Orchard Yield Estimation using Multi-Angle Image Processing

Leon Ripa

Submitted to the School of Food and Advanced Technology in partial fulfilment of the requirements for the degree of

Master of Engineering in Mechatronics

At

Massey University, Auckland

November 2023

Author . . . . . .

School of Food and Advanced Technology

November 7, 2023

Supervised by . . . . . .

Dr Khalid Arif

Orchard Yield Estimation using Multi-Angle Image Processing

by

Leon Ripa

Abstract

The rise of autonomous robots and deep learning techniques in recent times has sparked a surge in complex multi robot system (MRS), leveraging these technologies to handle intricate tasks and complement human labour. As the agricultural landscape has evolved from labour-intensive, small-scale farming to vast macro-managed expanses, precision agriculture (PA) has emerged to address the challenge. PA offers farmers micro-scale insights into their farms, enabling precise knowledge of pest presence, crop growth variations, and expected yields. For kiwi fruit farmers in New Zealand—spanning over 15,500 hectares and yielding more than 11.65 thousand trays per season—issues persist due to the absence of a mechanical harvesting solution. The inability to accurately estimate yields results in potential profitability concerns, often leading to over, or underemployment during fruit picking. Furthermore, the spread of viruses and diseases poses significant challenges, compelling the need to minimize human intervention and activity under kiwi orchards. Integrating PA techniques not only facilitates fruit counting but also provides crucial insights into fruit density, aiding in identifying underperforming areas for better farm management and potential yield enhancement.

This thesis introduces current methods used for orchard yield estimation and presents a novel approach tailored for estimating yields in kiwi fruit orchards. It discusses established solutions for similar agricultural challenges and explores their integration to devise the most effective method for estimating kiwi orchard yields. The proposed solution employs object detection through a convolutional neural network to identify, track, and count kiwi fruits. This is facilitated by images captured by a hexa-drone UAV flown beneath kiwi orchards, ensuring smoother camera capture for increased accuracy in object detection throughout the orchard. This data not only enables farmers to estimate current kiwi production but also aids in identifying sections of orchards that may be overperforming or underperforming.

Acknowledgements

I would like to express my deep gratitude to my supervising professor, Khalid Arif, for his invaluable guidance and support throughout this research project. His expertise and insights were pivotal in shaping my work and achieving the outcomes presented in this thesis.

I also wish to thank Muhammad Danial for his assistance with Pixhawk debugging and his expertise in drone building, which proved to be critical in the successful completion of this project.

Finally, I express my heartfelt thanks to my family and friends for their unwavering support, encouragement, and understanding throughout this journey. Their love and encouragement provided me with the strength and motivation to persevere in the face of challenges.

Abbreviations

UAV – Unmanned aerial Vehicle.

AGV – Autonomous Ground Vehicle.

PA - Precision Agriculture.

CNN - Convolutional Neural Network

DNN – Deep Neural Network.

CUDA - Compute Unified Device Architecture.

cuDNN - CUDA Deep Neural Network.

SSD – Single Shot Detector.

YOLOv7: You Only Look Once version 7.

FRCNN – Fast Region-based Convolutional Neural Network.

COCO: Common Objects in Context.

mAP: Mean Average Precision.

APtest: Average Precision on the test set.

APval: Average Precision on the validation set.

SVM - Support Vector Machine.

NMS - Non-Maximum Suppression.

RPI – Raspberry Pi.

CEDB - Coral Edge TPU Dev Board.

ASIC – Application specific integrated circuit.

TPU - Tensor Processing Unit.

TOPS - trillion operations per second.

HSV: Hue, Saturation, Value.

RGB: Red, Green, Blue.

RGB-D: Red, Green, Blue – Depth.

GPIO: General Purpose Input Output.

EMI - Electromagnetic Interference.

PDB - Power Distribution Board.

ESC - Electronic Speed Controller.

CCL – connected components labelling.

SLAM: Simultaneous Localization and Mapping.

EKF: Extended Kalman Filter.

MRS: Multi Robot System.

DOM: Digital Orthophoto Map.

DSM: Digital Surface Models.

DTM: Digital Terrain Models.

SfM: Structure from Motion.

MVS: Multi-View Stereo.

ExGR: Excess Green minus Excess Red Index.

PCB: Printed Circuit Board.

CCL: Connected-Component Labeling.

IoU: Intersection over Union.

SRCNN: Super-resolution convolutional neural networks.

SRGANs: super-resolution generative adversarial networks.

AG: Agricultural greenhouse.

CDF: Cumulative Distribution Function.

Contents

List of Figures

List of Tables

Chapter 1

1.1 Introduction

Autonomous robots and deep learning techniques have been gaining a lot of attention in recent times partly due to their rapid increase in development, real world uses and availability to the public. This has led to complex MRS's being developed which take advantage of these new technology's creating robotic systems capable of taking on more complex tasks than any one robot could, complementing human effort in a helpful way.

The introduction of agricultural machines such as tractors, and the vast types of harvesters has changed the way farmers operate and tend to their farms. From being labour intensive, small plots of land, being micromanaged, to now the large-scale macro managing of farms that extend for hundreds or even thousands of hectors. With the vast increase in farm sizes, the level of attention to detail, that used to be given to farms has been lost and is now too large a task for humans to take on. This is where precision agriculture (PA) comes into play. PA allows the farmer to have more micro scale knowledge of their farm, may this be for example, knowledge on where pests and diseases are present, where crops are not growing well or where they are doing better than expected. PA can allow a farmer to accurately know the crop yield he can expect or autonomously fertilise specific sections of crops or even detect harmful urine patches on farms[1]. In New Zealand one of the most popular fruits grown by farmers are kiwi fruit. With current agriculture techniques Kiwi farms in New Zealand now span over 15,500 hectares [2] and produce over 11.65 thousand trays per season [3].

Unfortunately for kiwi fruit farmers, a solely mechanical harvesting machine for the fruits has not yet been developed, and so picking the fruits must still be done by hand Figure 58. Issues arise when the farmers are unsure of the quantity of kiwis that their orchards have produced. Talking to local kiwi farmers, an average yield of less than 70 fruit per m^2 is considered non profitable. Additionally, during picking, farmers may hire too many or too few kiwi fruit pickers for the farm, lowering their yield or increasing the cost of production. With the handling of fruit picking being one major issue for kiwi farmers, another is the preventing of spread of viruses and diseases such as Cherry leaf roll virus and soil borne pathogens [4]. These can be spread faster from row to row or farm to farm if farmers are walking under and touching the vines and moving soil via their walking. Spending less time under kiwi orchards is preferred and so PA techniques that can mitigate the risk of spreading disseises is ideal.

In addition to counting fruits using PA techniques, higher level information about fruit density in specific regions of the orchard can also be obtained, allowing farmers to detect areas that unbeknownst to them are under performing, or areas that are over performing on average. Such higher-level Information can allow a farmer to regain important information to better manage and utilise current farm space to increase yield. Increasing yield can be done by grafting additional branches onto kiwifruit trees that are underperforming.

With two types of kiwi fruit being vastly dominant but different in colour and slightly different in shape, only the golden kiwi fruit will be discussed and talked about in this paper. Current trends in New Zealand show a steady increase in the percentage of golden kiwis being grown Figure 1. For this reason, green kiwis will be outside the scope of this project. Importantly only kiwi fruit quantities will be discussed, where judging the ripeness of the fruit is not in the scope of this paper.

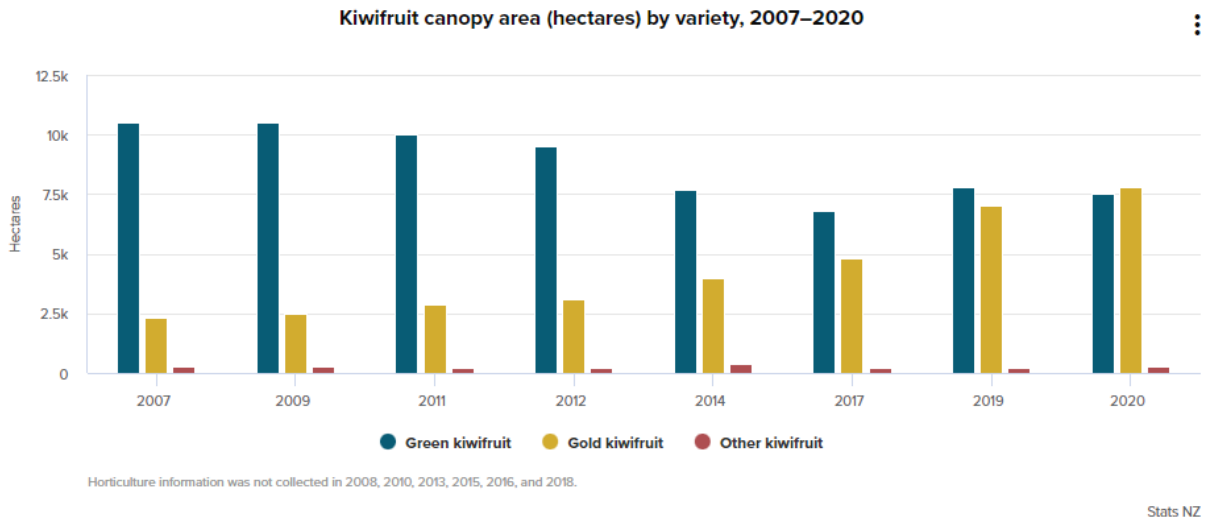


Figure 1: Kiwifruit Variety by canopy area.

N.Z. *Agricultural production statistics: June 2020*. 07 May 2021

1.2 Motivation

Motivation for this research comes in the form of having the ability to create a robotic solution for a large range of niche applications, living in rural New Zealand with kiwi farms in the area, adding to the niche of precision agriculture for kiwi farmers and what application they find useful. While talking to a small start-up company in New Zealand, it became clear that there is a high demand for PA in the Kiwifruit industry and has many research avenues to explore. Current research has been focused on finding the best method to count the fruits reliably. Several three and four wheeled remote-controlled robots, using a variety of camera, sensors and post processing techniques have been tested, but no commercial product has emerged hinting that the area of research is still ongoing to find the ideal solution. Noticing that no drones had been tested for such an application, the first idea was to fly a drone above the orchard, and simultaneously drive a remote-controlled car underneath. such a multirobot system could ideally identify more kiwi fruits and get more accurate results. This method turned out to not be as effective as originally thought, due to the heavy canopy above the fruits. Unlike most orchards, kiwifruits grow under the cover of a thick layer of leaves, hanging down on short vines under the leaves. The second method that is tested consist of flying a drone under the kiwi orchards. Having worked on a few projects focused on the use of Convolutional Neural Networks in the past, as well as ROS based projects utilising a Kinect Camera, the motivation to apply similar techniques to this project was a natural choice.

The aim of this project is to perform fundamental research that will determine whether flying drones under kiwi orchards allows for an optimal kiwi fruit counting method, and whether it is better than other current methods due to the inherent advantages. This information can be used to create useful products in the farming industry.

1.3 Project Scope

The scope of this project encompases the development and testing of methods that can identify the efficacy of counting kiwi fruits when using a drone flown under the orchard, as well as find other areas of research that are required to further develop this idea for widespread use of this technique in countries beyond New Zealand. The scope is restricted to developing the research platform necessary (the drone), as well as any additional platforms for testing purposes. In addition, the scope will be contained to the development of the appropriate kiwifruit detection framework needed. The scope will be limited using affordable and available components and software, where expensive hardware is less ideal for the overall aim of developing a cheap solution as well as staying within the projects budget. The development of a user friendly and visually appealing GUI will be out of this projects scope. Only the data for assessing the effectiveness of the fundamental ideas will be mandatory. The overall accuracy of the system will be used as a baseline that could be improved with hardware and software improvements, but perfection is not the goal here.

1.4 Objectives

To effectively accomplish the aim of this project, the following objectives have been identified.

- Identify and analys existing methods for orchard yield estimation in the literature, identifying techniques with high accuracy and effectiveness.
- Identify various drone platforms used for orchard-based precision agriculture implementation.
- Develop a research platform capable of carrying out all necessary objectives identified in relation to orchard yield estimation.
- Conduct experiments to determine the capability and effectiveness of the research platform.
- Develop kiwifruit detection method with a real time implementation for proof of concept.
- Conduct experiments to identify possible improvements or shortcomings with design and rectify where possible.

1.5 Thesis Outline

Chapter 1: Outlines the background information, the fundamental issue relating to further PA requirements in the Kiwi Fruit industry, motivations, and objectives within the project scope.

Chapter 2: Focuses on the literature review, current research methods related to drone yield estimation techniques are discussed, including multirobot communication and coordination methods that were researched prior to the initial testing phase. Additional in-depth methods used for object detection and appropriate hardware, that has been successfully implemented for orchard yield estimation, which is appropriate for this project is discussed.

Chapter 3: Investigates the physical side of the project, incorporating discussions of the physical development of the platform as it developed over time. Additionally, the hardware selection is discussed, including test methods and results that lead to a better final research platform. Finally in this chapter the electronics and circuits that are utilised / developed are described and discussed including some software features of the flight controller.

Chapter 4: This chapter goes into an in-depth discussion about object detection methods that were tested, test results are discussed, with an emphasis on the development choices and results of the resulting object detection YOLOv7 model are described.

Chapter 5: Discusses the experimental results obtained from the improved robotic drone platform and are compared to actual orchard yields obtained from picking.

Chapter 6: Discussions and review of the process/ methodology used to include justifications for decisions are made throughout the project, with future improvements also discussed.

Chapter 7: Concludes the thesis, disuses successful aspects and not successful aspects and describes how future works might proceed from here.

Chapter 2

Literature Review

2.0.0 Chapter Overview

The development of an agricultural drone, from design to onboard processing and algorithms spans a broad spectrum of research. For such a drone to fly remotely, autonomously and for extended periods of time with a reasonable weight requires research into drone layouts and hardware options as well as onboard flight controllers.

With the idea of using both a UAV (Unmanned Aerial Vehicle) and a AGV (Autonomous Ground Vehicle) simultaneously research into multi robot systems is required, to understand the ideal communication, cooperation and composition would best suit the kiwi orchard environment.

The key component of identifying and counting kiwi fruits hanging in an orchard must be performed by the done, via object detection and classification. Object detection and classification is a rapidly developing area of research and is able perform the task of counting objects with high speed and accuracy which is of the high importance in the scope of this project. Creating the ideal model for this application requires research into various methods of object detection, their various architectures, and implementation techniques, accuracy to be expected as well as processing power required are discussed in detail.

Sensors, microprocessors, and image capture techniques are researched to find the most effective and relevant for use on a drone, with the concepts of light weight, affordability, obtainability, and low power consumption aspects being some of the most important factors that are focused on.

2.0.1 Drone Platform

The drone platform should be able to carry all necessary equipment required for, navigation, object avoidance, and onboard object detection in real time, including the necessary cameras and sensors. The platform should be easy to use such that a farmer can operate it with minimal training. The flight time must be reasonable to conduct the necessary testing, but not to such a degree that a commercially available product would. The test platform must be modular to accept the testing of different components and configurations such to find an optimal design for further development.

2.0.2 Autonomous Ground Vehicle Test Platform

The ground vehicle platform is aimed to perform the same task as the drone platform and should also be able to carry all necessary equipment, for navigation, object avoidance, and onboard object detection in real time, including the necessary camera, with one exception. The ground vehicle may perform both the object detection in real time for the drone as well as for itself, as carrying the necessary equipment for this task is easier for a ground vehicle due to weight limitations on a drone. The ground vehicle must be easy to control and operate and be a modular test platform.

2.1.0 Multi Robot System Classification

MRS's are defined not only by the methods in which they communicate but also the relation that the robots within the given system have to one another. Thus, before deciding what communication system to use between robots it is important to define the type of relationship the robots have to one another. This is what MRS classification is. [5] has "defined five dimensions" in which any MRS communication/ information sharing methods can be classified under. These methods are as follows. Coordinated and Non-coordinated, composition, cooperative and competitive environment, communicating and non-communicating, reactive and deliberative communication Figure 2. The method used to communicate in a MRS depends on the task being performed.

2.1.1 Multi Robot System Coordination Overview

While standalone robots are useful there are many situations where the collaborative and coordinated work of multiple robots is needed to properly complete a task or increase efficiency to a level that any single robot could not achieve. Any number of scenarios can be imagined where more than one robot is needed, for example one application could be cleaning and picking up garbage from public areas. A flying robot can map, scan, and navigate a large area, locating coordinates for a ground-based robot to visit and perform cleaning or rubbish picking operations.

Complex multi robot systems such as this require a lot of coordination and communication to get the job done, this is usually archived via GPS capability's, RGBD vision systems, Wi-Fi communications, collision detection sensors and reachability. There are many more functions a multi robot system could utilise to complete a task, but choosing the ideal combination of functions, communication, and coordination techniques for any given multi robot system is very important.

Making any multi robot system a highly coordinated and communicating system may seem like a obvious answer, but it is not always so simple. There are many ways in which such a system should behave and can designed.

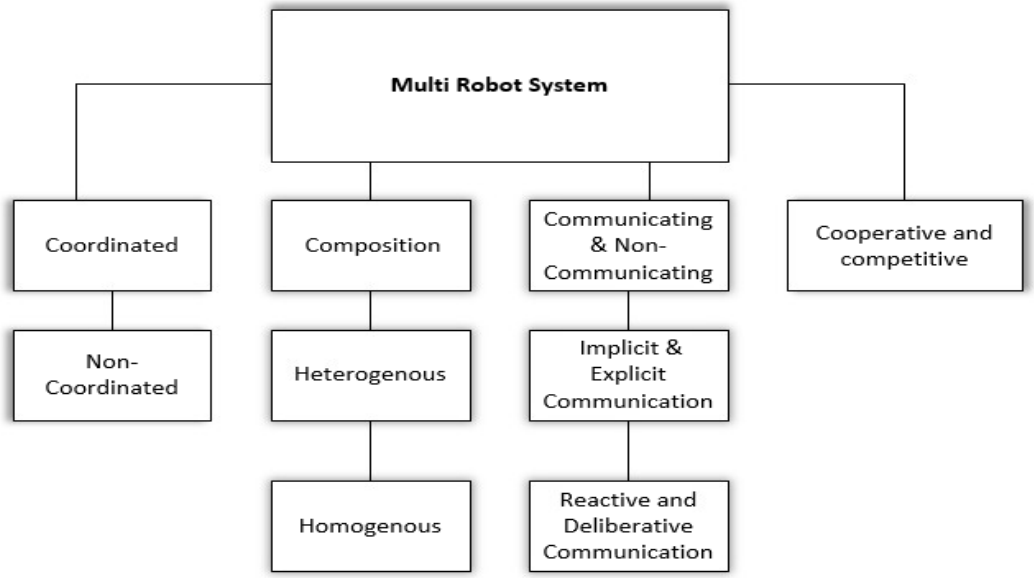


Figure 2: Multi Robot System's

2.1.2 MRS Coordination and Communication Classifications

Communication as discussed is essentially a method used to facilitate cooperation. Naturally when high levels of cooperation are needed in a MRS, high levels of communication are also required. Likewise, when a low level of coordination is used in a MRS, a lower level of communication can be used but this does not mean a low level of communication is required. When many multiple robots are used in a MRS not only does each “robot pay attention to their own works but also need to know if there are more urgent tasks from the partner” [6]. These varying levels of communication needs can be classified under such as decision-making protocol, speed of communication, centralised/ decentralised, and other hybrid variations. One such classifications can be seen in Figure 3

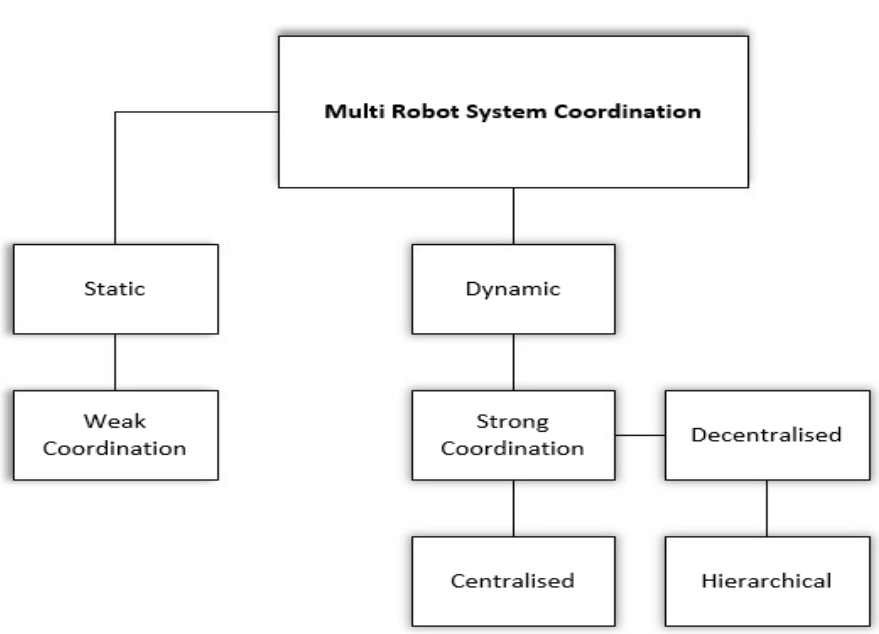


Figure 3 Multi Robot System Coordination.

2.2.0 Existing Drone Platforms

The idea of using drones for precision agriculture techniques is not a new idea, and many working systems have been developed on drone platforms. Fruit / tree counting with the use of a UAV have also been attempted. Four variants are discussed here; DJI Mavic 2 Pro [7]; custom HexaCopter [8] ; modified tarot T960 & APD-616X Agricultural spray drone [9]; modified DJI F550 HexaCopter [10].

The DJI Mavic 2 Pro platform used by [7] is designed to perform aerial surveillance, extracting canopy images from apple and pear orchards. Their workflow included construction of a digital orthophoto map (DOM), digital surface models (DSMs), and digital terrain models (DTMs) using the Structure from Motion (SfM) and Multi-View Stereo (MVS) approaches, as well as the calculation of the Excess Green minus Excess Red Index (ExGR) and the selection of various thresholds. This method resulted in a 95% accuracy in detecting the fruits. The use of the DJI Mavic 2 pro platform here benefits from the fine-tuned commercially available platform, with an integrated high quality 4k 30 fps capable Hasselblad L1D-20c camera. The use of a commercially available drone in this application is ideal as the drone is flown at 50m above the canopy, using only a GRB camera to obtain data. Additionally with the high-altitude flight no obstacle avoidance is necessary and the inclusion of the down facing camera of the DKI Mavic 2 pro is likely ideal for such an application. As the application is aimed at large scale counting, where regular use of such a technique is not required. This factor makes postprocessing of a custom desktop computer acceptable and lowers the required complexity of the drone platform.

The custom HexaCopter used In [8] hovers alongside orange tree orchards in a predefined path. The drone platform has frame arm dimensions of 21.5cm and a frame of 10.8 cm X 10.8 cm, such that the total diameter of the drone is around 53 cm. The drone uses a Pixhawk flight controller, capable of autonomous flight along predetermined flight paths. The use of the Pixhawk in this paper allows for pre-programmed take-off and landing, which is utilised in this paper for self-charging of the drone at the landing sight via solar panels. The flight controller allows for imaging individual trees by holding a hover position, before repeating the process at the next tree in the orchard Figure 4. A Raspberry Pi 3 Model B microcontroller mounted on top of the drone is employed to process the images after landing. Having employed a Rpi3B, the object detection via computer vision is ideal, and boats an impressive 80% accuracy in detecting the fruits, with consistent repeated results. With processing taking place after landing, the limited processing power of the RPI3B is somewhat overcome and is proven to work in this paper. To lower weight on the custom drone the frame arms are made of fiberglass, additionally the drone frame incorporates a inbuilt PCB making wiring easier while also saving weight on additional wires. The drone can take-off within 5 seconds, fly’s vertically at a hight of 10m, with a maximum flight speed of 3-4m/s. the 5000mAH LiPo battery provides a 15min flight time for the 3kg drone. Additional specifications can be seen in Table 1.

Table 1: Specifications of the Drone in [8]

Components	Commercial Drone	Customised Drone
ESC	Simonk 30A Single Shot	Simonk 30A Single Shot
Motors	1200kV	1400kV
Propellers	6”	10”
Weight	2Kg	3Kg
Radio Transmitter / Receiver	2.4 GHz 6 Channel	2.4 GHz 6 Channel
Battery	2200mAH	5000mAH

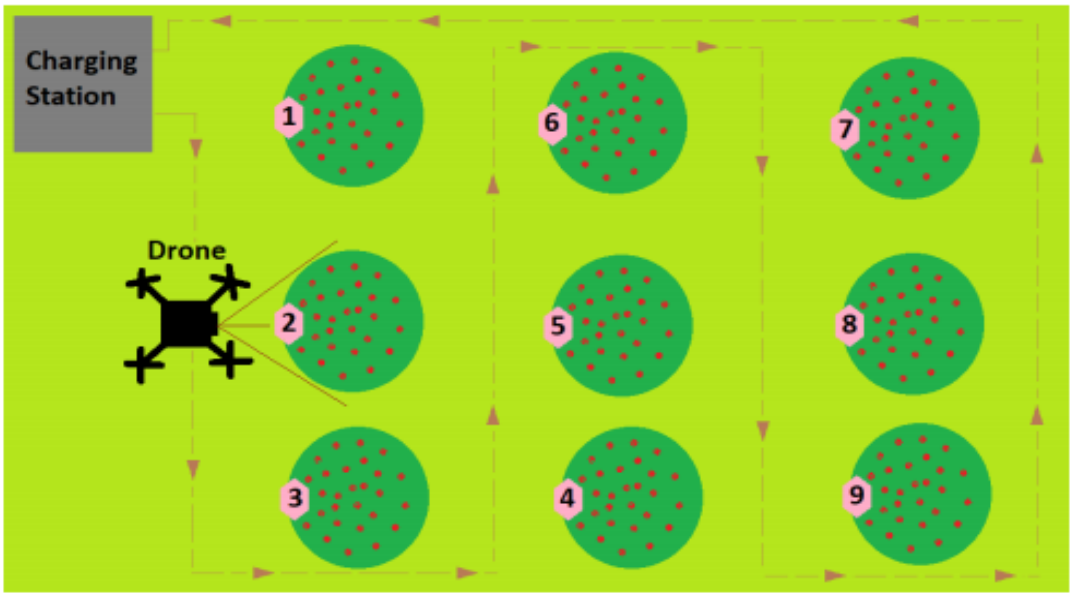


Figure 4 Aerial view of drone movement.

