require streaming telemetry input.
Simulates Mavlink telemetry sent back by the UAV.
IPC functions are defined on kdb server side.
Requires 32 or 64 bit kdb.

- To run:

  ```
  rlwrap q FASLogPlayback.q
  ```

Areas for further development:

- Effectively free up reserved, non-essential memory usage after ML model training/deployment
- Rewrite explicit Python variable deletion to work on negative checking. Delete if object is not:
  - Python system variable
  - variable inserted by q script
  - module imported
- Update log upload portal to tag logs with weather conditions during flight
- When running log playback, automatically flush RDB to HDB, purge RDB for incoming telemetry playback
- Create additional bash scripts to launch customised q processes based on operating system and kdb version
- Implement cross validation testing during training for GPS speed and LiPo voltage prediction models

- Implement transfer learning for LSTM to reduce subsequent training time and improve general accuracy from training with old and new telemetry data where applicable
- Further refine Rolling Launch Control ML training methodology
- Improve ML model deployment pipeline
- Implement option to use plaidml backend for keras, to allow LSTM training using OpenGL with integrated or AMD GPU, if CUDA is not available
- Implement Convolutional Neural Network to recognise scenario occurrences on dimensionality reduced flight data
- Integrate vanilla kdb ticker plant
- Improve feature engineering and selection capabilities with usage of more FRESH utilities
- Call 'del' on python objects not in use (especially during data synthesis and after model training is completed) to reduce Python memory bloat
- Update Tableau Dashboards view with more views useful for real-time flight engineering
- Switch to OpenFOAM for CFD work, with kdb integration for timestep logging
- Switch from manual CAD to machine-learning based generative design CAD with CFD to generate semi-optimal aerodynamic bodywork
- Deploy application as docker container
- Automate instance deployment, ML model archiving and instance teardown when retraining Rolling Launch Control on new data