Surekha, P., et al. *An Automatic Drone to Survey Orchards using Image Processing and Solar Energy*. in *2020 IEEE 17th India Council International Conference (INDICON)*. 2020. IEEE.

The modified tarot T960 & APD-616X Agricultural spray drone used in [9] work together, where the modified T960 acts as a reconnaissance drone, identifying pests on longan fruit trees. The T960 tarot reconnaissance drone has a much smaller diameter of 1m, capable of providing around 2kg of thrust per arm when using 340KV motors and a 6S LiPo. With a bare drone platform and battery weighing on average about 4.5-5kg the drone can carry an additional 3-4kg of weight [11]. In [9] the reconnaissance drone uses a ZENMUSE X3 camera from DIJ, a Pixhawk 4 flight controller with a jetson TX2 being used for onboard image processing in real time Figure 5. To identify the longan fruit the drone uses a custom YOLOv3 model, later changing to the use of Tiny-YOLOv3, a smaller and less accurate, but less computationally heavy and thus faster architecture. The technique used shoes a mAP of .93% at 2.98 FPS and .89% at 8.71 FPS for the YOLOv3 and Tiny-YOLOv3 respectively. The combination of these two drones working together has a reported 95% control of the pest being sprayed, a reduction in water volume used by 12.5% and a 50% reduction in manual labour.

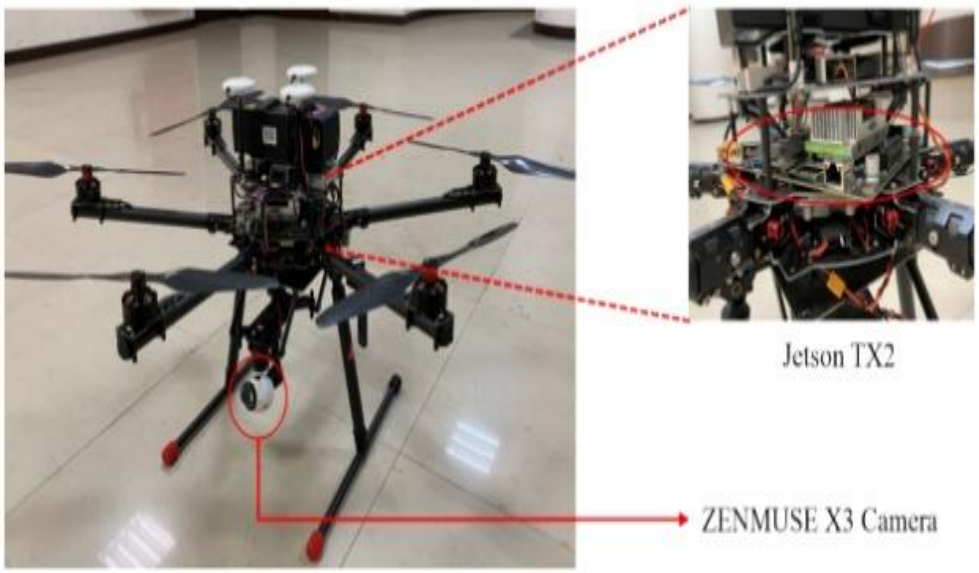


Figure 5 reconnaissance drone.

Chen, C.-J., et al., *Identification of fruit tree pests with deep learning on embedded drone to achieve accurate pesticide spraying.* 2021. **9**: p. 21986-21997.

A DJI F550 frame used in [10] which is also a hexa-copter is implemented with a Pixhawk flight controller once again, this time using two cameras, one being a monocular camera and a RGBD stereo camera that allows easier detecting of orchard boundaries Figure 6. The drone built here fly's horizontally next to the orchards allowing for a front on view instead of the usual above view gained by most UAV's. The paper tested the feasibility of orchard navigation with the use of monocular and binocular cameras, finding that the binocular RGBD camera was not able to be used to navigate areas in the same way a AGV can, this is due to the drone tilting forward during flight, causing the stereo reconstruction data to have an arbitrary rotational offset. Additional altitude variations introduce more noise. In this study the RGBD camera was not able to be demonstrate advantages over the standard monocular RGB camera. These results point towards a RGBD camera likely only having object avoidance and object detection benefits for a drone application.

Figure 6 DJI F550 frame with Pixhawk flight controller.

Stefas, N., H. Bayram, and V.J.I.-P. Isler, *Vision-based UAV navigation in orchards.* 2016. **49**(16): p. 10-15

In [12] the use of an Octocopter is employed to detect and count citrus trees in a cultivated environment spanning 73.5 ha in turkey. To count the citrus trees over such a large area the drone is flown at a hight of 55m at 5m/s. This pushed the need for an octocopter over a hexatone as it is more able to deal with larger gusts of wind and carry larger batteries for extended flight times over the large area. The frame used is a DJI v8 Octocopter model. Built very similarly to the tarot drones its frame, arms and landing gear are carbon fibre. The dry weight is 4.4kg with a 11Kg take-off weight [13]. The drone is powered by 320KV brushless motors, 6s 22000mAH LiPo. For added localisation accuracy a GPS triangulation method is used. Three identical GPS modules are mounted on the drone in a triangle configuration, this method greatly increases the accuracy of waypoint coordinates, and in-flight stability Figure 7. Although this paper uses three GPS modules any number of GPS's can be used to increase accuracy. To image the citrus trees a multi-spectral camera and RGB camera are used in combination. The Multi-spectral camera Figure 7, provides a larger amount of information, consisting of a red, green, blue band, Near-IR and Red Edge spectrums. These additional spectrums reveal additional information not visible to the human eye or RGB cameras, such as vegetation health, moisture content and other environmental factors. To count the citrus trees a comprehensive approach combining sequential CCL algorithm and morphological image operations was employed, the resulting accuracy was .979%, with the Red-Edge band providing the best results. To help with mapping the Pix4Dfields software is used.

Figure 7: Octocopter, GPS triangulation Left, Multispectral camera Right.

Donmez, C., et al., *Computer vision-based citrus tree detection in a cultivated environment using UAV imagery.* Computers and Electronics in Agriculture, 2021. **187**

2.3.0 Existing Fruit Detection Methods

The task of Fruit detection can be accomplished using the fundamental idea of image categorisation. Image categorisation is a technique used to pinpoint an image's characteristics and assign it to a certain label or category. This can be accomplished using a variety of different techniques such as Support vector machines (SVMs), computer vision, convolutional neural networks (CNNs), and deep learning are commonly used for image classification [14], [15].

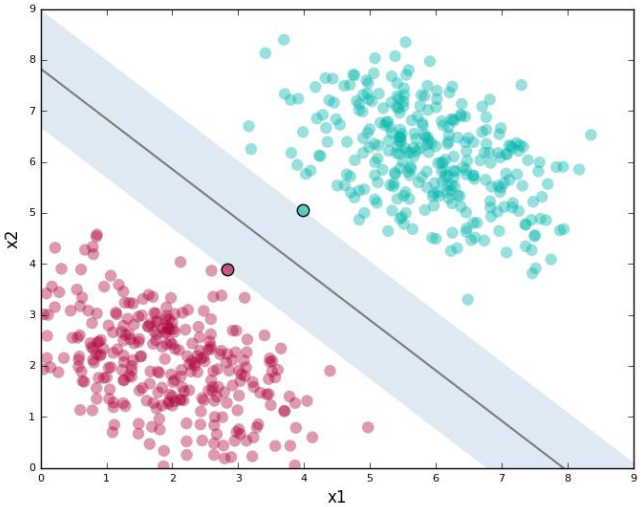


Figure 8: SVM visualisation showing margin.

Dutcosky, R., *Support Vector Machine (SVM) Practical Implementation.* 2020.

2.3.1 Support Vector Machines

All object detection techniques work on the principle of identifying points on an image, assigning the data points a value or weighting, followed by applying a given technique to separate the data points into predefined categories. SVMs are classified as supervised machine learning algorithms [16]. For SVMs data points are separated by what is called a hyperplane, this effectively separates classes data points, on opposite sides of the hyper plane (In two or three dimensions), with the distance that separates the categories being the margin. The goal of the SVM is to find the optimal hyperplane that maximises the margin between the two categories, this hyper plane is the support vector. This is a supervised learning algorithm, with advantages for small data sets [17], they can be easy to implement, use and interpret. Fundamentally it could be said that SVMs don't support multiple class classification although multi-class SVM can be created, and have been successful at identifying orange fruits in orchards[18]. Reference RBF image Figure 9.

In [19] with the goal of harvesting of citrus fruit to maximise market fruit value, a multi-class SVM was used to identify citrus fruits in orchards. To achieve this colour images of the fruit were taken on a clear day with natural daylight during the harvesting period when the fruits would be ready for harvest. The images contained both the fruit, leaves, tree-branches, and grass. Images were obtained using a colour CCD camera with 650*480-pixel resolution. Due to the complexity of the Images, standard computer vison thresholding and segmentation techniques were not successful in separating citrus fruits from the other "obstacles" in the image. Thus, a multi-class SVM was used. Using the features of each pixel to separate pixels into classes, such as sky, branch, leaf etc the citrus fruits were able to be segmented using a multi class SVM (using the one-against-one method), followed by a noise reduction thresholding method. To create a Multi class SVM, one can create multiple two class SMV classifiers, combining them to create a one-against-rest method or a one-against-one method, with other options / combinations also being possible. As only two classes can be compared at a time in a 2D SVM or 3 classes in a 3D SVM. Segmenting the fruits from the background after classification was done with an RBF Kernel function. The training of the Multi-class SVM was done using 87 images containing 592 mature Citrus fruits, with an accuracy claimed at 92.4% where the accuracy was calculated from the correct detection of 547 fruits out of a possible 597.

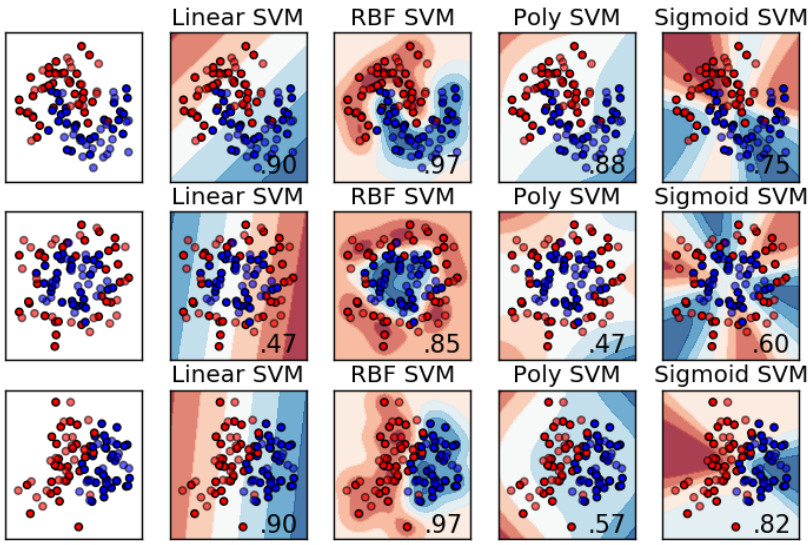
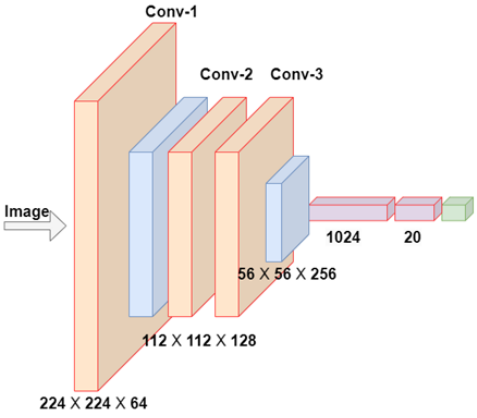


Figure 9: SVM Regularization RBF Kernel

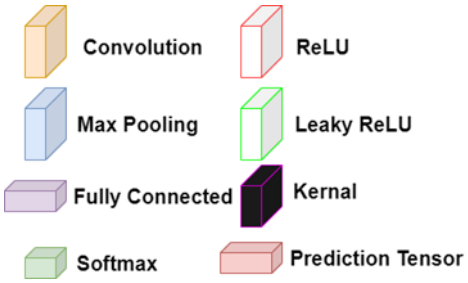
BILOGUR, A. Kernels and support vector machine regularization. 2018; Available from: https://www.kaggle.com/code/residentmario/kernels-and-support-vector-machine-regularization/notebook.

2.3.2 Convolutional Neural Networks & Deep Learning

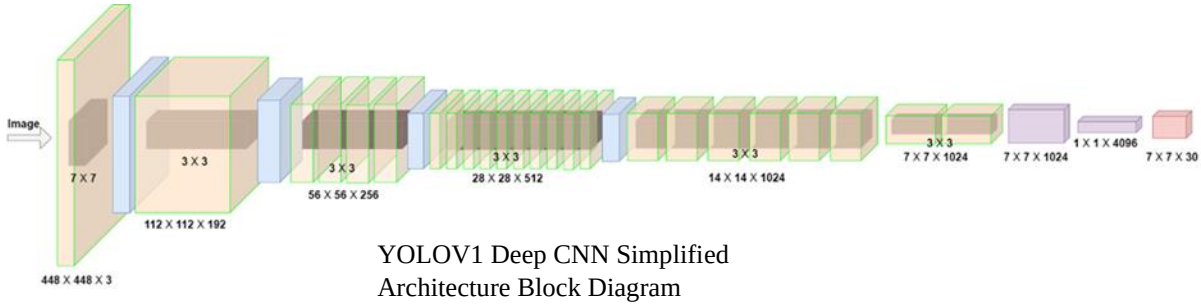
CNN's and deep learning approach are fundamentally very similar, with deep learning approaches being a more complex or "deep" version of the fundamental architecture that is a CNN. Deep learning methods typically use more layers to accomplish the same goal of image categorisation. These deep learning models are more computationally expensive than traditional machine learning models, however, generally offer higher accuracy and performance. As seen in Figure 10, the difference between a CNN and Deep learning is but a loose definition of increased complexity, where the separation point is in between the two examples shown. Model M2 in Figure 10 being that of the most simple CNN containing more than one Convolutional layer, developed by [20].Figure 10 also shows a box diagram approximation of the YOLOv1 Architecture, agreed in the literature [21], [22], [23] to be a deep learning model with an accuracy / mAP of 63.4 [24].



Model2 / M2, Basic CNN Architecture. 71.6% accuracy.



M2 & YOLOV1 CNN Block Diagram Key.



YOLOV1 Deep CNN Simplified Architecture Block Diagram

Figure 10: CNN vs Deep CNN diagrams with Key

2.3.2.1 Deep Learning & Tracking in Practice

The system created in [25] proposes the implementation of a multiclass-FRCNN, where one class is used to detect the presence of fruit and the other class detects quality of the sweet peppers. Their system takes inspiration from the DeepFruits technique [26]. Using a version of the FasterRCNN network, which utilises multiple convolutional layers, followed by a fully connected layer, is implemented in TensorFlow 1. Developing the number of classes, where N represents the number of quality grades, their system considers N + 1 classes, where one class is reserved for a negative class consisting of no fruits / background, and the remaining classes are for the fruit. Counting the number of fruits detected by the deep neural network, a tracking via detection method is proposed. The two-stage technique first starts by taking the initial frame of the image run and is considered the initialisation frame. All fruits detected in this frame are counted as unique fruits, and subsequently stored as active fruits and given an active track. On subsequent frames, detections are compared with previous detections, comparing their intersection over union (IoU), which is a method of calculating the “overlap” of the detections. This is possible with the set up as the camera is mounted to a driving robot where the camera is at a constant hight and angle, where the distance travelled between each frame is minimal. Thus, the IoU will be large for fruits that are being tracked. For the second part, detections that were not associated with an active track via the IoU equation, are evaluated to be counted as new fruits. This is done by calculating the boundary threshold between the detection and active tracks, the age of the detected fruit frames can be changed depending on the user’s needs, to fine tune the tracking ability.

2.3.2.2 Training Data & Effects on Accuracy

Data Acquisition for neural networks is a very important step to produce an accurate and versatile model that is able to perform under changing conditions. For example, being able to track fruits both on a bright sunny day where sun glare is present and also being able to track fruits on days where light levels are low due to cloud cover or rain. In [25] images of sweet peppers were taken using a RealSense SR300 for training. Images were taken in varying stages of maturity, with the largest amount of data taken when fruits were mature just prior to picking. Number of images taken was not specified. In [27] a model was trained on 1,120

images with 308 * 202 resolution, as well as images of size 500 * 500 pixels, with a total number of 9000 annotations of apples. Their FRCNN model has a yield estimation accuracy ranging from 95.56% to 97.83%. the resulting accuracy of a CNN can be directly linked to the quantity and quality of images it has been trained on as seen in [28]. Four different network architectures were tested on the same data sets, testing for accuracy before and after increasing the training data set size. Initial training was done using a “partial” data set of images taken from a drone flying over an apple orchard. The raw initial dataset is made up of oblique drone imagery where pixel resolution is 5472 * 3648, containing about 100 trees. Due to time restraints only 107 images containing 2,709 apples were used. The baseline models used vary in the backbone architecture, namely: inception V2, ResNet 101 and ResNet 50, later adding the Inception ResNet v2 Atrous network. For training, images were split 80:20 for training and testing data sets respectively, as well as an 80:20 split for training and validation data sets. After initial training the apple detection accuracy mAP were .5235, .6700 & .6160 for inception V2, ResNet 10 & ResNet 50 respectively. Increasing the data set size from 107 to 435 images, and the total apple from 2,709 to 13,439 resulted in an average mAP increase of 13% after approximately quadrupling the data set Figure 11. Class discrepancy also shrunk.

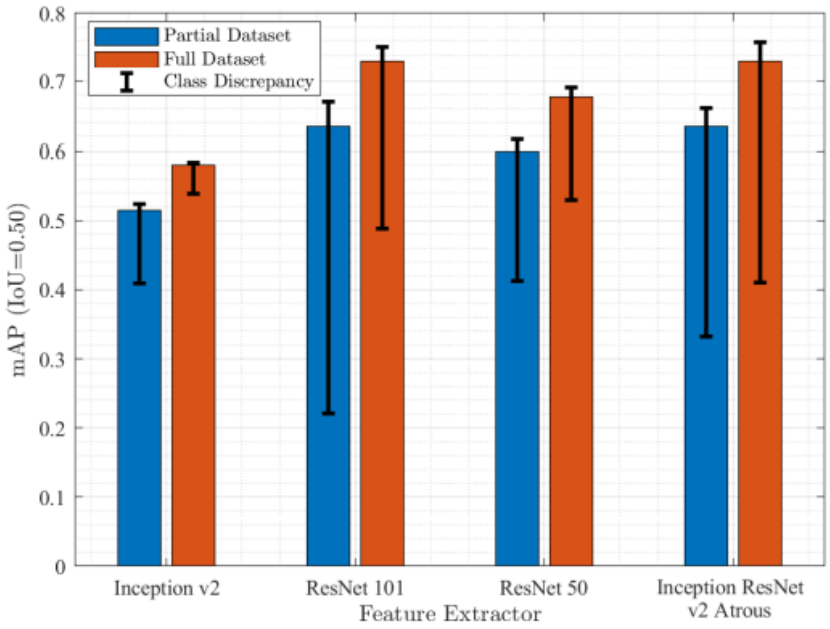


Figure 11: Effect of more training data on a deep learning-based apple detection mAP

Baird, A. and S.J.a.p.a. Giani, *An Artificial Intelligence System for Combined Fruit Detection and Georeferencing, Using RTK-Based Perspective Projection in Drone Imagery.* 2021.

Varying the training image resolution can have a significant effect on the final accuracy of a neural network, Higher resolution images have more detail, allowing the network to better distinguish between different classes[29]. However, higher resolution images also require more processing and memory to train a deep CNN. Therefore, the trade-off between accuracy and speed must be considered when choosing a resolution for training, but importantly the increase in computation time is only seen on the training side and is not carried over to the speed of the model in terms of image classification making the issue more of a “one off” trade.

Where increase image resolution allows for more detailed models, Down sampling images to a lower resolution can cause information loss, making image recognition more difficult. As its important to remember the convolution within the network already attempts to down sample image data to increase training speed when it comes to the fully connected layer. To address this issue, [29] found that an ensemble of models trained with high and low-resolution images could achieve a lower top-5 error rate than a single model trained with either resolution. Super-resolution convolutional neural networks (SRCNNs) and super-resolution generative adversarial networks (SRGANs) can be used to up sample images to a higher resolution.

Including high resolution and down sampled images within the training data set does help increase model accuracy, but other methods of data augmentation can also be used to increase data set size and training accuracy. Some common and basic techniques include Flipping, colour space transformations, cropping, rotation, translation, noise injection, kernel filters, random erasing, and image mixing [30]. Combining these image augmentation techniques is also common practice, with the use of these techniques individually, top-5 accuracy scores can see an increase on average by 5.6% [30], with even higher increase in accuracy given the use of a number of these techniques.

The performance of the Faster R-CNN, YOLO v3 and SSD models was tested in [31] for AG (Agricultural greenhouse) detection from high-resolution satellite images. For their tests two sets of images were used, GF-1 consisting of images with a resolution equal to 2 meters and GF-1 where resolution was halved to 2m. The best results were achieved using the YOLO v3 network, with Faster R-CNN providing accurate localization. Although SSD had lower accuracy when using lower resolution images, it was faster than Faster R-CNN. Spatial resolution is a key factor affecting detection performance and higher resolution yields better detection quality. It’s important to note that YOLO models are also SSD networks and have a faster detection rate as a fundamental feature of the networks simplicity.

Table 2: Metric comparison of various models with varying training data and resolution.

Li, M., et al., *Agricultural Greenhouses Detection in High-Resolution Satellite Images Based on Convolutional Neural Networks: Comparison of Faster R-CNN, YOLO v3 and SSD.* Sensors (Basel), 2020. **20**(17)

Metrics	Faster R-CNN	YOLO v3	SSD
mAP (GF-1 & GF-2)	86.0%	90.4%	84.9%
mAP (GF-1)	64.0%	73.0%	60.9%
mAP (GF-2)	88.3%	93.2%	87.9%
FPS	12	73	35

2.3.3 Computer Vision

Computer vision differs from the other object detection techniques as it follows a loose set of "rules" that allows a user to implement various algorithms that when applied in specific orders to images can result in the specific segmentation and identification of objects within an image. As Computer vision refers to the use of image manipulation via algorithms. Many researchers use pre-made libraries containing appropriate algorithms such as those provided by OpenCV to easier implement their ideas. OpenCV's powerful libraries can be used to identify the objects of interest, detect faces and facial features, recognize objects in a scene, and detect objects in videos. OpenCV also provides a range of pre-trained models that can be used for object detection. These models provide a good starting point for object detection tasks and can be used to quickly build an object detectors and counters for a specific task.

The work conducted by [32] looks to computer vision, assisted by open CV libraries to achieve detection and yield estimation of fruits. The techniques employed follows a familiar series of steps used by many computers vision architecture to achieve accurate object detection. A basic overview of steps that may be taken are as follows.

1. Acquire image.
2. segment fruit by colour and shape analysis.
3. colour analysis via thresholding.
4. shape analysis via contours and connected components.
5. counting via circular fitting

In [32] Images are captured and pre-processed using a graph cut method Figure 13, the background is suppressed by use of inbuilt convolution to extract the foreground. The foreground is then separated into individual colours after HSV from RGB conversion Figure 13. This conversion in necessary as with RGB, data can be hard to group in terms a computer can understand. This is where HSV and other colour space conversions become very useful.

An HSV conversion changes the saturation and hue value of the image; thus, the image can be decomposed into several parts, differentiated by colour. With the HSV conversion and decomposition the images components of interest can be extracted via thresholding. The thresholding range is manually selected to work with the object in question, also known as colour thresholding Figure 13.

Shape analysis is used to separate the now remaining individual and overlapping fruits within the image into individual components. This is done via canny edge detection Figure 13. Cany edge detection is a combination of inbuilt processes, namely, smoothing, gradient computation, non-maximum suppression, and thresholding. This method can give the optimal edges of the individual objects. To lower the noise remaining in the image a gaussian filter is used. The system used in [32] produced an accuracy of 83.33% for orange detection and a accuracy of up to 100% for cherry, mango, pomegranate, apple and lemons. Although their sample sizes were small the true accuracy of such a system implemented at large scale will certainly be less favourable.

Table 3: Computer vision counting accuracy of various fruits.

T. GAYATHRI DEVI, D.P.N., S. SUDHA, *Image Processing System for Automatic Segmentation and Yield Prediction of Fruits using Open CV.* International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC), 2017: p. 758–762.

Fruit	Colour	Actual count	Algorithmic count	Accuracy
Cherry	Red	8	8	100%
Mango	Yellow	2	2	100%
Orange1	Orange	9	8	88.88%
apple	Red	3	3	100%
Pomegranate	Red	3	3	100%
Orange2	Orange	18	17	94.44%
Orange3	Orange	12	10	83.33%
Lemon1	Yellow	14	13	92%
Lemon2	Yellow	3	3	100%

In [33] a computer vision AGV system is described for the use of detecting trees and picking oranges, the system uses two cameras and a Ultra sonic sensor: ZED Stereo camera for long distance measurements and tree detection and a web camera mounted on the robotic arm for picking.

Processing is done via an onboard laptop, with an Arduino micro controller responsible for the movement of the AGV and robotic arm and wheels. The system is divided into two main parts, one for detecting trees and the other for detecting oranges. The tree detection is accomplished

via green detection in a YCrCB colour space (Testing was done indoors with little green other than the trees present).

The system detects the tree by first handling the calibration of the YCrCB minimum and maximum values necessary for green colour detection. Once the calibration of the colour is correctly filtered, the filtered tree colour is stored in a threshold. Morphological operations, such as dilation and erosion, are used to eliminate the noise background that surrounds the tree. Erosion erodes away the boundaries of the forefront pixels, while dilation gradually enlarges the boundaries of regions of pixels that are in front. The tree is detected when the contour area of the tree provides a set of x and y coordinates as the centre of the tree area. The system then moves the AGV to the detected position of the tree using the value of the centre of the contour area with help of the depth given from the ultrasonic sensor Figure 12 left.

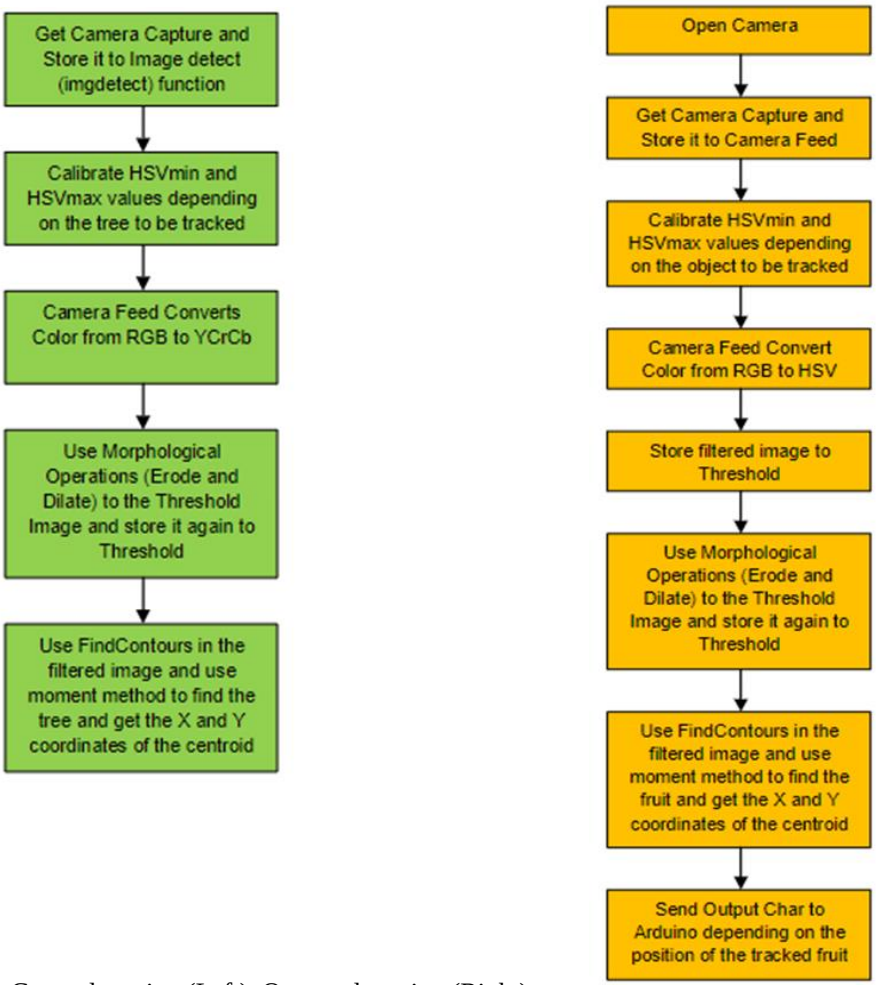
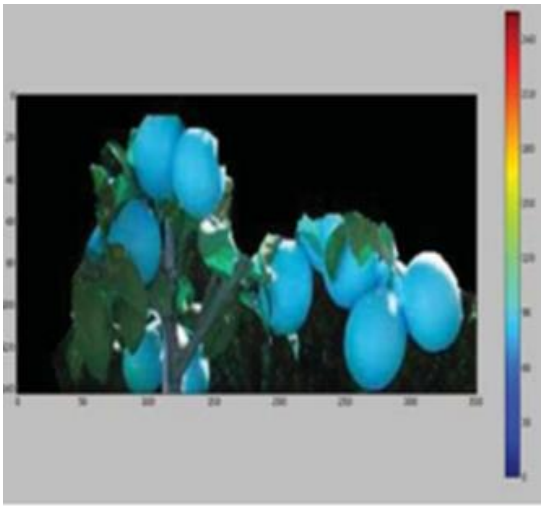


Figure 12: Green detection (Left). Orange detection (Right).

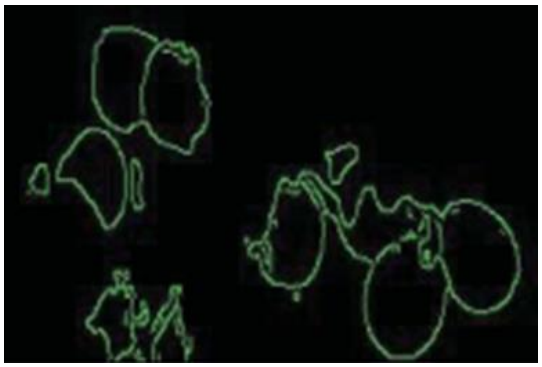
Kathleen Anne M, A., Rona Mae G. Babaran, Bryan Jones C. Carzon, Karl Patrick K. Cu, Jasmine M. Lalanto, and Alexander C. Abad., *Autonomous Fruit Harvester with Machine Vision.* Journal of Telecommunication, Electronic and Computer Engineering, 2018. **10**: p. 1-6.

For the detection of oranges, a slightly different approach is used. The process starts with video capture from the web camera, which is then processed using OpenCV in Microsoft Visual Studio. The captured image is converted from RGB to HSV and the HSVmin and HSVmax values are calibrated for the types of orange in question. Next, the image is divided into five divisions horizontally and vertically to make the movement of the arm more precise to how the orange is detected. Morphological operations such as erosion and dilation are applied to the threshold image to filter out noise and improve the accuracy of the detection. The filtered image is then passed through the Find Contours function, which identifies and locates the fruit within the image using the moment method. Finally, the X and Y coordinates of the centroid of the fruit are obtained and sent as output characters to the Arduino for movement of the robotic arm Figure 12 right. The arm had a reported accuracy of 26/30 correct grips (the accuracy is a combination of both the detection system and the programming of the arm movement). Overall, this project demonstrates the successful implementation of an autonomous fruit harvester using machine vision and OpenCV, which can be used not only in fruit picking but also in the manufacturing industry for pick and place processes

Image Input



HSV Image.



Canny Edge Detection

Segmented / Connected Components

Figure 13: Computer vision object detection in parts.

T. GAYATHRI DEVI, D.P.N., S. SUDHA, *Image Processing System for Automatic Segmentation and Yield Prediction of Fruits using Open CV.* International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC), 2017: p. 758–762.

2.4.0 Image Capture and Range Finder

To get a precise yield estimation of kiwi fruit and avoid hitting the orchards with the drone a well-suited imaging, obstacle avoidance and global positioning system is required. With the goal of object avoidance many 3D sensors and mapping technologies fall into the scope of appropriate technology to utilise. Creating a 3D geometric representation of the environment is a useful idea, given the requirements of the project. However, to simplify the project, and keep the cost down, 3D mapping techniques such as SLAM (Simultaneous Localization and Mapping) will be out of the scope of this project. There are numerous amounts of technology that can be employed as a solution, either through one integrated solution that combines both RGB imaging and range finding technology into one product, or the use of individual standalone RGB imaging and range finding. Both options have their advantages and disadvantages. The various technologies available to the project will be discussed in detail.

Figure 14: Image and ranging sensors. Zenmuse x3 [Left], RealSense D435 [Middle], TFMini-s [Right]. [43] [36] [40].

2.4.1 RGB Camera / Neural Network

The use of RGB cameras Figure 14, [34] to perform object avoidance instead of object detection is a far less common practice, especially when looking at the quantity of literature. To achieve obstacle avoidance with a RGB camera as with object detection computer vison or neural networks are required/ implemented. One example as seen in [35] demonstrates a UAVs ability to avoid trees in a park via a three class object avoidance geared neural network based on Faster R-CNN and Inception v2, and shows the use case for basic object avoidance via a CNN. The type of camera that is used for either object detection or avoidance via a CNN would have the same requirements. With many available on the market, the available options are large. As described in [29] a high resolution camera brings an increase in correct detections, a large field of view to capture the full breadth of a kiwi orchard (around 3-4 meters as measured locally). With sun glare being a real issue faced a camera with good exposure

control will help increase detection accuracy. When it comes to full frame camera a high shutter speed will be required to decrease motion blur, with image stabilisation being advantageous. The selected camera also requires a port for a live video feed. For a full-frame camera, the Canon EOS 5D Mark IV or Nikon D850 are good potential candidates for the purpose of image capture in orchards with drones. Both cameras can capture high quality video at 4k and offer a wide range of features, such as fast auto-focus, weather-resistant bodies, and long-battery life. Additionally, they are relatively light for a DSLR. Other potential camera options include the zenmuse x3 Figure 14, equipped with a wide angle lens, capable of 4k 30FPS video, and is specifically designed for DJI drones & successfully used by [9]. Currently the most used cameras on quad copters are the Yuneec Q500 4K, DJI Phantom 3 Professional and the GoPro HERO4 Black. With the use of a gyro, cameras without image stabilisation can become more suitable.

2.4.2 RGB-D Camera

RGB-D cameras Figure 14, [36], are a type of 3D camera that combines the capabilities of both a traditional RGB camera and a depth sensing camera into one module. There are two fundamental methods used to measure depth when using infrared light, namely time of flight and structured light [37]. As the name suggests time of flight sensing relies on measuring the time taken for light to bounce back from an object to determine its distance. The second variant uses structured light, this is accomplished by projecting inferred light to display a pattern of IR dots on its target, the pattern is then detected by a sensor, and by measuring its distortion of the unique pattern the depth can be calculated. This allows RGB-D cameras to accurately measure the distance between objects and the camera itself. The sensor then takes the data from the infrared projector and creates per-pixel depth information. This map is used to generate an image or 3D model of the environment, which can then be used for various applications including navigation and commonly used for SLAM and 3D mapping and tracking [38], as accurate integration of the depth and the colour images can provide robust frame matching and loop closure. More importantly for this project the advantage here would be the use of RGB-D for object recognition a seen in [26], [39], [40] being complementary to obstacle avoidance, as depth data can be correlated with the RGB camera, yielding an RGB image with a depth associated with each pixel. This is often represented as a depth cloud and converted into a point cloud.

Common RGB-D sensors include the Microsoft Kinect-1 & 2, ASUS Xtion pro, Intel RealSense and Orbbec Astra. Overall RGB-D cameras provide a denser information regarding the

environment, are useful for SLAM, 3D mapping, object detection, and pose estimation in robotics. They also have significant drawbacks including limited measurement ranges where depth information lacks in accuracy, resolution when compared to its RGB counterpart. With outdoor use of IR technology having the additional hurdle of IR noise emitted from the sun, the additional cost must also be considered, where such RGBD cameras can easily be multiple times the price of a comparable RGB camera.

2.4.3 Lidar / mm-Wave Radar

Lidar and millimetre wave radars Figure 14, [41] are active imaging sensors that have been employed in autonomous vehicle systems to provide relatively accurate measurements of obstacles in low visibility conditions. Lidar is an acronym for “light detection and ranging” and uses eye-safe laser beams to create a 3D representation of the surveyed environment. It is used in many industries, such as automotive, infrastructure, robotics, trucking, UAV/drones, industrial, and mapping [42]. MMW radar provides consistent range measurements and produces a large beam, so radar returns may be interpreted using the interaction of the beam with a finite but relatively large region of the environment. The TF mini–Lidar Module is a low-cost, miniature, and accurate ranging sensor with an excellent price to performance ratio. It is designed specifically for applications such as UAVs, robots, and autonomous vehicles. It offers a wide range of features, such as built-in motor driver, low-noise and low-power consumption, wide detection range, and high precision. It is capable of detecting objects up to 12 meters away, with a resolution of 0.4cm [43]. It is also able to detect objects in complete darkness. Mm-Wave radars are often found on Autonomous vehicles performing object avoidance. Like the RGBD camera, the mm-Wave radars are susceptible to noise from sunlight and reduces the effective detecting range in outdoor environments during the daytime. Additionally, such radars have a relatively small detection area, which increases in a cone shape, the further away the object lies. Other Lidars such as the OS0 sensors can be used on a drone to perform more complex object avoidance tasks or mapping via a SLAM technique. Such lidars are in the range of 0.5Kg and have many upsides when used for the right tasks, but for object avoidance in an orchard where 3D mapping is not required would be a bad choice.

2.4.4 Sensor Fusion

As expected, all sensors have advantages and disadvantages in their own regard, though some of the drawbacks can be mitigated via sensor fusion. Sensor fusion is the process of combining two or more sensors that gather similar information, whose data can be combined to reduce errors, increase accuracy, or increase the range of collected data [44]. The use of two separate yet identical sensors can also be used for sensor fusion to gain a wider range of view, as well as the use of two or more different and non-identical sensors such as a RGB camera and mm-Wave Radio. In [45] sensor fusion of a Xtion Pro RGB-D camera and Hokuyo UTM-30LX LiDAR were used for the tracking of a moving object. To locate the moving target visual tracking algorithms, depth information of the structured light sensor, and a low-level vision-LiDAR fusion algorithm were used. The use of these two sensors allowed the researchers to overcome the limited depth accuracy of the RGBD camera. The RGBD cameras depth accuracy significantly decreasing at a range of 5 meters. The 2D LiDAR scanner being tested up to 8 meters and proving to be accurate throughout the range. Although the LiDAR's range well exceeded that of the RGBD camera, the overall tracking failed past 5 meters due to the limited sensing range of the Xtion Pro GRB-D camera. Nevertheless, the sensor fusion improved the object tracking accuracy within the 5m range proving a robust approach for the given application, and highlighting the limits of sensor fusion, where not all the shortcomings of one sensor can be counteracted by the addition of a different sensor.

[46] Used sensor fusion to create an indoor object tracking system through the combination of two IWR1642BOOST mm-Wave radars via a laptop. Fusion of the two sensors data was achieved using a UKF (uncentered Kellerman filter), after finding it to be better performing than the EKF (extended Kellerman Filter). The methodology involved generating and parsing point clouds, reducing noise, data fusion, clustering objects, identifying centroids, and tracking the centroids using the UKF algorithm. The results showed that the fusion system was more accurate than previous systems, with an error of .2136m in the X direction and .2290m in the y direction. The average position errors obtained from a Texas Instrument sensor was .5401m and .5601m in the X & Y directions respectively. Combining the data of two mm-Wave radios improved accuracy, decreased error from occlusions at higher numbers of weak data and increased the field of view, claiming the fusion of these two sensors improved tracking accuracy dramatically.

2.5.0 Microprocessors

Figure 15: Microprocessors. Coral Edge Board [Left], Raspberry Pi 4 [Middle], Nvidia Jetson Nano [Right]. [47] [50] [54]

Microprocessors are small, integrated circuits that contain comparable components to that of a desktop computer. Microprocessors are found in smartphones, tablets, complex home utilities, cars, and drones to name a few. The value in microprocessors lie in the fact that they are essentially a very small computer, but with the addition of GPIO (General Purpose Input Output) pins that are not found on home computers, this allows them to be used to take outside information collected from sensors such as cameras, microphones, touch screens, manipulate the data and produce an output to motors, screens etc. all microprocessors contain a CPU, RAM, Memory, and a range of external ports and peripherals [47]. The main differences when it comes to various microprocessors are the potential inclusion of a GPU, the architecture, clocks speed, number of cores, available ports, number of IO pins, storage space, size, power consumption, price, and availability. These are all import aspects to consider when choosing a microprocessor for a given application. For the use of object detection in real time when mounted to a drone the ideal microprocessor would have a combination of small footprint, low weight and power consumption, good graphics processing / inference power and in the scope of this project a low price and ease of availability are important.

Table 4: Microprocessor and Edge TPU Accelerator Comparison

Micro Processor / accelerator	Size	CPU	GPU Cores	Power consumption	RAM	Price
Jetson TX 2	170*170mm	Dual Core 64 bit OR Quad core ARM	256 NVIDIA core + 256 CUDA cores	7.5-15W	8GB	680$
Jetson Nano	80*95mm	Quad core ARM	128 CUDA cores	5 - 15 W	4GB	350$
Raspberry Pi 4 model B	86*56mm	Quad core 64 bit	Integrated graphics	5W	1/2/4GB	230$
Google Edge TPU Developer Kit	86*56mm	Quad core 64 bit	Integrated graphics + 2X TPU	5 - 12.5 W	1/2/4GB	280$
Coral M.2 Edge Accelerator	22*30mm	-	2X TPU	4W	-	143$
Coral USB Edge accelerator	65*30mm	-	1X TPU	2W	-	200$

2.5.1 Google Coral Edge TPU M.2 & USB Accelerator

Figure 16: Coral Edge USB TPU [Left], Implemented on drone [Right].

Safa, A., et al., *Exploring Cross-fusion and Curriculum Learning for Multi-modal Human Detection on Drones*, in *System Engineering for constrained embedded systems*. 2022. p. 2.

The Google Coral M.2 and USB Edge TPU Accelerators [48], are small form factor ASIC Devices developed by google for the use of high-speed inference on TensorFlow neural network models. Essentially a comparable yet more efficient version of a GPU, that can only perform inference on Models of the correct architecture. Thus, models such as YOLO would not run on these devices. The Coral Edge TPUs are designed to also be more power efficient and cannot be used to train models. The devices vary only in terms of connector and amount of Edge TPUs contained within. The USB version Figure 16 has a USB connector and one Edge TPU, whereas the M.2 version can connect to a PCIe sort, A+E, B+M or M.2 E-Key slot, the latter containing 2 Edge TPUs. Each TPU is capable of 4 TOPS (trillion operations per second) at 2 TOPS per watt [49] Table 4. Looking at current availability of these TPU accelerators it is clear sourcing one would be a challenge, likely due to worldwide chip shortages. Work produced by [50] showed the use of a Coral Edge TPU combined with a Raspberry Pi to perform indoor detection of people via sensor fusion or a RGB camera, and mm Radar via a multi-modal CNN with cross fusion highways Figure 16. With the use of the raspberry pi and Edge TPU the cross-fusion CNN was able to operate at 30FPS (limited by the camera). Although not specified what TPU module was used (as the raspberry Pi has connectivity for various versions) or how much the TPU aided the performance of the custom CNN the addition of the Edge-TPU was likely necessary as the small Quad copter would not have unnecessary components added due to weight constraints. A microprocessor using Edge-TPUs will massively benefit from their low weight and power consumption compared to their huge computing power (although limited to the ASIC operations within the TPU.)

2.5.2 Raspberry Pi

The Raspberry Pi (RPi) Figure 15, [51], is the most common and most community supported microprocessor board on the market today, aimed at students, hobbyist, researchers, and often used in robotics applications. With its long history there have been many iterations since 2012. Namely the RPi Model A, B the RPi 2 and 3, and now the most powerful to date the RPi Model 4. Powered by a ARM8 64-bit processor running Linux, inbuilt Wi-Fi, Bluetooth, and extensive range of I/O ports is what makes the board such a strong contender when it comes to microprocessors. The processor included with the Raspberry Pi Model 4 is a quad-core ARM Cortex-A72, although powerful, it is the least capable when compared to the NVIDIA Jetson Nano and Google Coral Edge TPU Dev Boards processors, nor does it have ports for multiple high definition cameras through USB ports instead uses specialised MIPI CSI camera interface via the 40pin GPIO header, making it a trickier board to integrate with assorted hardware. As with the TPU accelerator RPIs are difficult to source at the time of writing. As shown by [50] the Raspberry pi is capable of fast real time object detection and computation of sensor fusion, but this was with the help of a Coral Edge TPU accelerator. Without the help of such an inference ASIC the Raspberry pi is a less capable board as shown in [52] the use of a RPi 3 Model B+ which is less powerful than the RPi 4 with the main difference being an increase from 1GB of RAM to 4GB of RAM. Never the less the results shown by [52] indicate that the use of a standalone RPi on a drone used for object detection was subpar. Using the SSDLite_MobileNET_V2 the RPi was performing at .71FPS/ 2.29% of the max FPS, where 100% would allow processing of 30 FPS. With the CNN processing 0.71 FPS, the real time response is not good enough for real time applications and would be a non-safe solution for the use on a drone. In [53] the RPi 3 Model B+ was implemented for the use of tomato detection via computer vision techniques, with a aim to perform colour discernment of ripe and unripe tomatoes, using OpenCV and a webcam. Although the algorithm is much less computationally expensive the RPi 3 was able to accurately discern between ripe and unripe tomatoes with an accuracy of 98%, making the use of the microprocessor appropriate in this case. The rate of throughput of tomatoes by the conveyor limited need for a fast implementation of computer vision, so the full extent of the capabilities of the RPi are not shown. Using machine vision with the RPi is a better choice than the use of a CNN as machine vision implemented through Open CV does not require large GPU processing power and relies more on the CPU, which is still very powerful on the RPi, thus likely giving such a good result in the detection of various ripe and unripe tomatoes.

2.5.3 Google Coral Edge TPU Dev Board

The Coral Edge TPU Dev Board (CEDB) Figure 15 is powerful microprocessor board specifically designed for machine learning tasks, includes a large number of ports and I/O pins, and includes an integrated dual Edge TPU processing unit [54]. The board finds itself most used in robotics and autonomous applications, edge computing, image and video processing, natural language processing and object detection in real time such as with surveillance cameras. As with the RPi the CEDB also runs on a Linux operating system, with the addition of the TensorFlow Lite API allowing for an easier setup of CNN. One drawback of the CEDB is that it can only perform well on TensorFlow Lite based neural networks, while being a popular and well established and well performing architecture it does limit the type of models available, some work arounds can be achieved to allow some YOLO Lite models to run using the TPUs also. With the board only running Edge TPU optimised models efficiently this limits the number of architectures that can be tested in a research-based environment, causing a researcher to be forced to use a TensorFlow Lite model in their projects without knowing if such a model is best suited to their application. Other than the TPUs imbedded on the board the integrated graphics included are not comparable in terms of running a non-TensorFlow model thus severely limiting the boards capabilities. Some advantages that the CEDB has is the fact that the I/O pins are based on that of the RPi. This allows a level of familiarity of the new board to researchers, reducing the learning curve, additionally the board becomes more compatible with pre-existing range of accessories and peripherals that are designed for the RPi such as cameras, displays, sensors as well as allowing someone with a CEDB board to look for help on issues with the RPi community thus benefiting from a larger knowledge base. Looking at the power draw of the CEBD Table 4. It is considerably less than a comparable machine such as the Jetson Nano using CUDA cores, this is due to the high efficiency seen with the Edge TPU. This is a major advantage to consider when looking at adding such a device to a drone running on batteries. The CEBD runs on a 5v 3A DC power supply when using the Edge TPUs. With CEBD being release in 2019 the number of papers containing the board are limited and no relevant examples are present, current chip shortages may be partly responsible also as it is very difficult to source a CEBD at the time of writing.

2.5.4 Nvidia Jetson Nano

Like the Coral Edge TPU Dev Board the Nvidia Jetson Nano Figure 15, [55], is specifically designed for machine learning applications, the main difference between being the inclusion of CUDA cores contained in the GPU of the Jetson Nano. The advantage here is the diverse workloads that can be taken on by such cores when compared to TPUs in the CEBD or accelerators. CUDA cores are designed to accelerate parallel computing tasks and can excite many tasks simultaneously. Applications range from simulations, video game graphics, and excel in parallel processing workloads such as deep learning and image processing. Having a Graphics processing unit that contains a mix of GPUs (General processing units), VPUs (Video processing units), TMUs (Texture Mapping Units) are useful for more general workloads and can be found in home computer graphics cards. When it comes to real time processing of images, a lower latency allows for higher frame rate cameras to be implemented, where a camera feed for 30FPS, the maximum latency is 33.3ms between frames, with faster cameras at 90 FPS the latency limit drops to 11.1ms, thus with better resolution and framerate the processing time allowed for real time solutions decreases requiring more streamlined and efficient processing solutions. To decrease the latency, it is best to maximise the efficiency of the task requiring the largest workload, in CNNs the maximum workload is found in the Convolution[56]. With the use of cuDNN (CUDA Deep Neural Network library) these heavy workloads can be optimised. Some major advantages of the Jetson Nano are its already widely tested use in robotics applications, with many successful published works regarding AGVs and UAVs. In work published by [57] a Jetson Nano mounted to a Tarot Quadcopter, using SSD MobileNet framework achieved 26 FPS with a mAP% of 92.7, for the use of detecting forest fires. This is a good example of a near perfect real time detection implementation. [58] utilised a NVIDIA Jetson Nano for Kiwi fruit classification, with a FPS of 30 being reaching, and a detection accuracy mAP of 97.79% further showing the suitability of the microprocessor for such applications. Other examples of a successful object detection implementations in the orchard realm via a jetson nano can be seen by [59],[60],[61] and many more previously mentioned. The NVIDA Jetson Nano is a flexible platform able to run both machine vision and a range of neural network architectures and traditional algorithms with a high efficiency and throughput. With its low 5-15w power consumption and small size it is an ideal good choice for developers looking for a versatile platform. Current availability of the Jetson Nano is also limited, but due to previous projects I have acquired one some time ago making availability a non-issue for this device.

2.6.0 Literature Summary

The literature review has extensively covered important topics directly related to the scope of this project, covering four main aspects namely Multi robot coordination techniques, existing Robot platforms, common fruit detection methods, sensors used for fruit detection & collision avoidance as well as appropriate processing hardware. The research into existing robot platforms highlighted techniques that have been proven to work, such as the common use of a Pixhawk flight controller to allow for an easier flight control and tuning of a custom drone as well as the ability to support autonomous navigation with ease. The chapter also reinforced ideas later touched upon such as the use of gimbal cameras, NVIDIA based microprocessors and CNNs and computer vision used for object detection thus giving a good understanding for the importance of later sections covered.

The next sections compared various techniques critical for the implementation of object detection, highlighting accuracy advantages that come with CNNs, and their draw back on the need for high computational power, when compared to other computer vision and supervised machine learning techniques. The widespread use of CNNs due to their advantages is seen reflected in the vast number of papers implementing them, including the amount of research done towards CNN optimisation further highlighting confidence in the field. When discussing the necessary hardware for object detection and collision avoidance it became clear that many researchers have opted to use a combination of RGBD cameras and sensor fusion of a secondary sensor and to assist with a better mAP% / accuracy and collision avoidance. Although the number of research into the use of RGBD cameras was significantly high, the main advantage of a RGBD camera seems to be when the addition of environment mapping is required via a SLAM technique on a AGV. RGBD cameras are shown to be useful for object detection but when compared to techniques using only a RGB camera the increase in mAP% does not stand out.

Finally, the “brain” of the robotic system was discussed due to its obvious importance. When looking at the most current microprocessors used in the hobbyist / researcher field today all three microprocessors compared very closely with one another when the addition of Machine learning Accelerators was taken into consideration. The Raspberry Pi, Jetson Nano and Google Coral Edge TPU Dev board are capable of real time object detection when the appropriate detection algorithm is chosen for board in question, or in the case of the RPi the TPU Accelerator is used to assist with inference bringing it closer to pa with the other two boards when using a CNN. Current availability due to chip shortages will thus be a main factor to consider for this project.

Chapter 3

Drone Research Platform

3.0.0 Chapter Overview

This chapter discusses development of the UAV robotic research platform used for capturing orchard video data Figure 17. The design has been chosen to be affordable, durable, re-creatable and can be used in a variety of different kiwi orchards. The mechanical, electrical and software systems used are discussed. Initial mechanical and electrical testing of the UAV with and out the integration of the Kiwifruit detection hardware system, is discussed in this section. Challenges are addressed and the methodology used for in orchard kiwifruit data acquisition is described.

Figure 17: Drone Platform

3.0.1 Research Platform

The research UAV aims to use readily available and affordable components to allow for future replication, repeatability, and further research development of such technology in later research. To achieve these goals well-established hardware and software is used, and when custom parts are needed additive manufacturing/3D printing is utilised.

To properly tune and develop the system, one must expect accident and crashes to occur, so the UAV is designed to be robust as possible given the restraints. This calls for the use of carbon fibre and engineering grade 3d filaments. For autonomous movement through orchards the drone must also be capable of accurate GPS navigation and collision avoidance. For ease of use a real time processing approach is ideal, with an output easily visible for users to read basic information critical for orchard yield estimation, without the need for complicated post processing that can't be expected to be performed by farmers.

3.0.2 System Requirements

Creating a system that can exceed the minimum requirements of a research platform and can instead already be used for useful data acquisition of a kiwi farm is a much better goal to aim for, as it pushes the design closer to that of a commercial grade application thus furthering the usefulness of such research for later analysis. Thus, the system requirements are that the drone must be able to perform orchard yield estimation, yield mapping, undergo orchard navigation via a pre-determined path, have a robust collision avoidance, and have a flight time capable of navigating an entire orchard, and be designed with the ease of use in mind such that a regular farmer can perform these tasks on their own.

3.1.0 Mechanical System

The Drone System Figure 17 is a HexaCopter based on a second hand commercially available Tarot 680 carbon fibre platform. The Drone consists of five levels, the lowest level holds the LiPo batteries, GoPro3+ mounted to a dual axis Gyro and two TFMini-s mm-Wave radars for obstacle avoidance. The second level houses the foldable propeller arms, telemetry, and GPS Mount. The third level has the most accessible surface area and holds many electronic modules such as the PDB, FPV camera, Secondary GPS, radio receiver, microprocessor

voltage regulator and more. The fourth level is where the Jetson nano is mounted, and above, on the fifth level is where the Pixhawk Flight controller and LCD is located such to have minimal EMI (Electromagnetic interference) and provide a better centre of mass / gravity.

To allow for such a setup several mounts were created in solid works and FDM printed out of appropriate materials. Again, from the bottom up, two battery spacers were designed such that a 6000mah and 16000mah 4s LiPo batteries could be used as not to require re-calibration when switching between batteries due to a change in COM (Centre of Mass). On the second level a new landing gear brace was designed as the second-hand drone had seen some damage and the braces had been replaced by aluminium sheets. Additionally, a taller GPS mount was created due to high EMI. On the third level the battery and associated low power cut off circuit powering the jetson nano is used, and a suitable cover is designed such to keep it safe during a potential crash. Between the 3rd and 4th level, a mount to hold the NVIDIA Jetson Nano was designed as to keep it away from the PDB. And a 5th level mount to hold the Pixhawk Flight controller and LCD screen was designed.

3.1.1 System Layout and Overview

The current and final UAV research platform has gone through many stages of development and redevelopment, to arrive at the current point. In its final design review Figure 17 the UAV features, a full carbon fibre folding tarot-680 frame, Turnigy Multi-star 3508-640Kv 14 Pole Multi-Rotor motors, dual cameras, one adjusted for an ideal FPV (First Person View), the other a GoPro-Hero3+ Silver, 2-axis gyro, Matek PDB, Pixhawk 2.4.8, Cam-Link, ESCs. The UAV research platform has gone through several stages of redesign to become a more stable, more durable, modifiable, and functional drone system. Figure 25 show the final configuration which was used to test the CNN detection system within the KIWI orchards. Figure 18, shows the setup used for under orchard test flights.

Figure 18: Prior drone Platform iteration.

The difference between the final drone platform and the prior iteration Figure 18 is the fixing of some stability and obstacle avoidance issues. Namely the difference lies in the layout of components, and addition of obstacle avoidance systems. In the prior iteration the five distinct levels also existed as size is a restraint, most of the available space is found vertically. This caused inherent stability issues, although this could also have been overcome with a wider drone, although this would restrict the number of orchards the drone can operate under.

In the prior iteration the bottom level provided space for a S4 Li-po battery holder, and mounts for the GoPro gyro/ GoPro camera. Mounting the batteries and other heavy equipment on the bottom level is a relatively common design choice for drones except for some FPV drones as it allows for more stabilised flight. The second level housed the six foldable carbon fibre arms, with ESC wires running through the inside of the arms, with additional space to spare the FPV radio telemetry module and Pixhawk communications is mounted towards the front of the second level. The third level is the most accessible and so contained most of the electronic components. These include the FPV camera mounted to the front, secondary GPS module a Camlink module to allow for real time video capture from the GoPro, Radio-link 9-channel receiver, buzzer, and a low power cut-off, 5v5a power supply for the Jetson Nano. The fourth level housed the Pixhawk 2.4.8. Finally on the top the Jetson Nano and a 16x2 LCD screen for real time detection outputs were mounted. The GPS stand is mounted to the lowest level, but the GPS sits above the rest of the electronics.

The main differences between these two designs were the swapping of the Pixhawk and Jetson Nanos positions, as well as the addition of the forward and downward facing TFMini-s mm-Wave radars. These additions have helped improve flight stability but were not necessary to compete testing of the real time object detection system.

3.1.1 Electronic Hardware Components

The final iteration of the drone comprises the electronic components listed in Table 5. Programming was carried out using the NVIDIA Jetson Nano and the Pixhawk 2.4.8, both of which are connected to the other hardware components on the drone.

Table 5:Hardware components in final drone iteration.

Pixhawk 2.4.8	5.8G Gimbal transmitter	4S 16000mAh LiPo
Matek PDB XT60 12v	FPV camera	4s 6000mAh LiPo
APM2.8 power module	M8N GPS and compass	3S 2000mAh 10c discharge
Anti-vibration board	Secondary GPS	Turnigy 3508-640KV motor
FPV radio telemetry module	Go-Pro Hero 3+	Platinum Pro 30A ESC
Eachine 7” 5.8GHz FPV Monitor	Tarot 2-Axis Brushless Gimbal	14X4.7 Slowfly propellers
Safety Buzzer	Camlink 4K	Nvidia Jetson Nano
Safety Switch	5V5A voltage regulator	Secondary altimeter
Radio Link AT9 controller	HDMI to 90ºMicro HDMI ribbon	12x2 LCD display
2* TF Mini-s mm radar	I2C interface	barometer

3.1.2 Additive Manufacturing

Additive manufacturing, also known as 3D printing, is a process that involves creating a physical object from a digital model. It uses a variety of techniques, including extrusion, powder bed fusion, and vat photopolymerization. The process is used to create a wide range of products and parts, including prototypes, tools, and end-use parts. It can be used to produce parts with complex geometries, which would be difficult or impossible to make using traditional manufacturing techniques. Additive manufacturing can offer significant advantages over traditional manufacturing, including faster production times, lower costs, and improved product performance. For this project it was very useful for producing prototypes and light weight end use parts from a low-cost standpoint.

3.1.2.0 PLA+ Filament Properties

The most popular filament among 3D printing enthusiasts is PLA+, a thermoplastic polymer made from renewable resources like corn starch or sugar cane that is simple to use and produces a smooth finish. In addition to being non-toxic and one of the most reasonably priced 3D printing materials, it is also safe to use at home. PLA+ is also more resilient than conventional PLA, allowing for the printing of bigger and more complicated things without the worry of warping or cracking. Lastly, PLA+ is the best filament for producing delicate and detailed prints since it has a low shrinking rate and strong layer adhesion. This is ideal for the printing of the battery spacers and some parts used to house electronics as the parts are designed to be as thin as possible such to keep the weight down. PLA+ has a lower density than PETG, making it ideal for parts that are not directly exposed to the sun on the drone platform. The Print Properties [62] are shown below in Table 6.

Table 6: PLA + Properties

PLA+	
Elastic Modulus MPa	3500
Poisson's Ratio	0.36
Shear Modulus MPa	1287
Mass Density g/cm^3	1.252
Tensile Strength MPa	59
Flexural Strength MPa	87
Flexural Modulus MPa	3600
Yield Strength MPa	70
Elongation at break	7%

3.1.2.1 PETG Fillament Properties

PETG (Polyethylene terephthalate glycol-modified) is a thermoplastic filament material used in 3D printing. It is a popular 3D printing material because of its combination of excellent physical and chemical properties [63] and is ideal for the use in outdoor parts that will experience high levels of UV light and moisture [64] such as those found in New Zealand.

Table 7: PETG properties

Physical Properties	Chemical Properties
High strength and stiffness	High chemical resistance
Good impact and flexural strength	Low toxicity
High heat deflection temperature	Low odour
Good chemical resistance	High biocompatibility
Low water absorption	Good thermal stability
High dimensional stability	Good flame retardance

Table 8: PETG Print Properties

PETG	
Elastic Modulus GPa	22
Poisson's Ratio	0.38
Shear Modulus GPa	1.275
Mass Density kg/m^3	1280
Tensile Strength MPa	53
Compressive Strength MPa	55
Yield Strength Pa	4.79e7
Elongation at break XY	6.8%
Elongation at break Z	1.3%

3.1.2.2 FDM / 3D printer overview

The 3D printer used to produce and test of components is an Original PRUSA Mini + kit. This printer was aquired for its high quality prints and low cost of entry from the prusa range, with a custom fillament run out sensor. The FDM printer is devided into the components listed in Table 9, depicted in Figure 19

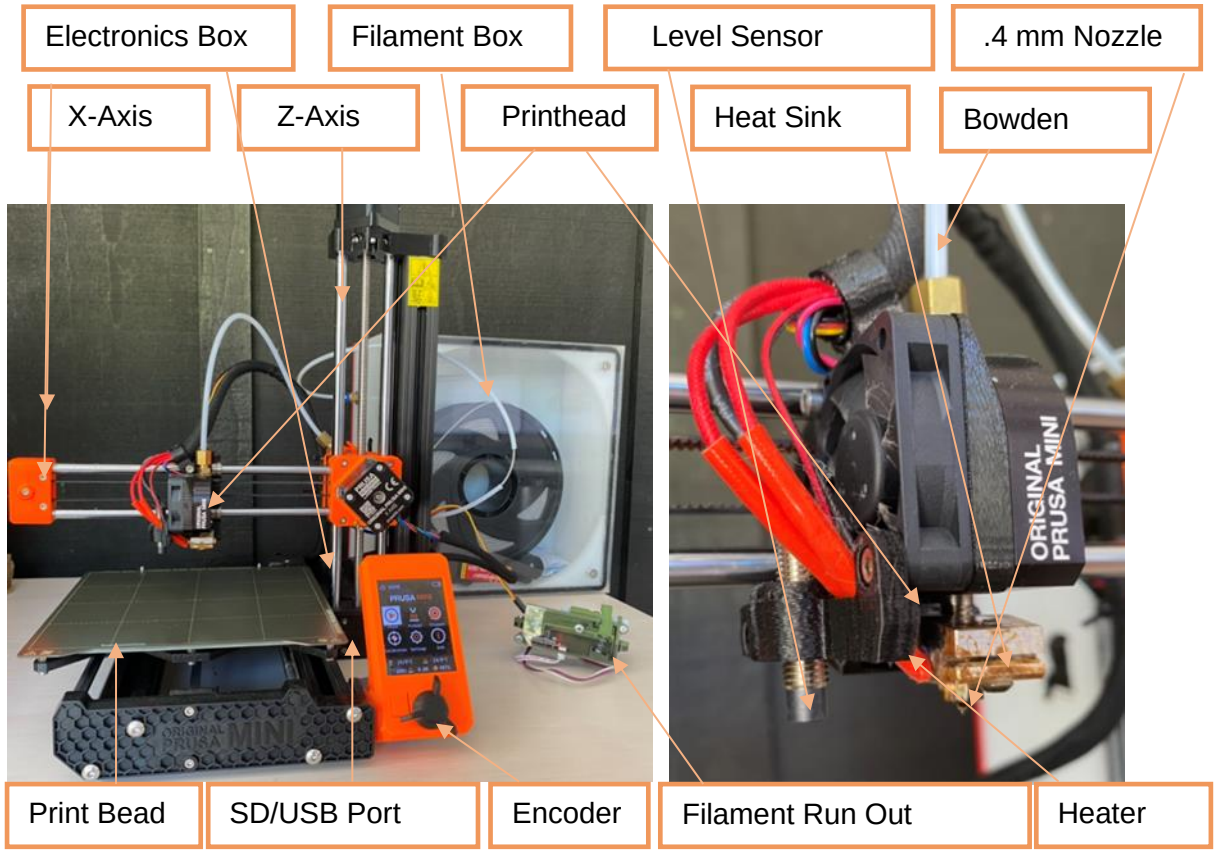


Figure 19: FDM Printer Details.

3.1.2.3 Printing Calibration

To get an ideal print quality with any filament the 3D printer must be calibrated so suit the individual printer. Using trial and error the settings were honed in. Adjusting the initial first layer print temperature to 235 °C then increasing to 245 °C for other layers gave good bed and layer adhesion when using PETG. Support contact was set to .3mm top and bottom to create a cleaner finish after removing supports, with the elephant's foot compensation set to .2mm to get dimensionally accurate edges on parts with no brim. more printing properties for this specific Prusa Mini printing PETG filament from Marvle3D can be seen in Table 10. Print Properties used for PLA + can be seen on Table 11.

Table 9: 3D Printer Components / Details.

Printer Part	**Part Description**
Print Head	Houses the level sensor, nozzle, heater core, Bowden tube.
X-Axis	Is the printer's X-axis. Max length of 180mm.
Y-Axis	Is the printer's Y-axis. Max depth of 180mm.
Z-Axis	Is the printer's Z-axis. Max hight of 180mm.
Filament Box	This box keeps the hydroscopic filament dry, to ensure print quality.
Electronics Box	Houses the motherboard & stepper drivers.
Bowden Tube	Provides a reactive force on the filament, to maintain feed pressure.
Heat Sink Fan	Regulates printing temperature and cools filament to prevent drooping.
Level Sensor	Measures the hight and warp of the bed to set the nozzle at a predefined hight before each print.
.4 mm Nozzle	Standard .4mm brass nozzle with .4mm print diameter.
Heating Element	Sets the temperature of heater core to pre-melt the filament for extrusion.
Filament Run Out Sensor	Sensor to pause prints when filament runs out. (Currently un-plugged)
Encoder	Allows navigation of the LCD / settings.
SD/USB Port	Allows the user to upload G-code from printing software.
Print Bead	A heated, flexible steel bed to adhere the print and release after cooling.

Table 11: 3D Printing Settings for PLA + on Prusa Mini

Printer Setting PLA +	**Value**
Infill %	10%
Solid layers Bottom	3
Solid layer Top	3
Overhand threshold	Draw on supports
Infill Pattern	Gyroid
Layer Hight	.2
Perimeters	4
Minimum Shell Thickness	.7mm
Bed Temperature First Layer	60 °C
Bed Temperature Other Layer	60 °C
Nozzle Temperature First Layer	215 °C
Nozzle Temperature Other Layer	210 °C
X & Y Movement Speed Perimeters	40 mm/s
Infill Speed	80 mm/s
Bridges Speed	25mm/s

Table 10: 3D Printing Settings for PETG on Prusa Mini

Printer Setting PETG	**Value**
Infill %	20%
Solid layers Bottom	4
Solid layer Top	4
Overhand threshold	50%
Infill Pattern	Gyroid
Layer Hight	.2
Perimeters	4
Minimum Shell Thickness	.7mm
Bed Temperature First Layer	85 °C
Bed Temperature Other Layer	90 °C
Nozzle Temperature First Layer	235 °C
Nozzle Temperature Other Layer	245 °C
X & Y Movement Speed Perimeters	40 mm/s
Infill Speed	80 mm/s
Bridges Speed	25mm/s

3.1.2.4 Manufacture Software (SOLIDWORKS & PrusaSlicer)

Software selection has a large impact on part quality and manufacture time [65], with PrusaSlicer being one of the top and open source slicers on the market, as well as being specifically designed to work well with the PRUSA Mini, it was an obvious choice. To create the G-code used on the Prusa Mini FDM Printer, the 3D models were first created on SOLIDWORKS. As PrusaSlicer utilises STL (Standard Triangle Language) to create G-code, thus the SOLIDWORKS models were exported in a high resolution STL format, as the quality of the STL files determines the dimensional accuracy of the printed part. once STL files are imported into PrusaSlicer Figure 20 the G-code generation can begin, this is called slicing.

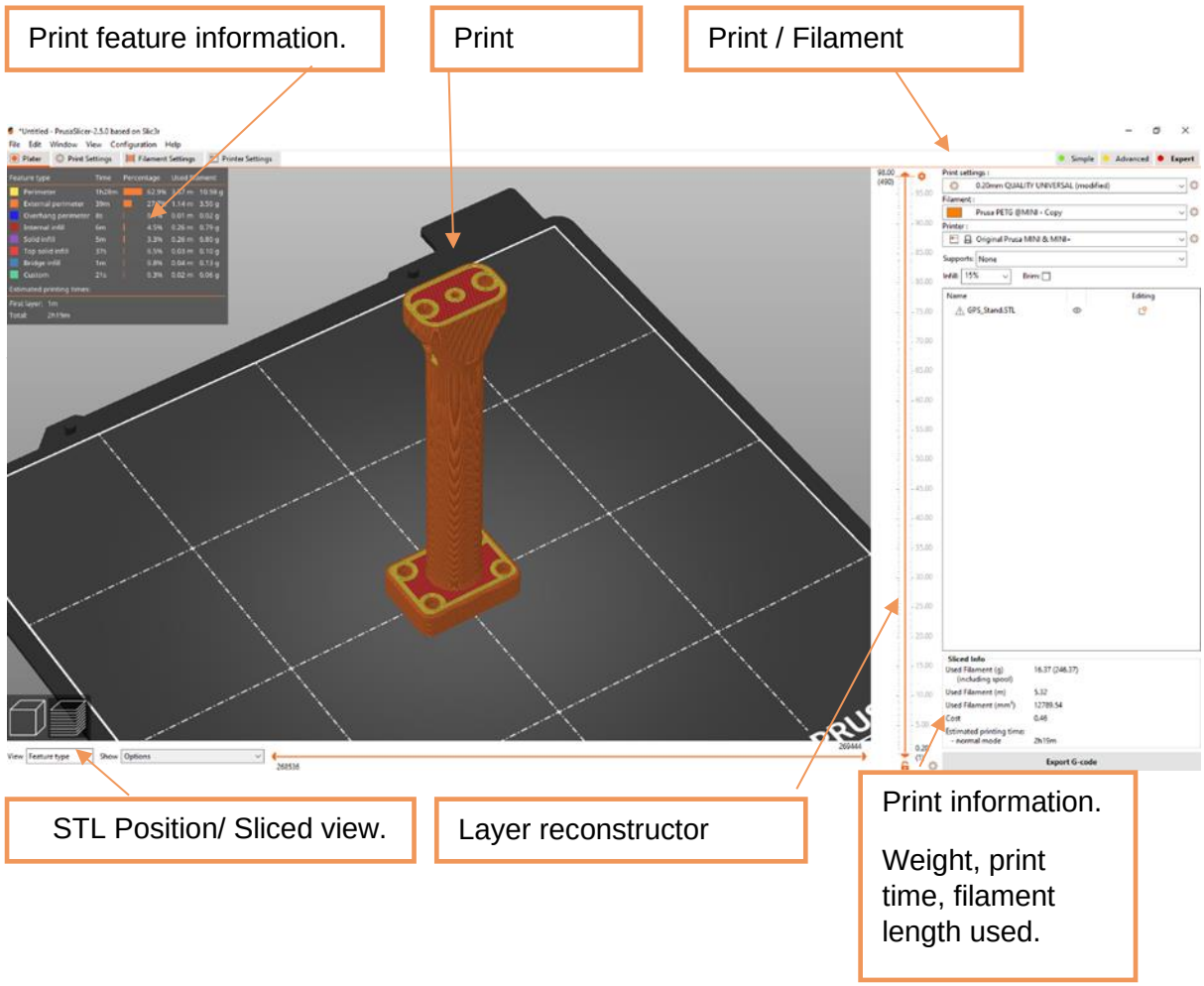


Figure 20: PrusaSlicer user interface.

UAV Platform Development

3.2.1 Flight Controller, Microprocessor and PDB Mounting.

Utilising the Pre-built Tarot-680 frame comes with many advantages when compared to creating a custom frame, namely having most components built from carbon fibre allows for a light weight and rigid drone. The Tarot-680 frame also has many extra mounting holes that can be used for further customisation. When adding many more components than what a standard drone has this comes in very useful. The addition of the NVIDIA Jetson Nano microprocessor is the largest and heaviest non-standard component that need to be added, the weight distribution, ease of accessibility and EMI (Electro Magnetic Interference) must be considered when designing the mounting of all components. This is more exaggerated with the microprocessor due to its high current use. Initially only the ease of accessibility was considered such that various models and programming changes could be made. The previous iterations of the drone saw the NVIDIA Jetson Nano mounted on the 5th level, Figure 21, while using the smaller 6000mah LiPo battery with no spacer mount. Subsequently this resulted in a shaky flight due to a COM being above the rotor plane, additionally carbon fibre blades were used during these test flights. The rigidity of carbon fibre blades reduces the ability for a large discrepancy of COM from the static rotor plane, as the rigid blades create a very exact rotor plane, with little flexibility in the hight of the driven rotor plane. As shown in [66] an unstable pair of nodes are generated when the COM moves above the rotor plane, causing vibrations Figure 21. A second factor contributing to unstable flight during testing of the entire system (Including the real time kiwifruit detection), was EMI effecting the GPS and compass.

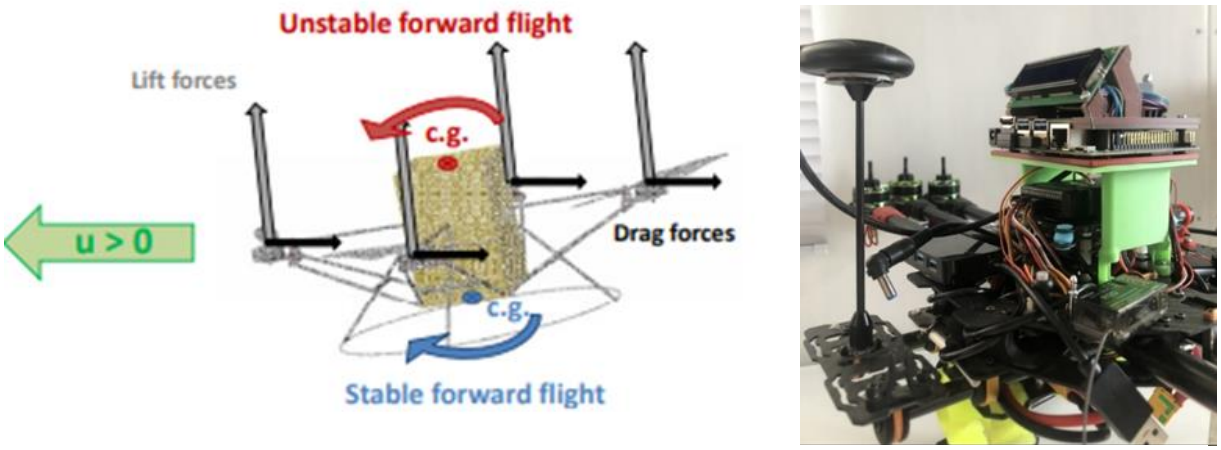


Figure 21: Impact of Centre of Gravity during flight [Left]. Early UAV platform iteration [Right].

Pierre-Jean Bristeau, P.M., Erwan Salaün, Nicolas Petit, *The role of propeller aerodynamics in the model of a quadrotor UAV*. European Control Conference 2009, 2009: p. 683-688.

These fundamental issues are discussed and resolved in later iterations of the UAV platform. The mounting of the PDB, Jetson Nano and Pixhawk flight controller is spread over three mounting platfourms Figure 22. The lower level PDB mount is designed to have sturdy mounting points, where the lower four “pillars” are directly screwed into the carbonfibre drone frame. Printed out of PLA to keep weight low. Directly to these “pillars” the second level where the Jetson nano frame are screwed in using M3 heat set thredded brass insirts. The jetson nano then has a protective cover to prevent damage to electronics from above, also printed from PLA. The third level is screwed directly into and over the Jetson Nano hetasink, that has had M3 holes tapped in. This mount has a hole to prevent overheating of the Jetson Nano, and is printed in PETG as it is sun exposed and more vountable to damage during crashes (this was discovered the hard way)Figure 22. Using M3 heat set threaded insirts again the Pixhawk fligt controller dampner is screwed in to this mount, where the Pixhawk is mounted to the damper via shock absorbing double sided foam tape as recommended by pixhawk manual. The 2x16 LCD adds regidity to the frame holding it up completing the stack of electronics.

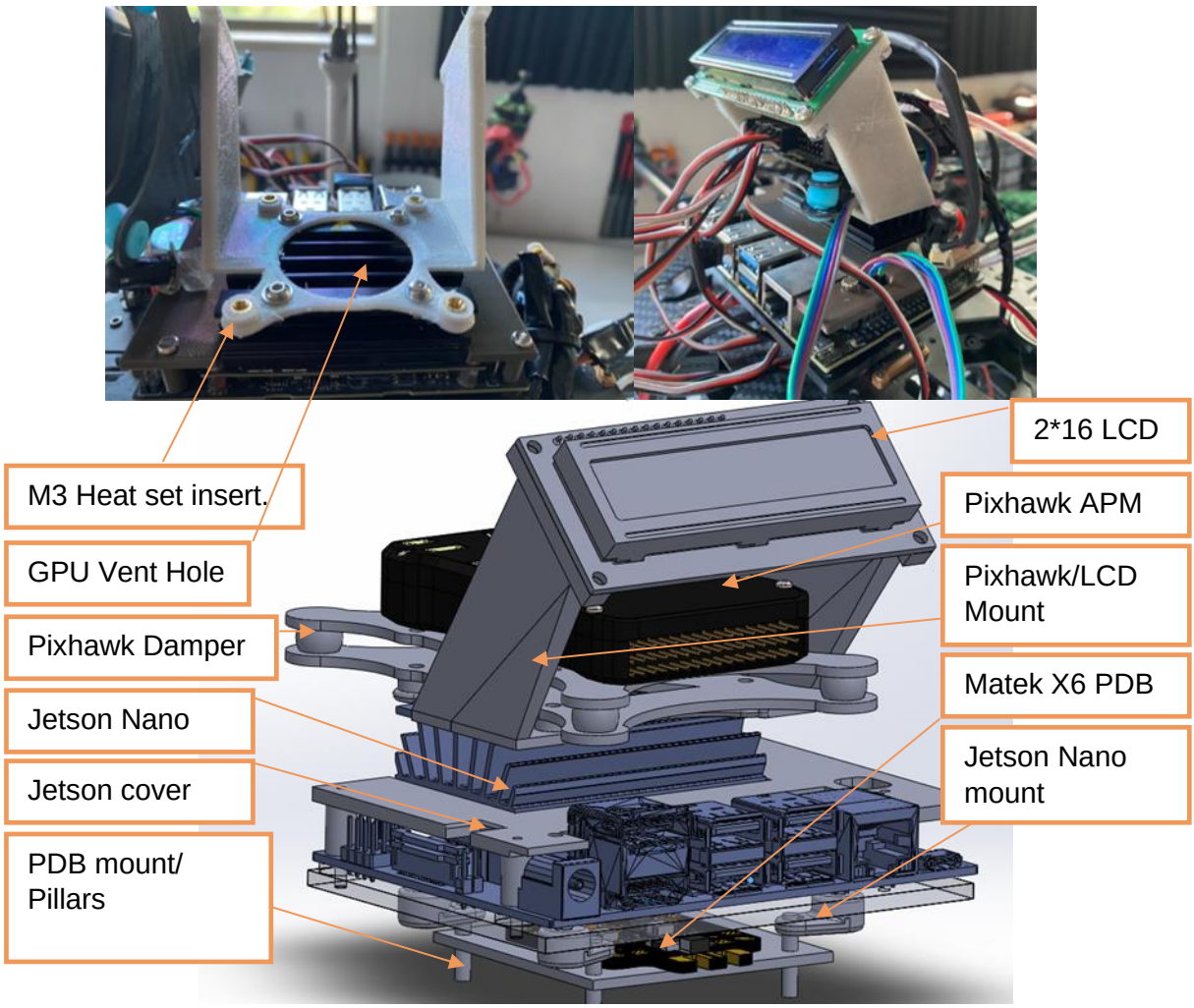


Figure 22:Pixhawk & LCD mount [Top] Drone Mounts CAD model [Bottom].

3.2.2 Battery Mount

A modular battery holder was designed and 3D printed, using PLA for its low weight/ density and ease of printing Figure 23. This design is intended to compensate for the use of a smaller 6000mAh LiPo battery, or when in the shorter configuration, to hold the larger 16000 mAh LiPo battery. After several test flights, additional bumps were added to the holder to prevent the battery from moving during flight, causing the COM (Centre of Mass) to change.

The battery holder is designed such that the COM does not change regardless of what battery type is used, rather only the total weight of the Drone changes. This massively improves stability/ removes the need to recalibrate the autopilot after changing batteries and is only done to save money and time. Buying two identical batteries, is recommended.

The battery spacers were designed by first measuring the centre of mass of the drone, by hanging it from a string (without batteries), from multiple angles. The COM is .043m from the bottom of battery mount of the drone. The COM for the batteries, small and large respectively .022m & .037m from the mounting edge. Their masses are .588kg & 1.318kg respectively. The large battery spacer is .02m in height. Thus, the spacer for the smaller battery is .104m derived from the following equations.

$$x = \frac{m_2 d}{m+m_2} \quad x = \frac{(1.318 * .100)}{(3.112+1.318)} \; d = \frac{mx}{m_2} + \frac{x}{1} \quad d = \frac{3.112 \times .03}{.588} + \frac{.03}{1} \quad (3.2.2.1)$$

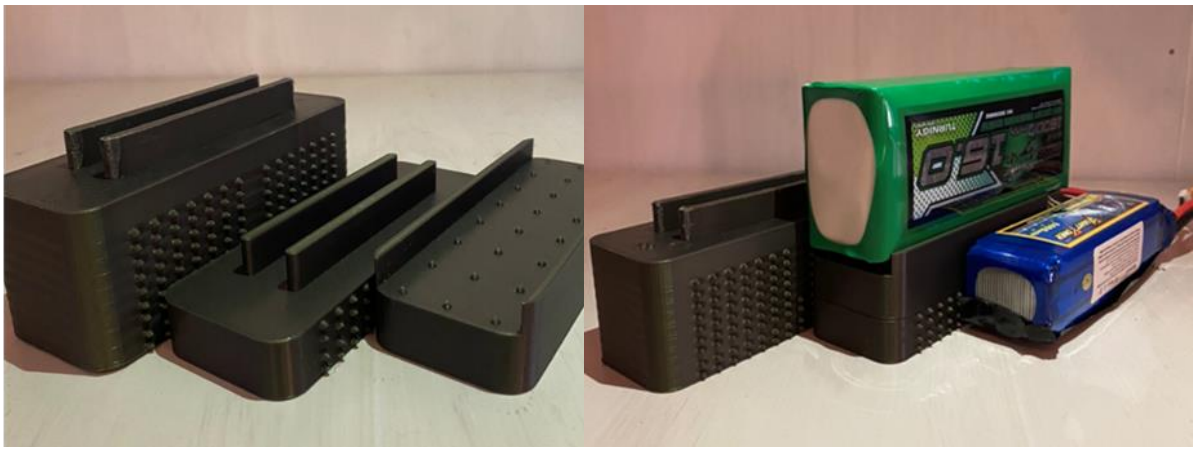


Figure 23: Modular Battery Spacer Holding 16000mah LiPo next to 6000mah LiPo.

3.2.3 GPS/Compass Mount

The GPS mount that came with the drone may have been well suited for standard drone set up, but with the additional Jetson Nano, and associated electronics the EMI (Electro Magnetic Interference) induced onto the GPS was considerably higher, so a taller mount was designed. Generally, the ESC's are mounted close to the PDB, with the signal wires stretching to the motors, this is done to reduce the EMI induced [67], but with space constraints, and design of the Tarot-680 frame such mounting was not possible, this is likely the reason a standard GPS mount did not suffice.

To keep the weight of the drone as low as possible, the GPS bottom bracket was designed with the use of as little material as possible and 3D printed out of PLA, for the mast, the inside was hollowed out and printed out of PETG, as it would be a sun exposed part of the drone, as well as having a higher impact tolerance, making it more likely to survive in a crash then if it was made out of PLA [68].

To test the validity of the printed mount an EMI test was conducted, by inverting the propellers, and rotating them one position around the frame, and loading the motors at max thrust/ current, thus pushing the drone down into the ground when powered on, allowing a test at full load.

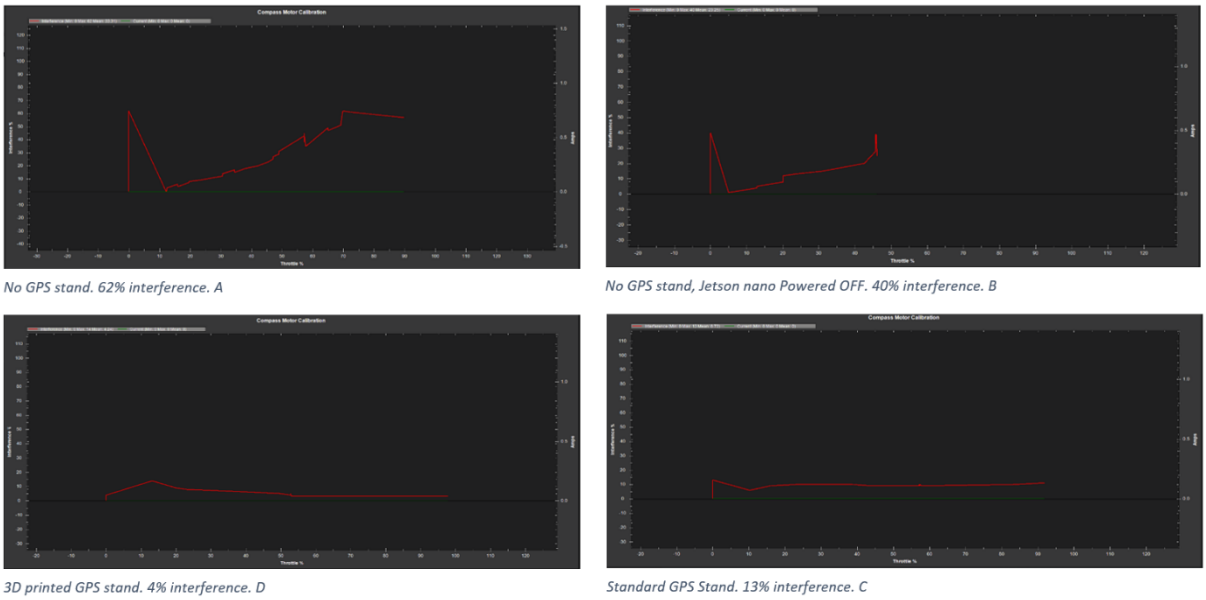


Figure 24. Electromagnetic interference in GPS, various mounts.

3.2.3.0 EMI Interference Levels.

As seen in Figure 24 the internal EMI in the GPS is very high with no stand, reaching 62% when the rate of change of throttle/current is high, as well as the proximity of the 5v5a Jetson Nano power cable. With the Jetson powered off the same test was conducted, and the interference dropped to 40%, this is likely only because the GPS is close to the 5V5A voltage regulator that powers the jetson.

The GPS was then mounted to the original stand and again the EMI was tested and found to have 13% interference, This is considered acceptable [69]. But for autonomous flight the steadier the drone can be the better and is directly correlated to EMI. In graph D, finally the EMI is very low at 4%, due to the larger distance between GPS and other current carrying wires. Other methods for lowering EMI were considered such as using a shielding cover [62], but this would require a minimum of adding grounded aluminium foil wrapped around all wires capable of affecting the EMI, and this would add considerably more weight to the drone. Safe GPS and Compass distances from small metal objects and electronics such as phones is set at 15cm by [69] further reinforcing the need for the addition of the GPS stand. The Final iteration of the GPS mount that resulted in minimum EMI, with Jetson Nano mounted Under Pixhawk Flight controller, at 16cm from the LCD.

Figure 25: GPS / compass mount after accounting for EMI

3.2.4.0 ESC and Prop.

An ESC (Electronic Speed Controller) is a necessary component to regulate the speed of motors in drones. This is done by varying the power delivered to the motor, which can be used to control the speed, direction, and braking of the motors. Not all ESCs are the same, as the way the PWM (Pulse Width Modulation) signal is sent to the motor affects the thrust produced at a certain RPM. Initially, the ESC that came with the drone was used for testing, but due to the heavier weight of the drone, modifications were necessary to prevent the ESC or motor from burning out from an extended flight or overheating. Therefore, selecting the correct ESC, motor, and propeller combination is essential to achieve optimal thrust and efficiency in a drone. The ESC acts as the “brain” of the system, sending power to the motor, the motor determines how much power can be delivered to the propeller, and the propeller determines how much, and how fast air can be moved at a certain RPM, resulting in the thrust. Thus, even when consuming the same amount of power, a different combination of these three items can have large differences in thrust at varying loads.

The selected components can be seen on Table 12, where each component was tested against one another for thrust output at selected RPMs, and a constant voltage. The resulting combination will be able to extend the drones battery life (potentially allowing for the use of a lighter weight battery), lower noise levels, and increased longevity of the electrical components involved.

Table 12:Thrust Optimization Components

Component	Price (per item)
Turnigy Multistar 3508-640Kv 14 pole motor	$52.74 NZD
Platinum pro 30amp ESC	$49 NZD
TURNIGY 40amp ESC	$40.47 NZD
HQProp Multi-Rotor 12x4.5 propeller	$9.6 NZD (pair)
Quantum Carbon Fibre 13x4 propeller	$20.93 NZD (pair)
GWS Style Slowfly 14x4.7 Propeller	$7.82 NZD (pair)

Figure 26: Thrust Optimization Components.

3.2.4.1 Thrust Optimisation Results.

To Optimise thrust using the various components and to lower the required load on the motors and ESC to produces the required lift, a test rig was created that can incorporate all combinations of the Propellers, ESCs and Motor, thus testing the efficiency of each combination against one another. Rather than testing each individual component separately. The test rig consists of a HX711 Load Cell amplifier circuit board, a 5KG rated loadcell, Arduino UNO microcontroller and a custom motor mount Figure 27.

This test rig allows the accurate measurement of thrust produced at various RPMs, produced by the 6 combinations of components being tested. The Load Cell contains two strain gauges and two 120Ω resistors arranged in a Wheatstone Bridge configuration, secured to a rigid aluminium frame. The strain gauges return a resistance value dependant on the amount of deformation/strain experienced by the circuit. The Loadcell is connected to the Load Cell AMP HX711 such to convert the signal from the Load Cell into a legible signal for processing on the Arduino UNO. The Load Cell Amp is thus connected to the Arduino UNO, running a custom script that allows both the reading of the loadcell and a PWM Duty Cycle input signal to be sent to the ESC. The Arduino UNO is therefor also connected to the ESC and PC. The ESC is powered by the 16000mah LiPo used in the drone, additionally the LiPo is not directly connected to the ESC but rather to the drone and the ESC is connected to the drone for a more accurate current supply making the test more accurate due to current drop experienced through the drone wires. As described the Loadcell returns an arbitrary resistance value dependant on the strain experienced, therefore a calibration measurement must be carried out to determine the conversion rate of Ohms to lift Force. This is done by first reading the resistance value (an arbitrary large number) given to the Arduino when the loadcell has only the motor mount attached. This value can be considered the null point where “no force” is experienced by the loadcell. Next a 100g weight is placed on the motor stand and the now changed arbitrary value is measured again. This new value corresponds to a force = to 100g. This ratio is calculated and repeated for 200g and 500g to gain an average value, concluding the calibration method for the loadcell. Each combination of components was tested at five intervals starting at the minimum Duty Cycle required to spin the blades sufficiently, to maximum RPM. The result from this experiment shows the best parts to use in combination are the Platinum Pro 30A ESC, 14X4.7 Slowfly propellers and Turnigy 3508-640KV motor, as listed in

Table 13.

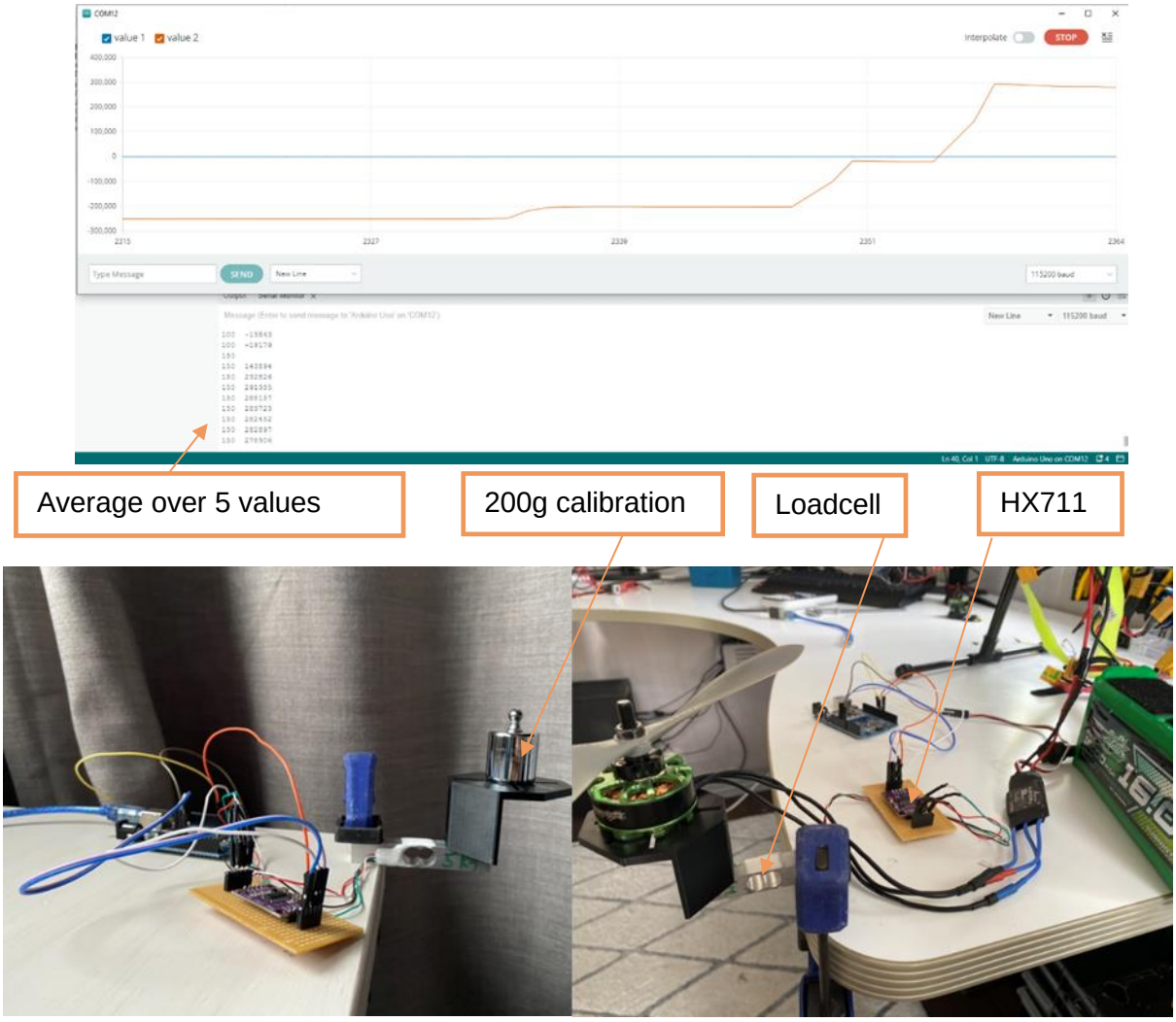


Figure 27: Loadcell strain VS PWM plot [Top], Test Rig calibration [Left], Test Setup [Right].

Table 13: Lift Force generated at various PWM Duty Cycles

PWM Dudy cycle	Black Prop Turnigy ESC	Carbon Prop Turnigy ESC	Silver Prop Turnigy ESC	**Silver Prop Platinum ESC**	Carbon Prop Platinum ESC	Black Prop Platinum ESC
Lift(g) 5% Dudy cycle	38.31	185.2	256.4	**18.9**	13.8	21.0
Lift(g) 25% Dudy cycle	196.2	723.9	870.6	**406.65**	223.5	234.4
Lift(g) 50% Dudy cycle	541.1	1087.5	1248.9	**1157.75**	809.7	804.7
Lift(g) 75% Dudy cycle	967.4	1057.9	1207.8	**1371.2**	1318.4	1338.3
Lift(g) 100% Dudy cycle	963.0	863.9	1159.6	**1378.8**	1341.1	1324.3

3.2.5 Landing Brace

The second-hand Tarot-680 frame had been damaged in the past, most notably on the landing gear mounts, which hand been replaced completely with an aluminium part Figure 28. While the repairs are adequate for casual flying of the drone, due to the added weight that the drone now must carry, any possible weight saving should be pursued.

The 2mm aluminium landing gear mount that was used to repair the drone weight 16 grams, with a total of 4 being used thus adding up to 64 grams. Although already a relatively light and strong repair, the part was hand cut and loose due to an incorrect/ inaccurate through hole size being used Figure 28, with further weight savings possible.

Using SOLIDWORKS the aluminium mounts were used as a template to understand where the mounting points must be positioned, a square mount was designed to be made from PETG. To further reduce the weight a topology study was conducted as suggested by [70], with best stiffness to weight ratio, and mass minimisation restraints. As seen in Figure 29 the resulting shape was 50% lighter than the original PETG square iteration, as well as being able to withstand a fully loaded landing at 2G, with a safety factor of 1.5. the resulting part weighs just 4.5 grams. A small handle was also added to allow easier lifting of the drone. Some precautions should be taken when performing a FEA analysis using 3D printing materials as the layer adhesions are not considered and can result in unaccounted for weak spots in the design, hence the 1.5* safety factor. The printed PETG part can be seen installed in Figure 29.

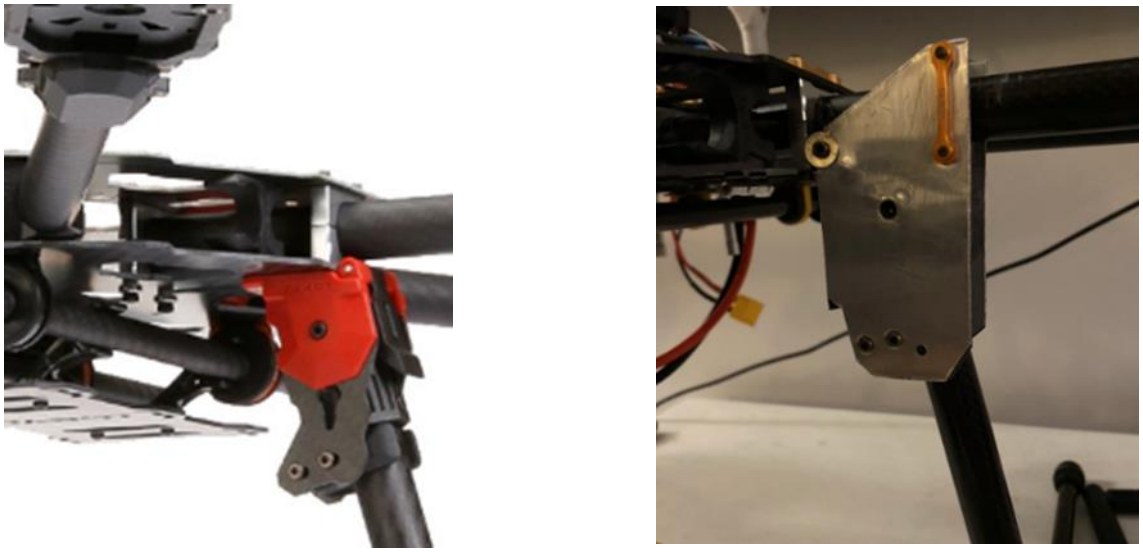


Figure 28: Original Tarot-680 Landing mount [Left], Old Aluminium landing gear mount [Right].

Team, H.S. *TAROT T960 HEXACOPTER KIT*. 2022; Available from: https://hobbystation.co.nz/tarot-t960-hexacopter-kit/

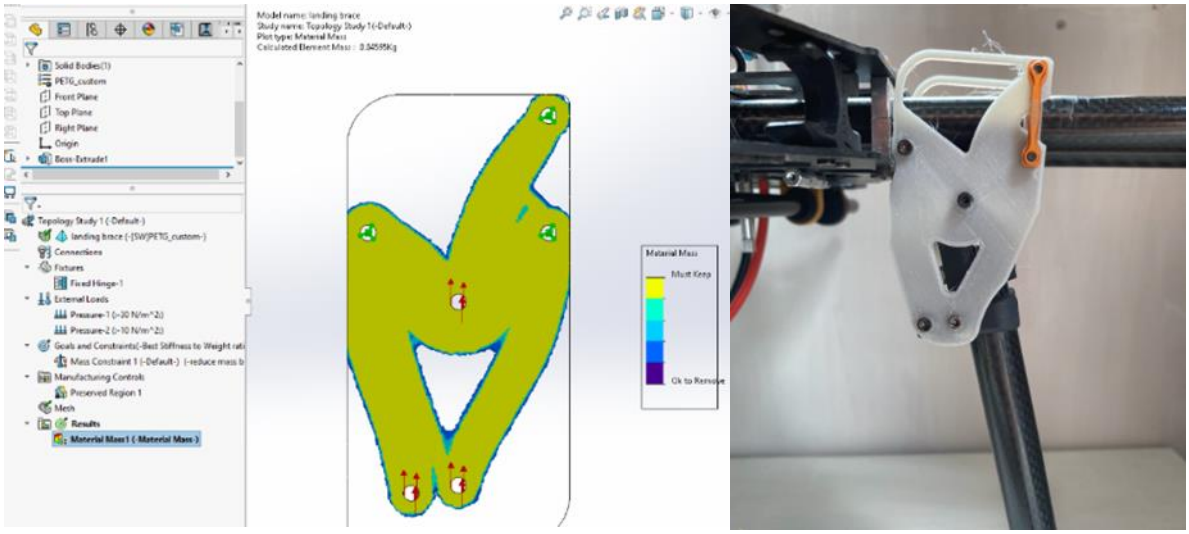


Figure 29: PETG Topology study [Left]. PETG Revised mount [Right].

3.2.6 Go-Pro Mounting

The GoPro HERO 3+ silver is mounted to the gyro in an upside-down fashion Figure 30, this is due to the micro-HDMI port being obstructed by the mounting mechanism when mounted the correct way around, with the lens being surrounded by the blue bracket. Mounting the GoPro upside down brings the lens further down and allows the micro-HDMI port to be accessed. Flipping the output image can be done in the GoPro settings via: menu>settings>capture settings>UP>ON>EXIT. The menu button is not obstructed by the mount in this configuration, although it is not required to be pressed at any time. The HDMI to micro-HDMI cable is a custom light weight ribbon cable with a 90^0 micro-HDMI port Figure 31. This is a requirement as a standard thick cable has too much weight and tension which prevents the gyro motors from keeping the GoPro at the desired angle, disrupting stability. The HDMI cable is then directly plugs into the Elgato CAM LINK video capture device registering the GoPro as a webcam on the Jetson Nano.

3.2.7 Experimental Flight

To validate the drone platform's ability to fly within a confined space, both vertically and laterally the drone is tested in the orchard. Due to the time of year, there are no kiwi fruits hanging from the orchard vines and so provide an extra 6 inches of vertical hight. Figure 66 show the drone in the orchards and detailed images Figure 30. With lateral flight being stable, but vertical flight is difficult to achieve at a stable altitude due to barometer readings being highly inaccurate bellow 20m elevation. To increase the test flights accuracy the full object detection system is mounted and turned on for further electronic testing. Although no object detection is taking place, provides good feedback for the progress of the project. (Some components such as the updated landing brace, GPS stand, Blades, Go-Pro cable, updates electronics stack etc were a result of this test flight and so are not in the images)

Figure 30: Early orchard test flight [Top], Upside down GoPro mount, without ribbon micro to standard HDMI cable [Left]. Rear of drone [Right].

Electronics

3.3.1 TF Mini-s mm Radar

After initial flight testing through the orchards, it became clear that using elevation data from the GPS and barometer is not sufficient to keep the drone at a stead hight, especially when accounting for the fact that many orchards are planted on a gradient. To fix this the TFMini-s mm-Wave radar is employed, with and Additional TFMini-s used for obstacle avoidance. Using sensor fusion for both TFMini's, one pointing down and the other forwards, a stable vertical flight as well as obstacle avoidance is achieved. To keep a safe distance from the top of the orchards, as well as any low hanging vines/ leaves a high of 1m is used. For the obstacle avoidance, the TFMini can sense up to a 12m range [43]. Due to outside conditions such as light which can introduce noise into the radar, a shorter distance of 6m is more-suitable, but still well within an acceptable range for this application. Lowering the sensing range to two meters, and a command to stop at a 1m distance from obstacles the drone can more accurately determine when the end of the orchard is reached and decreases power consumption by the sensor. Mounting the TFMini-s on the front of the drone was done via a custom PETG mount, that positions the sensor directly at the front of the drone, fastening slightly behind the cameras point of view. (It is slightly obscuring the FPV camera however)Figure 31. This mount incudes 2 grub screw locking points to prevent sliding back and forth on the rails. The TF Mini-s responsible for high sensing is mounted on the rear of the drone, under the GPS mount Figure 31. This is a simple flat PETG print with the appropriate mounting holes attached to a spare gyro mount. The programming & flashing of the TFMini-s was done with the help of the product manual which supports Pixhawk flight controller [43], and the supporting software available on Benewake website.

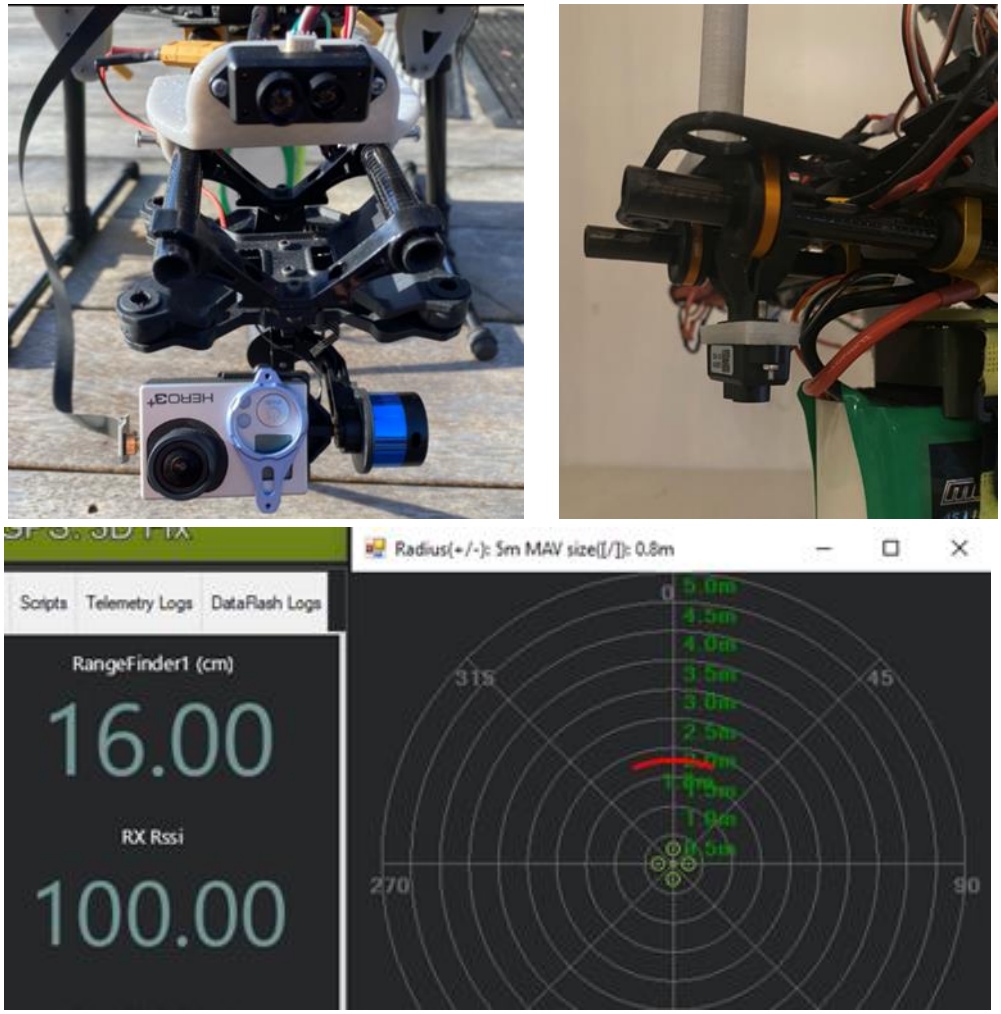


Figure 31: TFMini-s front obstacle avoidance [Top Left], TFMini-s Hight sensing (Rear) [Top Right], TFMini-s Hight estimation (Rangefinder1) and Obstacle avoidance tuned to 2m [Bottom Right], (sensing my hand at 1.8m).

3.3.2 5V5A Regulator

To power the Jetson nano a 5V5A power supply is required, unlike most electronics the current supply must be very close to the specified 5A or the jetson will not power on and seems to be a common issue among many users. Initially a 5V5A buck converter Circuit was purchased for the use on the drone, but it was not able to supply a stable current likely due to cheap components on the board. Additionally using the large LiPo to power the Jetson as well as the flight controller, additional electronics and the motors would have negatively affected the performance of the drone due to the high current draw. For these reasons, a custom 5V5A voltage regulator was designed and manufactured. Utilising voltage regulators is a cheap method of achieving the intended goal but come with the downside of being less efficient than other methods, with the drone being a test platform this is acceptable and within the scope of the project. Additional time was limited with Kiwi fruit being in season for only a few days before picking. Purchasing a new pre-made voltage regulator with low voltage cut off protection was not an option due to time restraints. A custom 5V5A low power cut off circuit to run off an additional onboard 2000mAh 3s LiPo, was designed with an inbuilt Seeeduino Xiao. The purpose of the circuit is to deliver an accurate and stable 5V5A supply to the NVIDIA Jetson Nano, where the low voltage cut off circuit prevents over discharge of the battery such that it can be safely recharged after use.

Table 14:5V5A Low voltage cut off Voltage regulator components.

2* LM1084 5V5A Regulator	OJE-SH-105HM,5V Relay	2* 10nF capacitors	2* 10µF capacitors
Seeeduino Xiao	1200 Ω through hole	110 Ω through hole	Screw Terminal 2 hole
Barrel Jack Male	Barrel Jack Female	BC337 NPN Transistor	20W Heat Sink

The Circuit is designed using Altium Designer, a leading PCB and electronics design software. With the intent of using an available PCB router, the design is kept simple, as only a single-sided PCB can be created from such a router. The use of a 5V5A capable Lm1084 linear voltage regulator was chosen due to ease of accessibility, but it does come with some drawbacks. Ideally a Buck converter or premade power supply module would be used, or a switching regulator to increase the efficiency of the system. The use of a 3S LiPo battery was chosen as a 7.4V 2S LiPo with sufficient mAh, is hard to find.

Using a 11.1V 3S LiPo is not ideal when combined with a Linear Voltage regulator as the relative power loss will be large and will require a large heatsink adding more weight to the overall design. Nevertheless, the circuit was designed due to time constraints, and proved successful for the use of testing, but should by no means, be used on a commercial version of such a drone. The Schematics and traces can be seen in Figure 32.

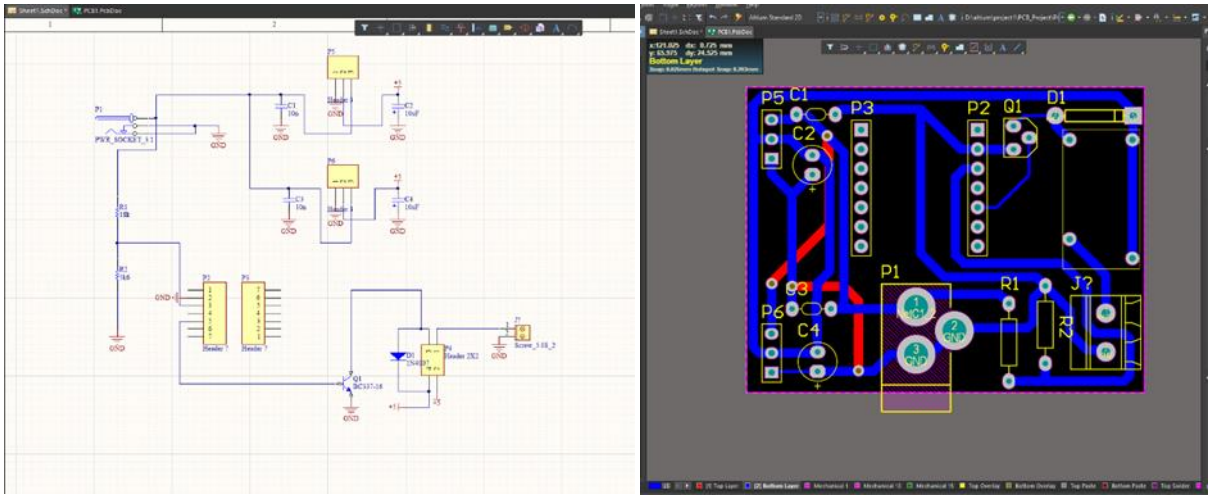


Figure 32: 5V5A Low voltage cut off Voltage regulator Altium schematics [Left], Altium traces. [Right].

Using two linear voltage regulators in series was chosen due to the increased stability gained, where the first regulator will provide a relatively stable output voltage, and the second regulator will further filter any noise left by the first, as well as any fluctuations from the power source. This is of high importance as mentioned the Jetson Nano is very sensitive to noise. Additionally connecting these in series provides a higher efficiency, due to lower dropout voltages causing less power to be dissipated in heat.

The choice to use both 10µF and 10nF capacitors also provides advantages, namely the 10nF capacitors placed before the linear voltage regulator filter out high-frequency noise, while the 10µF capacitors placed after the linear voltage regulator filter out low frequency noise improving ripple rejection, thus resulting in lower output impedance, and may be somewhat overkill for the application.

The other half of the circuit is the low voltage cut off, starting at the voltage divider the 1200Ω and 110Ω resistors provide a peak 2.81V to the 3V analogue input pin A1. Using this as an input. The voltage supplied to the A1 pin, as the battery voltage drops allows the microcontroller to cut power supplied to Jetson Nano once the 3s-LiPo battery voltage drops too low preventing further discharge, for safety of the LiPo and the subsequent charging process. The microcontroller is connected to the relay, via the NPN transistor, where a high signal is sent to the relay when the voltage to the A1 pin drops below a threshold signalling the LiPo is at the minimum 5.1V. The microcontroller is powered by a 5v supply from the drones PDB. The circuit is placed into a 3D printed PLA cover for protection Figure 33. Due to the use of the linear voltage regulators an appropriate heat sink needed to be added. A 25w heat sink was selected given the 25W of power supplied. Although the equation $P_d = (12.6V - 5V)\ X\ 5A\ = 38W$ suggest the use of a larger heatsink, due to the nature of the large airflow over the drone a smaller heatsink is acceptable, with the addition of adding a small 20m fan on top of the heat sink via the Jetson Nano GPU fan connector, which is not in use.

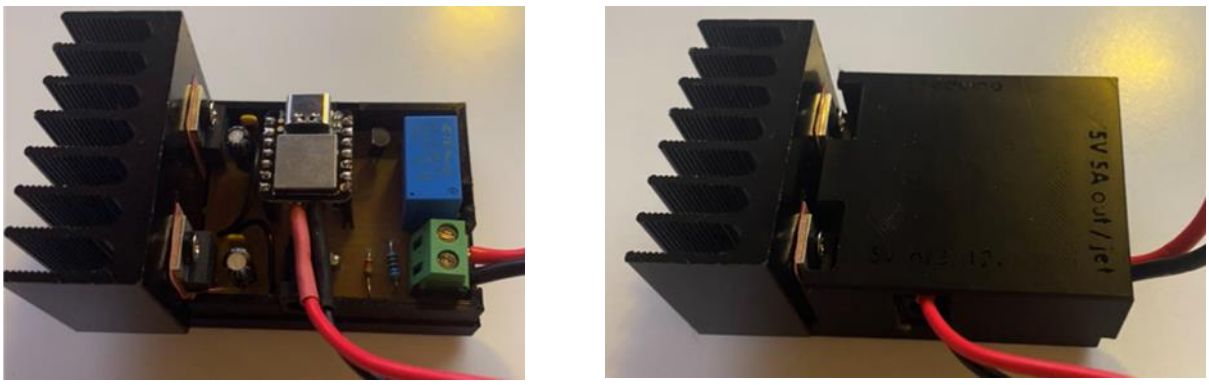


Figure 33: Voltage Regulator Circuit, with and without cover.

Equation used for voltage divider:

$$V_{out} = (V_{in} \times R_2)/(R_1 + R_2)$$
$$V_{out} = (14.8V \times 1200\Omega)/(110\Omega + 1200\Omega) \quad (3.3.2.1)$$
$$V_{out} = 2.81V$$

3.3.3 PDB

The power distribution board is the center of the electronic connections on the drone, here the 14.7V Dc supply from the 4S LiPo is devided between the 6 motors, the 12V outlet is used to power the gyro and FPV camera. The 5V out is used to power the Seeeduino Xiao on the Jetson Nano voltage regulator. The Pixhawk flight controller is powered by its own voltage regulator connected in parallel to keep a stable voltage supply to both the PDB and Pixhawk. The Matec PDB is of a good quality that can provide a stabel power output to the drones motors for iumpoved stability and safety Figure 34.

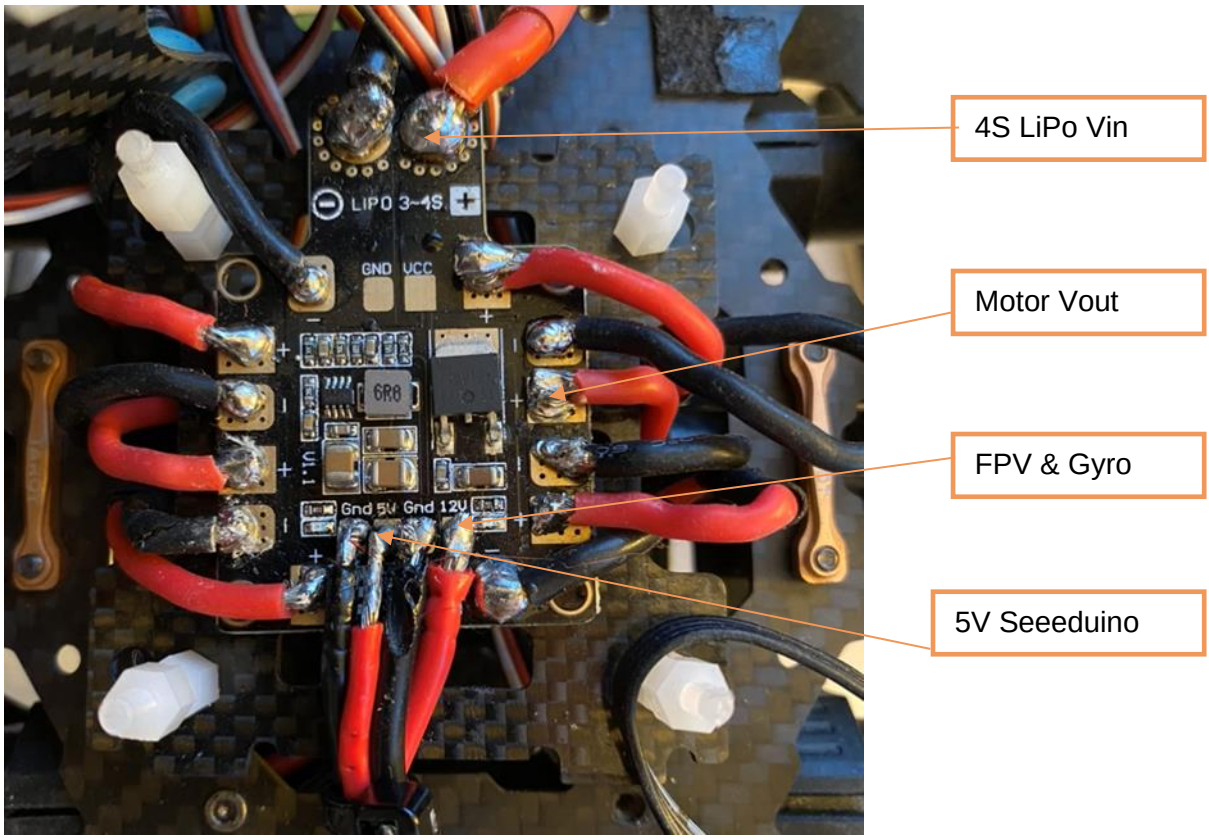


Figure 34: PDB.

3.3.4 Autonomous Flight

Setting up the Pixhawk autonomous mission is relatively straight forward. To test the system a test flight is conducted from home, using the mission planner API. The take-off, waypoints and landing position can be set, along with many other possible options such as loiter time, delay between each waypoint, the altitude, the flight modes and options and reactions to sensor inputs such as the range finders. To test the viability of the system a simple loop flight is set up from home Figure 35. Due to the hilly nature of the property an altitude of 110m is set. Using the Pixhawk API determining the altitude of certain areas is possible, showing an elevation of 105m at the take-off sight. To lower the chance of any crashes the drone is set to take off from the deck, fly in a C shape across the field and land on the driveway, approximately 40m from the take-off site. Getting the GPS coordinates accurate on the first flight is difficult, as the Pixhawk GPS coordinates and the actual positioning of the drone are hard to determine. The test flight, including autonomous take-off and landing is successful, being completely performed without the need for a remote control. Using a telemetry connection from the Pixhawk to the desktop computer allows for adjustments to the flight path, even while the flight is in progress, with the addition of receiving telemetry data such as current GPS position, altitude etc. Using the telemetry, the flight path can be fine-tuned, and any prior errors between the GPS position and actual position can be adjusted. After the first flight the landing sight position is found to be approximately 3m offset from the GPS position selected on the mission planner API.

Figure 35:Home loop Autonomous flight path

3.3.4.1 Orchard Autonomous Flight

To carry out an autonomous flight in the orchards, accurate coordinates would be required, due to the lack of a laptop the autonomous flight is not tested in the orchards, but a mock flight path, encompassing all five rows of the golden kiwi fruit orchards Is seen in Figure 36. Additionally, the gold kiwi orchard is highlighted in yellow, with green orchards highlighted in green Figure 36. To create an autonomous flight under the orchards we can first measure the width of each row. The kiwifruit orchard used for testing has a width of 4m, with some variance along the rows due to protruding branches a general width of 3.5m can be assumed. with the drone's width including the slow fly propeller is 1.1m. to determine a GPS accuracy a minimum clearance of 0.5m on either side of the drone should be assumed. This would give a GPS tolerance of 2.1m. Given the specifications of the GPS module used [71] a normal position reporting accuracy is 3-5m. Although this is larger than the 2.1m accuracy required, areas within range of (SBAS) Satellite Based Augmentation Service, accuracy can improve to 1m. Additionally further accuracy can be obtained from the use of F9P GPS modules [71]. Australia and new Zealand have a joint project names South-PAN in development to provide SBAS across Oceania, and is planned for 2028 [72]. Due to the covered nature of the kiwi fruit orchard, creating a flight path without further adjustments in the field, is not possible. As seen in Figure 36, judging the separation point of each orchard is difficult from areal imagery. Although not attempted the autonomous flight within the orchards should be achievable using the Pixhawk fight controller in the future and is an area in need of further research.

Figure 36: Potential Autonomous Kiwi Orchard Flight Path.

Chapter 4

Convolutional Neural Network

4.0.0 Chapter Overview

This chapter will cover the widely used convolutional neural networks Faster R-CNN, YOLOv7 and YOLOv7-Tiny, for object detection. It will cover the selection, creation, comparison, and implementation of these networks.

4.1.0 TensorFlow's Faster R-CNN

TensorFlow's Faster R-CNN is a very respected object detection neural network and has been implemented for the use of orchard yield estimation with a high accuracy result in [73],[27] & [25] to name a few. When Faster R-CNN was released in 2015, designed as an improvement on prior iterations namely R-CNN and Fast R-CNN, it outperformed prior state of the art models in terms of mAP scores on popular benchmarks such as PASCAL VOC and COCO datasets. Despite not being the fastest Neural Network, being outperformed by SSD and YOLOv2 in speed, its accuracy and precision remains high and competitive. Additionally, being part of the TensorFlow framework it is a very flexible and adaptable model with the major advantage of being massively speed up when TPUs are used, which are specifically designed to maximise the inference and training of TensorFlow based models, making Faster-RCNN more attractive for time sensitive Realtime object detection applications, outperforming traditionally faster models such as YOLO.

4.1.1 YOLO V7

YOLOv7 was released during this papers testing stages of Faster R-CNN, making a compelling case for outperforming Faster R-CNNs mAP scores, combined with a higher detection speed (when not considering the physical implementation of TPUs on the drone platform) YOLOv7 is an attractive architecture to use. [74] compares YOLOv7 with F-RCNN-R101-FPN+, a variant of the Faster R-CNN architecture that uses a backbone combination from ResNet-101 and feature pyramid network (FPN), with additional feature maps to improve performance. Tested on the 2017 COCO train and test data sets, the results showed a

AP_{test}/AP_{val} of 44.0% vs 51.4% for YOLOv7. But when looking at the Map when confidence is set to .5 in APtest50 the R-CNN based network shows a score of 72.4% vs YOLOv7 at 69.7% which is interesting Table 15. Additional advantages are the size of the YOLOv7 model being smaller, using only 36.9m parameters while Faster-RCNN uses 60m parameters, as expected a larger number of parameters correlates with a slower detection speed as more calculations are required in the convolutional layers etc. additionally YOLO is based on a single shot detector (SSD), giving it more advantages in terms of speed, higher computational efficiency, require less memory, and are better suited to real time detection on low power devices. With the main drawback being a decrease in accuracy when compared to traditional methods (in theory).

4.1.2 YOLOv7-Tiny

YOLOv7 is capable of a high mAP score, but its main advantages are focused on the speed of detection, it would be ideal to maximise the benefits of the detection speed, and sacrifice some of the accuracy/ mAP score. This is what YOLOv7-Tiny is designed to do. YOLOv7-Tiny is a lighter version of YOLOv7, designed for faster detection speeds with the compromise being a lower mAP score when trained on equal data. YOLOv7-Tiny has only 6.2 million parameters. In [74] when compared to YOLOv7 on the COCO dataset 286 FPS with a AP_{test}/AP_{val} = 38.7%/38.7%. with the highest being a $AP_{test}50$ = 56.7%. this shows it is likely necessary to set the detection confidence to a higher level such as .5 to get accurate results when comparing YOLOv7 $AP_{test}50$ = 69.7% Table 15. The Massive increase in FPS makes this model a good choice for real time object detection implementation on an Edge devices, drones, smartphones and other low processing, low power devices. The increase in FPS comes from using fewer convolutional layers and a smaller number of filters in each layer. It also uses anchor boxes to improve the accuracy of object detection.

Table 15: various model scores on important parameters tested on the same COCO dataset.

Model	#Param	FLOPs	FPS	$AP_{test}50$
F-RCNN-R101-FPN+	60.0M	246.0G	20	72.4%
YOLOv7	36.9M	104.7G	161	69.7%
YOLOv7-tiny	6.2M	13.8G	286	56.7%

4.2.0 Training Data Collection

Figure 37: Padded Store Bought [Left], Padded Web Image [Middle], Padded Drone Capture [Right].

Collecting the training data to create a custom neural network is the most important step regarding the model accuracy that can be achieved, as no amount of image manipulation, to increase dataset size and variance can make up for a bad training data collection [28]. To create a neural network with the foremost goal of accurately counting kiwi fruit orchard yields prior to picking, the neural network should be trained on ripe kiwifruit just prior to picking, making the same conditions as a farmer would deploy the system in. On the 26th of April 2022, images of a local kiwifruit farm were acquired between 12:00 noon and 5pm. The day was a mildly overcast day with good sunshine at times. Flying the drone over the top of the orchard with the hopes of being able to capture kiwifruit from the above angle proved unsuccessful, and so plan B, where flying the drone underneath the orchard worked very well giving a smooth image capture of the kiwifruits. The GoPro lens proved to have a wide enough angle to capture the whole width of each kiwi row within a very near distance. To acquire images the drone was flown through several rows using the GoPro that will be used to perform the object detection later. Additionally, more images were taken at a different resolution with two iPhones (iPhone 8 and iPhone 11). To further increase the accuracy of the model [30]. additionally golden kiwis were purchased from a local supermarket, and photos were taken of them with different backgrounds (to prevent underfitting the store-bought kiwis had their stems reintroduced). This technique further improves the accuracy of the trained models as the background in kiwi orchards will not always be so similar. For example, the kiwi farm we visited was outside, but other farms may be under protective domes or have different colour soil or fencing as in Figure 37 [Web Image]. To further increase the generalisation of acquired images several appropriate golden kiwi's orchard images were acquired from the internet and handpicked for image quality.

4.2.1 Image Augmentation

From the 600 Raw images, three image augmentation techniques were selected to increase the dataset size. First all 600 images were resized using padding to a 1000 X 1000 resolution. Picking the correct resizing size is more complicated when the same images are used across different model architectures. For Faster-RCNN a larger size such as 800x1200 or 1000x1200 is recommended due to its multistage architecture, whereas YOLOv7 works best with square shaped images, ideally at a multiple of 32 due to its SSD architecture. As neither can be satisfied with the same images, a square of 1000*1000 was chosen to better fit the high-resolution, wide-angle images, while not unfairly giving an advantage to either architecture for testing. Padding is done via a Python script. After padding the 600 images Figure 38, augmentations are applied. An intensity-based augmentation is applied by increasing the brightness and contrast. This type of augmentation makes the model more robust during changing lighting conditions, ideal for an outdoor farm application. Secondly salt and pepper noise was introduced to a further 600 images. This augmentation helps prevent overfitting, forcing the model to learn more robust features, additionally it helps create a more generalised model, by learning the underling structure of the kiwi. Lately random translation was added to 600 images. Translation is ideal to use as the kiwi is relatively featureless, this augmentation increases feature recognition robustness [30].

Padded Kiwi.

Padded + Intensity Augmentation.

Padded + Salt & Pepper Noise

Padded + Translation.

Figure 38: Image Augmentation Techniques used.

Figure 39: Orchard Training Image Capture Via Drone.

Although a larger number of raw images were taken via the video and combined iPhones, reducing the number of raw images was necessary due to time restraints, with many images containing more than several hundred kiwis each, labelling images becomes extremely time consuming. Image labelling was done using the Labellmg software. Using close-up high-quality images from the iPhone for training allows the model to become more diverse and create better feature maps to identify kiwis, this will come in useful when the drone is flown at different hights by various farmers as the kiwis will appear larger or smaller. Image augmentations were performed with the help of ImageJ an open-source image processing and analysis software. Conversion of image labels for training from .XML format, used by the TensorFlow framework for F R-CNN were converted to a YOLO Keras .txt format, with a python script to save time relabelling all images a second time.

3303	padded_n	1000	1000	kiwi	456	472	507	539
3304	padded_n	1000	1000	kiwi	499	474	544	539
3305	padded_n	1000	1000	kiwi	522	454	551	508
3306	padded_n	1000	1000	kiwi	480	477	520	532
3307	padded_n	1000	1000	kiwi	439	432	511	523
3308	padded_n	1000	1000	kiwi	489	456	554	554
3309	padded_n	1000	1000	kiwi	469	482	526	554

Figure 40: 3309 individual kiwis in test labels. (.CSV)

9160	padded_n	1000	1000	kiwi	455	408	573	533
9161	padded_n	1000	1000	kiwi	321	475	433	603
9162	padded_n	1000	1000	kiwi	238	513	347	658
9163	padded_n	1000	1000	kiwi	554	527	664	653
9164	padded_n	1000	1000	kiwi	489	471	602	579
9165	padded_n	1000	1000	kiwi	377	716	452	791
9166	padded_n	1000	1000	kiwi	419	743	492	817

Figure 41: 9166 individual Kiwis in Train labels. (.CSV)

4.3 Model Selection

Using the TensorFlow object detection API, Training the data set was done using the Faster R-CNN inception v2 coco (FRC-NNV2) model using TensorFlow 1.15.0. FRC-NNV2.

YOLOv7 was also tested; YOLOv7 is a single stage detector with the advantage being a higher processing time, but lower accuracy in theory to other architectures. YOLOv7 currently claims to “surpass all known object detectors in both speed and accuracy” [74]. Using supervised learning both models are trained on equal training and validation data sets. Calculating the mean average precision or mAP is a standard method used to understand the accuracy of a given model. mAP considers the trade-offs between the precision (the model’s ability to find true positives given all positive predictions) and recall (the models ability to find true positives given all predictions)

Training is performed on a home PC using a 2060 super NVIDIA graphics card. Using an older graphics card was required, as CUDA 10.0 and cuDNN 7.4.2 used by TensorFlow 1.15.0 does not support the architecture of newer graphics cards. Using CUDA to train both models allows for much faster training times [56] (around 24 hours). CUDA was also used to train YOLOv7 and YOLOv7-Tiny, with 600 epochs used for training. Testing various amounts of images and types of images was conducted, until the final 2400 images as described was found to be most effective. This is where testing the effectiveness of both image manipulation, as well as the use of closeup individual kiwis were introduced as suggested by [30].

To compare the effective mAP of the models, 10 images containing a total of 334 kiwifruits Figure 64, with a variety of environment cases were chosen from under orchard drone footage. Images of clearly visible kiwis, sunstruck images, and low light level images were tested with a IoU range of 0.05:0.1:0.95 + .005 the addition of the .005 range was added due to its significance in test results. The tests show all three models accurately predicting correct kiwifruit locations, but often miss possible detections, resulting in a high precision but relatively low recall, with the Faster-RCNN model at IoU = 0.5 having a particularly bad recall under 0.5 meaning less than half of the total kiwis in each image are not detected. The low recall could be caused by a lack of training image diversity, suboptimal model architecture or hyperparameters. This could be improved by further training experimentation. At IoU of 0.5 The Precision of the Faster R-CNN model is .982. this makes the model good at predicting true positives, showing it is not inaccurately labelling objects as kiwis. With an F1 score of .621 and AP of .699 the model has moderate to good performance. Comparing these scores to recorded scores of similar Faster R-CNN models we can see that the model likely needed a few more training images to be trained appropriately. Expected mAP scores are in the range

of 0.7 - 0.8 on popular datasets such as COCO or PASCAL VOC, so a score of 0.699 falls just short.

Testing the YOLOv7 Model shows a better result across each category, with a mAP of .898 the model is well trained Table 16, Table 17, Figure 42. When looking at the precision and recall in the IoU range of [0.005, 0.05, 0.1, 0.5] we see precision of [0.949, 0.972, 0.996, 1] and recall [0.901, 0.718, 0.706, 0.575] These results show that the model is performing better on average, the lower the IoU threshold. While the model can make predictions with high precision across the range of IoU values, the recall increases steadily as IoU drops, thus detecting a higher percentage of all objects within view the lower the confidence. The mAP indicates the model has a high accuracy in the given class. The F1 score shows a good balance between precision and recall. Overall, the metrics show the model is performing well, and I suspect that with less challenging test images the results would be of higher accuracy across the range. As the images used for testing were not contained in the training data, it can be said the model is not showing overfitting and is generalising well. But as the test images are still from the same orchard used to train the data, so more tests will be needed to confirm this accuracy.

Testing Yolov7-Tiny showed some surprising results, with a mAP score = 0.844 Table 16, we see that it is slightly less accurate when compared to YOLOv7 across a broad range of confidence. Looking at the precision and recall in the IoU range of [0.005, 0.05, 0.1, 0.5], precision is [0.962, 0.988, 0.966, 1] with recall [0.832, 0.739, 0.729, 0.605]. we see that again a higher overall AP can be found when the IoU confidence is low. This is again due to many obstructed Kiwi Fruits in the test images, just as would be seen in an orchard situation. The high precision in this case prevents the model from finding too many false positive predictions when the confidence is lowered, and instead can start detecting kiwis that are, 40,50,80% and more obstructed behind other Kiwi fruits. Overall looking at the total number of kiwi fruits detected, vs the total number contained in the test images, at no tested IoU threshold do the number of predictions exceed the total number within all the images, although, on some individual images the total detection count was higher than the total number of kiwis present. This only occurred at IoU thresholds =< .005. furthermore, with the addition of the tracking algorithm in the upcoming section the likely hood of counting a false detection is reduced again. Overall, both the YOLOv7 and YOLOv7-Tiny models show very good results with mAP = 0.898 and 0.844 respectively. The selection of the CNN to be used on in the drone will also depend on the FPS capability of the model and is discussed in the next section.

Due to the high amount of obstructed fruit, looking at Table 17, using a IoU threshold around 0.1 will be most useful, in regard to predicting accurate yield estimations when the object

detection is combined with the object tracking algorithm. Even a moderate amount of false positives predictions is acceptable, along as the true positives+ false Positives is ≈ actual yield count. As the tracking algorithm can, in some ways, eliminate false positives, if they are not too prevalent, the highly obstructed and sun struck fruits will benefit from a low IoU threshold.

$$Precision = \frac{TP}{\mathrm{TP} + \mathrm{FP}}$$

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

$$F1 = 2 * \frac{\mathrm{Precision} * \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

$$mAP = \frac{1}{\mathrm{c}} \sum_{k=1}^{N} (\mathrm{P(K)} \Delta R(K)$$

Figure 42: mAP sun struck Test Image, YOLOv7[Left], FRCNNv2[Right], YOLOv7-Tiny [Bottom].

Table 16: Table of mAP scores across 10 images at IoU of 0.5.

model	mAP @0.05:0.1:0.95 + .005	Average Precision At IoU = 0.5	Precision At IoU = 0.5	Recall At IoU = 0.5	F1 At IoU = 0.5
FRCNNv2	.782	.699	.982	.466	.621
YOLOv7	0.898	.712	1	.567	.719
YOLOv7-Tiny	.8442	.777	.998	.615	.754

Table 17: YOLOv7-Tiny mAP value matrix, sum of data points across 10 images tested at various IoU ranges.

Yolov7-tiny conf	#KIWIs	True +	False +	False -	Prediction conf	precision	recall	F1
.005	334	278	11	56	61.63	.962	.832	.892
.05	334	247	3	87	75.5	.988	.739	.846
.1	334	243	1	91	81.17	.996	.728	.816
.2	334	230	0	104	83.5	1	.689	.816
.3	334	217	0	117	84.2	1	.650	.787
.4	334	207	0	127	85.1	1	.620	.765
.5	334	202	0	132	86.4	1	.605	.754
.6	334	195	0	139	88.9	1	.584	.737
.7	334	189	0	145	90.5	1	.566	.722
.8	334	170	0	164	92.4	1	.510	.675
.9	334	109	0	225	95.3	1	.326	.492
.95	334	22	0	312	97.1	1	.065	.124
.995	334	0	0	334	0	0	0	0
Average	334	177.6	1.25	156.4	78.6	.919	.576	.741

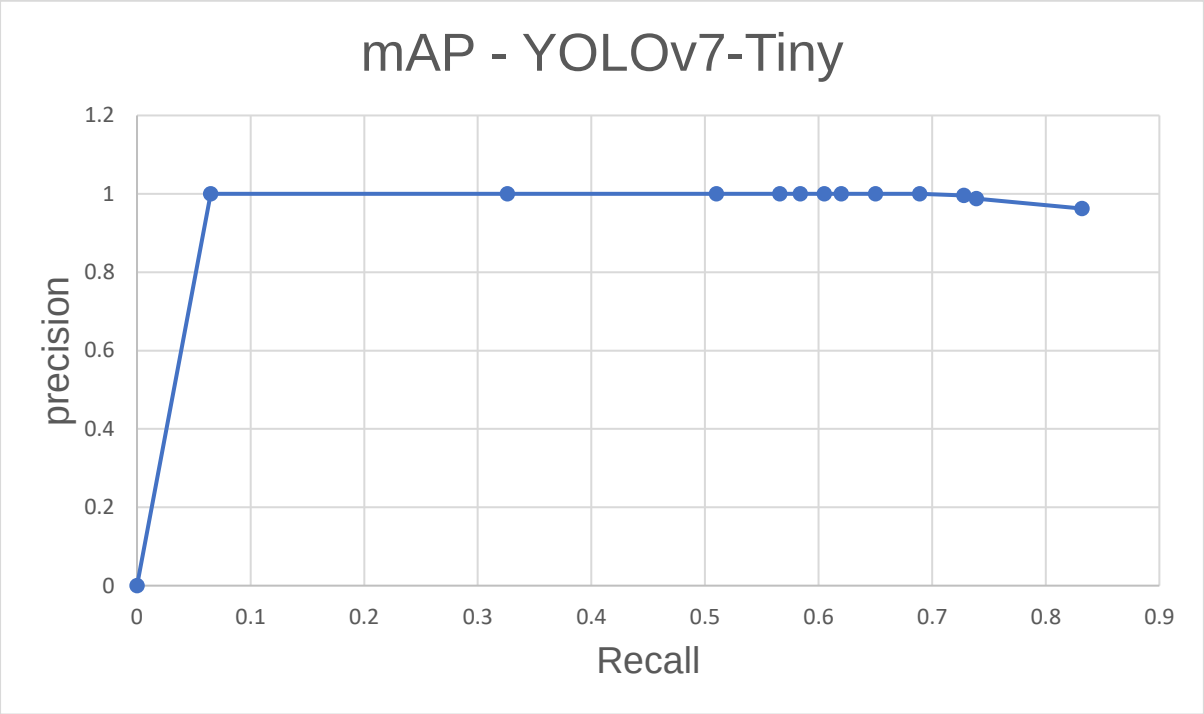


Figure 43: YOLOv7-Tiny median AP@0.05:0.1:0.95 + 0.005 mAP = .844

Table 18: YOLOv7 mAP value matrix, sum of data points across 10 images tested at various IoU ranges.

YOLOv7 Conf	#kiwi/img	True +	False +	False -	Average prediction conf	precision	recall	F1
.005	334	301	16	33	58	0.949527	0.901198	0.924731
.05	334	240	7	94	66.4	0.97166	0.718563	0.826162
.1	334	236	1	98	71.4	0.995781	0.706587	0.82662
.2	334	219	0	115	73.7	1	0.655689	0.792043
.3	334	211	0	123	75.1	1	0.631737	0.774312
.4	334	198	0	136	78.9	1	0.592814	0.744361
.5	334	192	0	142	83.3	1	0.57485	0.730038
.6	334	188	0	146	86.8	1	0.562874	0.720307
.7	334	182	0	152	92.2	1	0.54491	0.705426
.8	334	158	0	172	95.5	1	0.478788	0.647541
.9	334	83	0	215	98.2	1	0.278523	0.435696
.95	334	3	0	331	0	1	0.00898	0
.995	334	0	0	334	0	0	0	0

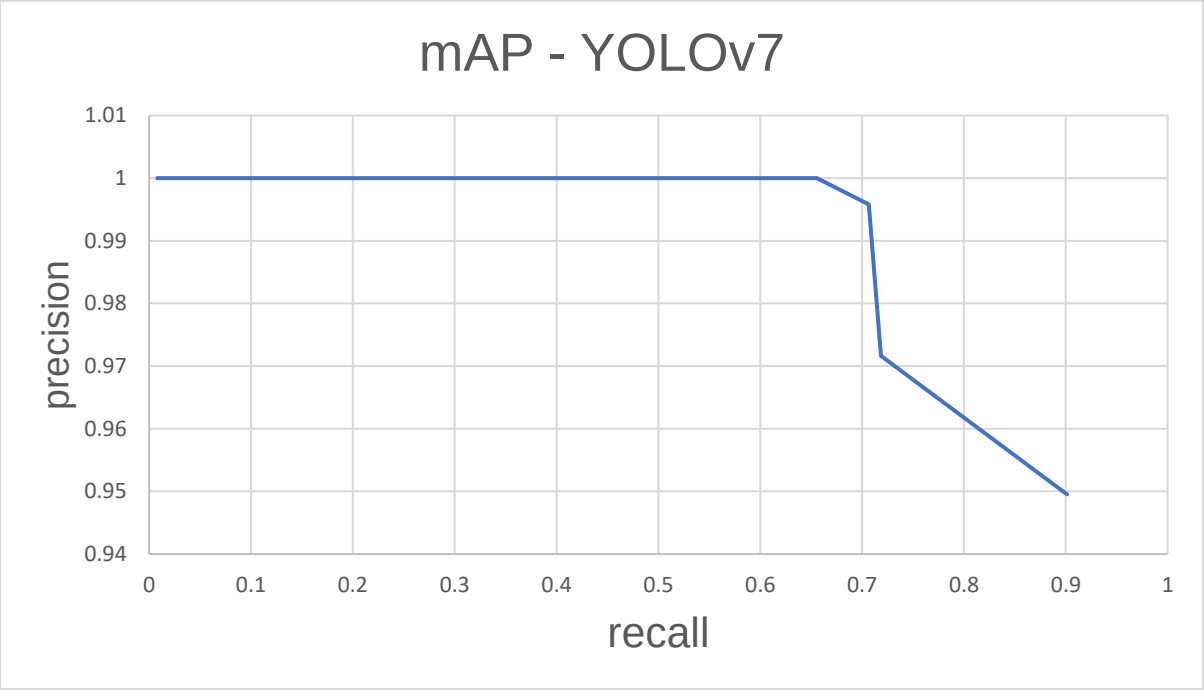


Figure 44: YOLOv7 median AP@0.05:0.1:0.95 + 0.005 mAP = .898

Comparison and selection of the model's accuracy is performed on the same PC used to train the models, the PC data can be used for mAP, Recall, Precision and F1 scores. However, when it comes to FPS the models will need to be tested on the hardware to be used on the drone namely the NVIDIA Jetson Nano. For comparison the 2060 Super GPU in the PC has 2176 CUDA cores where the Jetson Nano has 128 Cores. While testing on the PC the Faster-RCNN model averaged 8 FPS while the YOLOv7 Model was able to perform at 25 FPS, equal to the video input. For these reasons the YOLOv7 and YOLOv7-Tiny Models are chosen to be implemented and tested on the Jetson Nano. Due to time restraints only the YOLO models that have the same CUDA requirements are tested on the Jetson, as the ARM architecture requires a long setup time and debugging for such implementations. These results are again agreeing with [74] where Faster R-CNN reached 20 FPS and the YOLOv7 model reached 161 FPS on the COCO dataset. (a cloud-based GPU was used for processing). The processing time per image capture is important as a longer processing time results in an effective lower FPS of the camera. This results in either a decrease in accuracy in kiwis counted, or a slower flight speed for the drone, neither of which are advantages.

4.4 YOLOv7 FPS on Jetson Nano

To test the FPS capability of the JETSON Nano, a test video is used, that had been captured by the drone in the kiwi orchard, the day before picking. As testing the hardware on a live video stream in the orchard is not possible, getting truly accurate results is somewhat tricky. The time to load images from the internal SD card and the output from the GOPro3+ silver is slightly faster. Initial testing is done on the same kiwi orchard video between models and iterations in this section.

Testing the YOLOv7 model named “yolov7_custom” shows a total frame processing time in the range of ~699ms / 1.4 FPS to ~544ms / 1.8 FPS. This result is suboptimal, as this has all CUDA cores enabled. The JETSON Nano comes with 4GB of ram and an additional 494.6 megabytes per CPU core of swap memory (a SSD partition allocated for the use of extra RAM storage) while this swap memory is considerably slower in read and write speed, it can prevent bottle necks, system freeze and crashes due to kernel panic as seen when I took the screenshot in Figure 45.

As the JETSON Nano does not allow for the addition of RAM, I allocated 2GB of swap memory per CPU core into the partition, totalling 8GB of swap memory. This allowed the YOLOv7 model to speed up a little, inference time decreased by around 50ms or an extra .2 FPS, and average inference time became more consistent.

Now testing the speed of the YOLOv7-Tiny model shows a significant increase in FPS, additionally this model consumes around 5.3gb of RAM, thus only using 1.2 GB of additional swap memory, much less than the standard 2GB of swap found on the OS on startup, thus freeing up more space on the SD card for additional video storage.

Running the YOLOv7-Tiny on the same video with a bitwise_and operation, additional swap memory, and the tracking and counting algorithm (with an LCD output) the inference time + the NMS (Non-Maximum Suppression, post processing step removing overlapping detections) is between 85.5ms and 94.3ms, in the range of 10.6 and 11.7 FPS. This result is much more favourable than the 1.4-1.8 FPS rate with the YOLOv7 model.

To determine the accuracy of the two models, we can look at the total kiwi fruit detection count when using the track and count algorithm on the same video. The final kiwi count of the YOLOv7 model with a IoU of 0.5 shows a total of 932 and 809 Kiwi Fruit for YOLOv7-Tiny respectively. The actual number of kiwis in the video is unknown to me as it would be extremely difficult to count each one individually.

This video is footage of 1/3 of a kiwi orchard row. The accuracy of the count may be increased with a modification to the Region of Interests width, discussed later.

The system seems to allocate some memory into the Swap Memory partition no matter how much free space is available on the RAM, this may cause an additional bottleneck that is avoidable. Ideally all swap memory is removed in the final iteration to force the use of the RAM module exclusively Figure 45.

The use of a more optimised model via implementation of Tensor RT is also possible, but current development for implementation of a custom YOLOv7 model is not yet possible, such an implementation could see a higher FPS of around 20-25 FPS based on non-custom YOLOv7 tenors RT models running on a jetson nano 4GB.

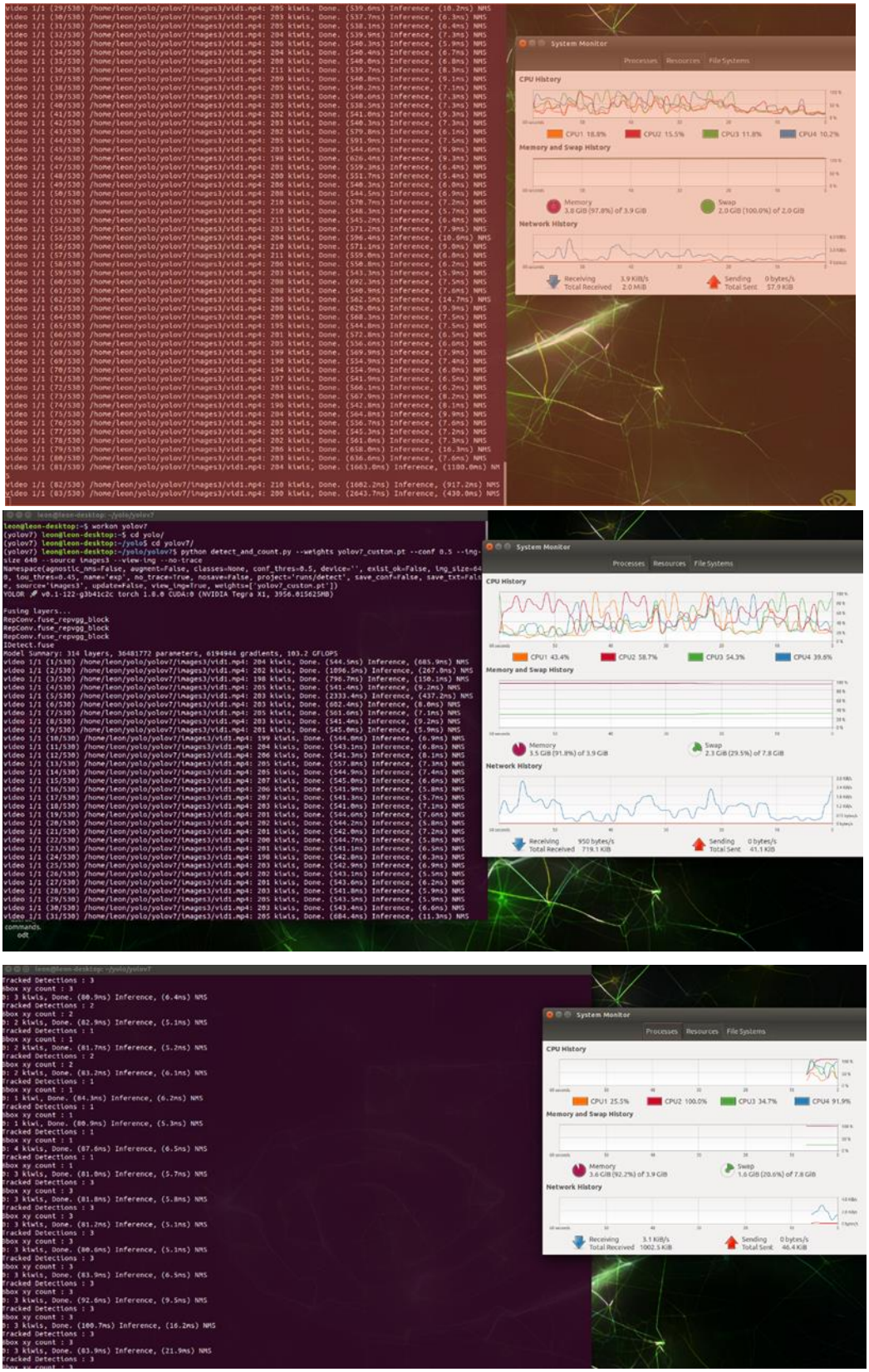


Figure 45: YOLOv7 inference + NMS time at 2GB Swap Memory [TOP], YOLOv7 8GB swap memory [Middle], YOLOv7-Tiny 8GB swap memory [Bottom].

4.5.0 Track and Count Kiwi Fruit

Detecting fruit using the YOLOv7 Convolutional neural network is half the process required to achieve yield estimation. The tracking of individual kiwi detections, followed by accurate counting is required. To try speed up the system more an additional bitwise_and operation was performed on each incoming image in the detection algorithm, with a .PNG mask that contained all (0,0,0) black pixels to cover out any parts of the image that was not crucial for detecting the kiwis, ideally this would prevent predictions being made in non-useful areas of the image, and is a feature included during the model training stages, where (0,0,0) padding was used to obtain images of the correct dimensions. Surprisingly this method did not result in a large improvement of inference speeds. Notably, without this operation the model could be used to also detect floor kiwi fruit, via an additional ROI in the tracking algorithm to further increase PA information. Next a tracking algorithm is used to prevent counting the same object multiple times. To do this the Deep SORT (Simple Online and Realtime Tracking with a Deep Association Metric) algorithm, which has been shown to be highly effective at tracking objects in challenging and crowded scenarios. To use this tracking algorithm to count, the following is done. Each kiwi is given an ID and added to a list via DeepSORT. Next adding two lines that stretch from one side of the image to the next allows for a region of interest in which any tracked kiwi, that first enters this region is counted and considered a true positive detection. This allows the algorithm to count each kiwi only once. Essentially the added constraint of needing to have been tracked for X frames in a row is what allows the tracking and counting algorithms to work together, producing high accuracy results. The algorithms can be finetuned to give an optimal result, which has been done in detail, in the following section. The addition of the tracking algorithm does not affect the speed of inference by any noticeable amount.

Figure 46: raw video input with total current detection output

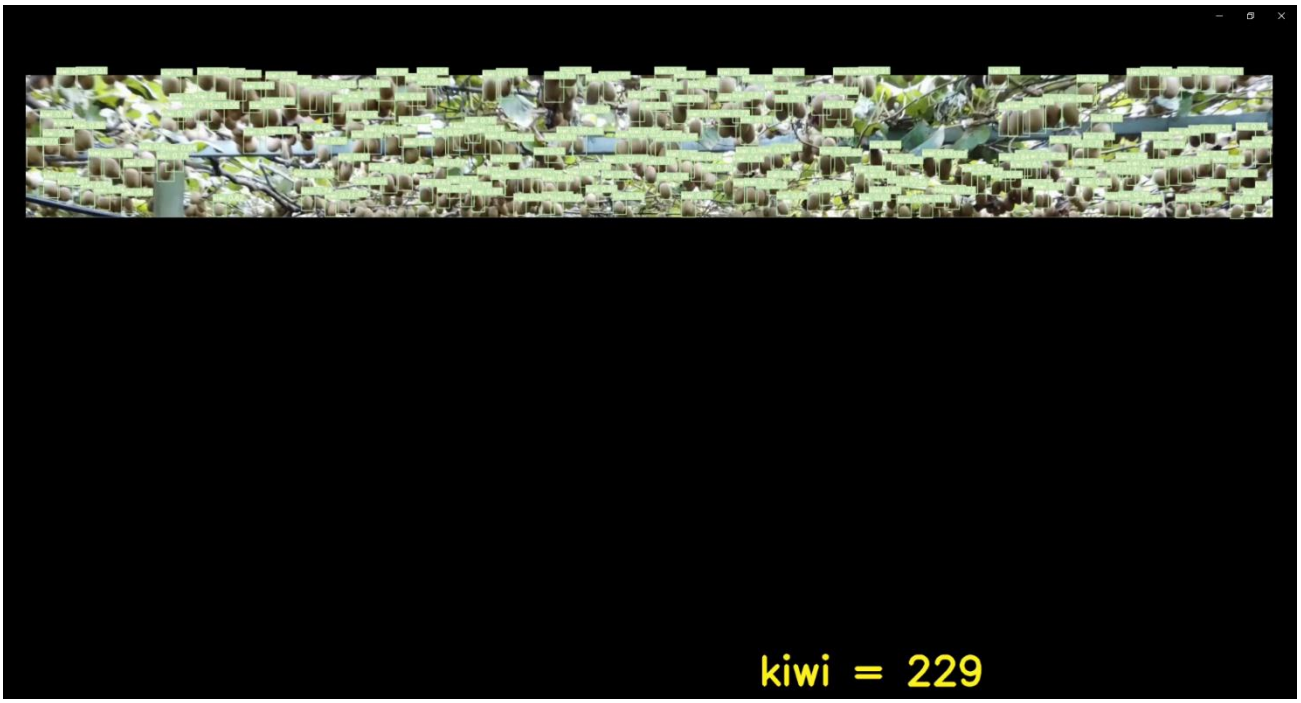


Figure 48: Bitwise_and on input image

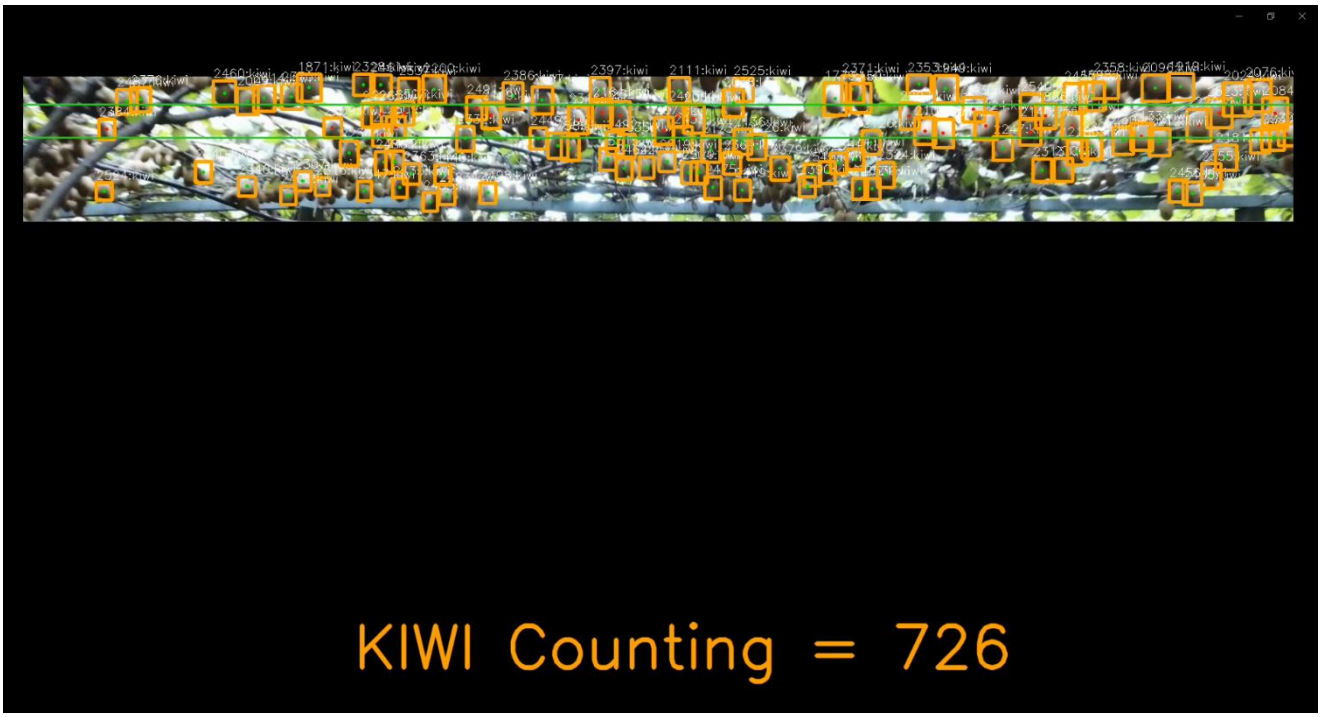


Figure 47: track and count with bitwise_and operation pre-Edge detection optimisation.

The LCD screen count Figure 49, allows for the real time output of the count number, thus making the whole process from setting up the drone to acquiring the total count number much shorter for the farmer. Gaining more accurate information such as region-specific yield estimation requires a keyboard, mouse, and monitor.

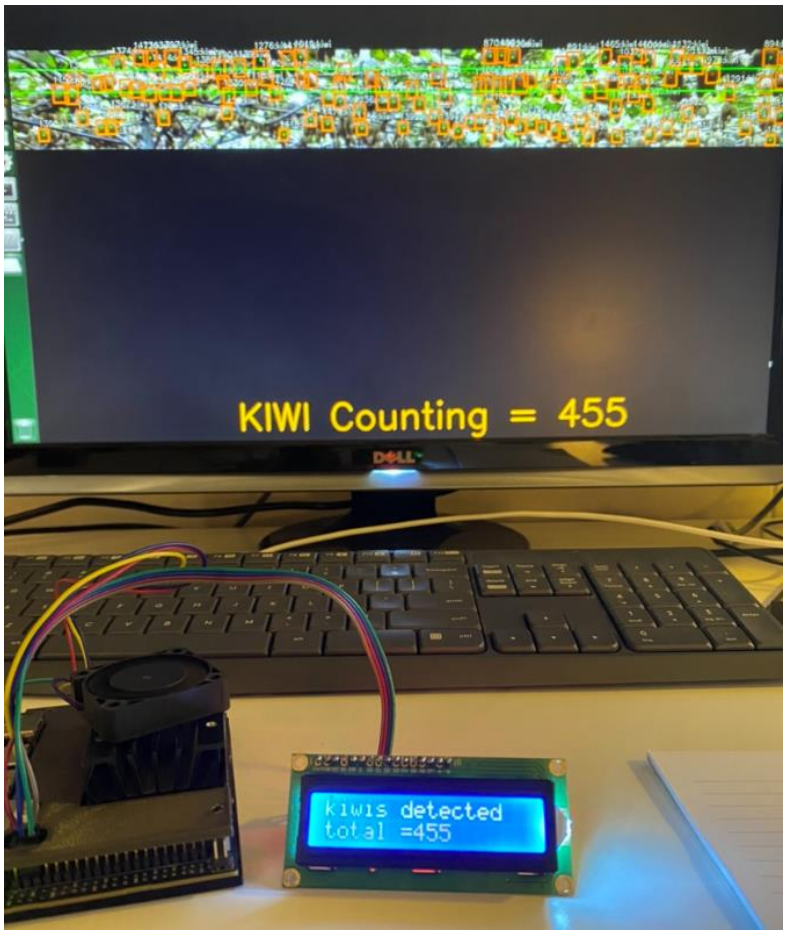


Figure 49: YOLOv7-Tiny on JETSON Nano + LCD

Additionally, the YOLOv7-Tiny model and track and count algorithm, is scripted to launch on power up, with an additional Q button press for closing and saving the detection, track, and count detection file for later analysis, further streamlining the operations needed to operate the orchard yield estimation system for the farmer.

4.5.1 Track and Count Accuracy

To determine the accuracy of the count number produced by the track and count algorithm, a few tests can be conducted. To determine the ideal settings to produce the more accurate results several parameters must be adjusted. Namely. IoU threshold of the detection algorithm, region of interest hight and location used for counting, “Sort_max_age” variable used to discard incorrectly identified kiwi fruits after X iterations of no subsequent detections following the initial detection, in other words the number of frames an unmatched tracker ID exists for. “Sort_min_hits” a variable used to count kiwi fruits only after they have been detected X frames in a row. “sort_IoU_thresh” used in the sort script and has the same function as the regular IoU. Additionally, the bounding box sizes intersect/ overlap can be adjusted to improve tracking with a lower FPS.

To simulate the real-world scenario more accurately, a 25FPS, 5 second video with a known number of kiwifruits is selected. The parameters mentioned are adjusted until the number of detections matches as closely as possible the number of known kiwis in the video. Additionally, a second video is used, which has had every second frame removed from the 5 second video, to simulate a 12.5 FPS video stream, which more closely represents the FPS achieved on the JETSON Nano.

To further increase the accuracy of the tracking algorithm, two additional regions of interest are placed on the very edge of the screen, this allows for a “wider” view angle Figure 50. Due to the drone moving forward in the centre of the orchard, fruits follow a path on the screen, that is not a straight vertical line from bottom to top, rather they move outwards from the centre, thus fruit on the edge of the screen will move out of the field of view via the left or right side of the camera field of view, before being given a chance to be counted by the horizontal region of interest at the top of the “screen” field of view. When fruit are counted their centre point turns red.

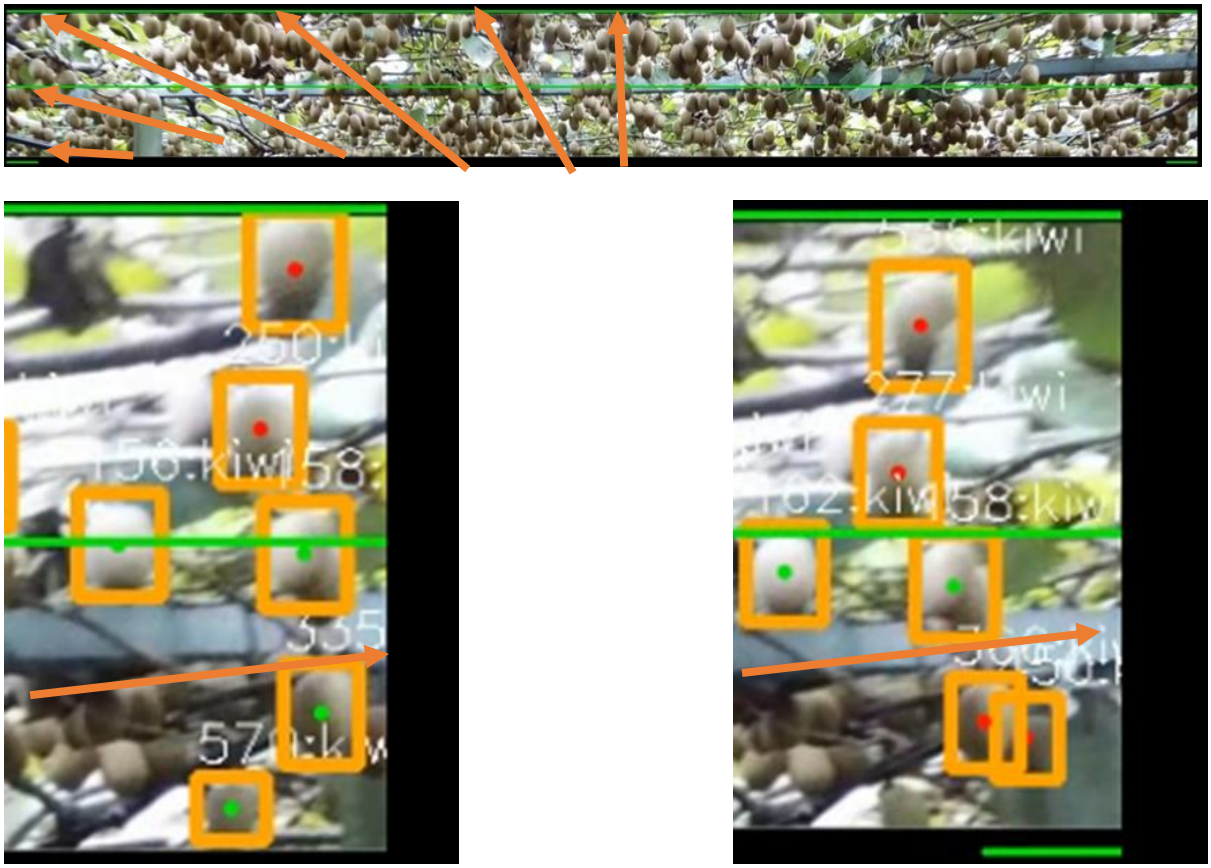


Figure 50: visualisation of fruit perceived movement [Top], uncounted fruit, moving off edge of screen. [Left], additional ROI for detecting fruits moving off side of screen, but being counted [Right]

To determine the number of fruits in the five second video, the video is split into 2 frames, clearly containing all the kiwis in the video, that end up in the ROI. This is done simply by analysing the start and stop positions of the Region of Interest, in the short five second video, and subsequently extracting 2 frames that encompass this area. Using the detection algorithm on the two still images, the total number can be obtained with reasonable accuracy, additionally the fruit were counted by hand and a total of 341 acceptably visible fruit were counted. Thus, a detection result of 341 will be considered a 100% accuracy result for the track and count algorithm.

Figure 51: Five second test video, to fine tune and determine tracking algorithm accuracy split into two frames.

Table 19: Total Kiwi Fruit count, given various deep sort variable weights on Yolov7(Green) & Yolov7-Tiny (Yellow) from a 5second video containing ~341 Fruit.

IoU Threshold	Sort Max Age	Sort Min Hits	YOLOv7-Tiny	+edge detection	YOLOv7	+edge detection
0.05	5	2	427	479	572	426
0.1	5	1	432	483	408	461
0.1	4	1	435	484	417	470
0.1	3	1	443	492	430	483
0.1	2	1	450	498	445	499
0.1	1	1	470	517	485	541
0.1	5	2	379	419	527	407
0.1	4	2	379	419	363	412
0.1	3	2	380	420	369	418
0.1	2	2	380	423	382	433
0.1	1	2	395	431	412	464
0.1	5	3	**346**	380	334	380
0.1	4	3	351	384	338	384
0.1	3	3	351	384	**344**	390
0.1	2	3	351	384	353	401
0.1	1	3	358	390	382	429
0.1	5	4	327	356	327	367
0.1	4	4	329	359	329	369
0.1	3	4	328	358	334	374
0.1	2	4	327	356	**341**	382
0.1	1	4	331	359	364	405
0.1	5	5	320	**344**	313	353
0.1	4	5	323	**348**	319	359
0.1	3	5	319	**344**	328	369
0.1	2	5	314	339	345	386
0.1	1	5	316	**342**	345	386
0.1	6	5	305	327	313	353
0.1	6	6	305	327	306	**343**

Running YOLOv7 object detection on a 25 FPS video allows us to see the ideal settings given an ideal circumstance, which can be used to understand the results in the YOLOv7-Tiny at 12.5 FPS case. Additionally this information is useful for further research purposes where a more powerful microprocessor may be implemented. In terms of the scope of this project, only the Yolov7-Tiny results are of true importance. And the parameters will be tuned to generate the most accurate result for Yolov7-Tiny at 12.5 FPS.

Sort Max Age: The Sort Max Age parameter determines how many frames a once detected fruit has its ID stored for, after not being detected, before it is removed from the list. Increasing the Sort Max Age can help prevent true positives from being excluded from the count, but it can also cause false positives detections to be counted, as the new detection of a false positive with the same ID can then be counted assuming it passes the Sort Min Hits check. In this case, we can see that increasing the Sort Max Age results in a decrease in the total count number for both detection methods, suggesting that the algorithm is losing track of false positives, then counting them later due to the increase probability that it is detected again and stored as a detection.

Sort Min Hits: The Sort Min Hits parameter determines how many frames a detected fruit must be tracked before it is added to the count list. Increasing the Sort Min Hits parameter can help filter out false positives, but it may also cause valid detections to be missed if they are not tracked for long enough. In this case, we can see that increasing the Sort Min Hits parameter generally results in a decrease in the total count number for both detection methods, indicating that the algorithm is detecting and tracking false positives, but has a lower possibility of continuous tracking and counting of the false positives.

Edge Detection: The edge detection parameter affects how fruits that move off the edge of the frame are counted. When edge detection is turned off, fruits that move off the edge of the frame are not counted, resulting in a lower total count number. When edge detection is turned on, fruits that move off the edge of the frame before entering the region of interest are counted, resulting in a higher total count number. In this case, we can see that turning on edge detection generally results in a higher total count number.

To find the perfect balance between tracking a true positive and counting it before it moves off the screen, as well as deleting false positive IDs before they are counted suggests a higher Sort Max Age, combined with a higher Sort Min Hits. The addition of the Edge Detection also increases the accuracy of the tracking algorithm, by gaining a “wider view” to count the full width of the orchard. This is reflected in Table 19, where looking in the yellow column, we see when Sort Max Age and Sort Min Hits are high, the accuracy is improved.

4.6 Summery of Chapter Four

Comparing the CNN models, provide a clear answer as to the ideal model given the scope of this project. Seeing the advantages in both mAP score and inference time on the YOLO models, when compared to the FR-CNN architecture the answer is clear. YOLO v7 is able to outperfourm the FR-CNN model. With the addition of YOLOv7-Tiny. A smaller, faster, although slightly less accurate model architecture, we are provided with the perfect mix between accuracy and inference time, allowing this project to be successful in achieving its real time detection goals. Training the models, proved time consuming and more complex than one might first imagine. Over 12,000 individual kiwis contained within images were used to train the models, this was required to achieve a well performing model. Having to use a generally lower IoU threshold than what is standard, is likely due to large amounts of partially visible Fruit, this may be unavoidable, unless a model is specifically trained to detect partial fruit with more certainty. Using DeepSORT for tracking proved successful, with the correct tuning of tracking parameters, accurate to within a 97%-99% range, although further testing is needed to confirm this with higher certainty. While the use of other frameworks such as keras were not explored due to their lack of previlance in letrature related to fruit detection. These other archetectures are certainly an avenue for further research in the future. Additinaly the use of pre trained models/ Libraries to identify kiwifruits was also not explored in depth. This is due to pre trained models such as the massive tensorflows COCO model data set being trained on 2.5 million label instances of common objects (Common Objects in Context). Such detailed models only contain regular or common objects such as houshold items, animals etc. other popular pre trained models compatible with YOLO or tensorflow were investigated, but the class for kiwifruit was not found.

Although pre trained models were not used, the use of pre labeled images of kiwifuris were also explored, but due to the specific natuire of images required by the drones angle, no usefull librarys were found. This resulted in all images and labels being created from scratch via the drone combined with manual labelling.

Chapter 5

Experiments and Results

5.1 Chapter Overview

This chapter presents and analyses the results from the experiments encompassing all aspects of the project into a single experiment, the methodology used to obtain the data is explained, with results for various augmentations of the system being discussed. The experiments consist of results obtained from the drone system, being compared to results gained from the picking and counting of the orchard, to better understand the accuracy and implement ability of the overall system.

5.2 Method

Testing the drone orchard yield estimation system under the orchards is done on the 7th May 2023, one year 11 days after the image acquisitions of kiwi fruit from the same orchard. The test flight took place on an overcast day, with intermittent rain. During the two-week period before kiwi fruit picking took place, New Zealand was experiencing a period of heavy rain, stormy, windy weather, and severe flooding. The drone flight is done two days before flooding of the orchards, with picking planned for the next day, but was delayed for 9 days. It is not known how many fruits had fallen to the floor during the storm, comparison image before and after of the orchard floor. Looking at the ripeness and shape of the gold kiwi fruit from this season, compared to the last season from which images are acquired, the shape, size and colour is very similar. This is ideal for the detection system. However, the overcast weather does reduce lighting conditions bellow what was seen during the image acquisition, Figure 62. Due to rains the orchards were wet and dripping occasionally, for this reason a protective cover was placed over the electronics of the drone to prevent damage Figure 52, leaving only the LCD screen visible. An additional tuning flight was carried out with the cover in a nearby field. Additionally, the FPV camera was not plugged in and the jetson nano was powered off to save battery during fine tuning, also seen in Figure 52. Due to space restraints in the orchard, the stability tune needed to be caried out in the nearby field.

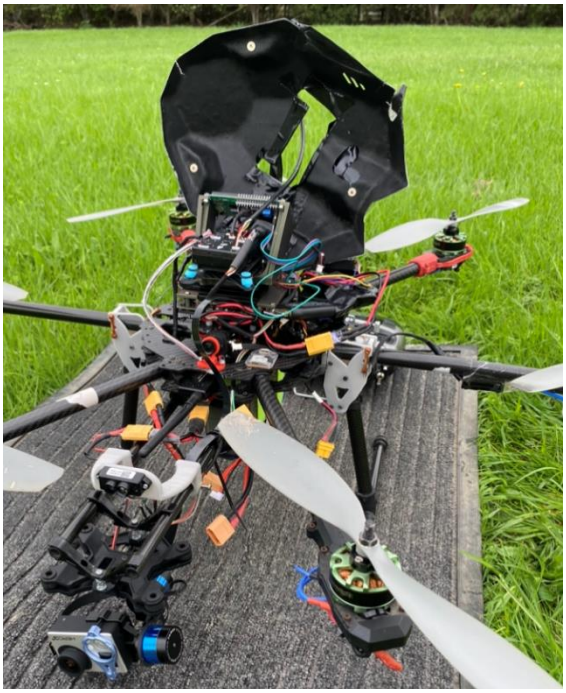


Figure 52: Field Tune flight with open cover

Figure 53: Hover in field

To carryout the test flight, the drone is placed on the takeoff/ landing matt, to allow for a smoth take off, preventing grass tangeling the landing gear, fortunately the grass had been mowed around the outside of the orchard anyway. As seen in Figure 54, Figure 55 the orchard floor has not been mowed towards the end of the row (the row actually continues for a extra 20m after the high grass), but it had not been mowed or trimmed in a long time, preventing the drone from flying the whole way. The high grass cuting across the middle of the orchard row is present due to a recent removal of a canopy, leaving a large 4m wide gap in the orchard. So the decision is made to fly the drone remote-controlled upto the end of the visable orchard, and land on reaching the “end”, while still under the orchard. Due to flying the drone, taking images of the drone flying under the orchard was not possible. Flying the legth of the orchard, at a speed of around 1m/ 3-5s the flight time is around 3 miniutes. After two flight causing issues with hight and obstical avoidance due to hanging vines, the third flight was able to reach the end of the half row in a constant speed, resulting in a yeild estimate of reading of 8782 fruits Figure 55. To further assess the accuracy of the system a second video is taken on the day of picking, using the flight only to record the orchard, to determine the diffrince on count when using post processing rather than real time processing, thus resulting in a higher FPS video. These results are also discussed.

Figure 54: Pre-flight take-off position.

Figure 55: Drone at the start of the tall grass [Left]. LCD detection result [Right]. Length of orchard flown [Bottom].

5.3 Kiwi Fruit Picking

On the 16th may kiwifruit picking was carried out by a team of ~ 40 pickers. To gain an accurate understanding of the yield, four pickers were happy to assist in the project. The method of picking is simple, one tractor carrying either 3 or 4 kiwifruit crates is driven down a central row, with a row on either side of the tractor. With five rows of orchards on this section, two tractors were driven down the central rows. Pickers then make their way down from the top of the row, following the tractor, to the bottom (nearest to the entrance) picking fruit into their basket, then emptying their baskets into the central crates pulled by the tractors Figure 59. Once the tractor crates are filled level a new set of crates is brought in. To gain an accurate estimate of total yield one could ask each picker to count every kiwi fruit they pick and tally their total down, although possible this was not done. Instead, for the four men picking fruit in the row of interest, an average of the number of fruits that fit into their picking basket was taken, and the total number of baskets filled is counted, thus the total number of fruits picked from the row is an estimate. Each of the four pickers is asked in intervals to count the total number of fruits that fit into their basket, results ranged from 120-156 fruit per basket. As mentioned earlier prior to picking and fruit spraying a video is taken to obtain 48 FPS video (the highest FPS available on the Go Pro at 1080p resolution) of the full orchard row, this video is used to run Yolov7 and Yolov7-tiny on the home desktop, to compare the neural network to the actual count acquired later in that same day. To prepare for picking, the fruits are sprayed down with a consumption safe preservative three hours before picking starts. Due to light rains earlier, the pickers wait for the fruit to completely dry in the sun, this prevents moisture from causing bacterial growth and rotting during later storage, this puts a time pressure to pick all the fruits before another light shower begins.

Figure 56: Fruit preservative spraying pre picking.

Figure 58: Four pickers picking only from the row in question.

Figure 57: level full bags ~ 120 fruit

Figure 59: tally result from first and second half of row. [Top], Crate of kiwifruit filled [Left], Crates waiting to be filled [Right].

5.4 Actual Fruit Yield

Table 20: quantity of Kiwi fruit per basket

Number of fruits per basket.	Picker 1	Picker 2	Picker 3	Picker 4
	156	138	155	120
	129	134	146	132
	138	148	134	133
	140	134	153	140

total baskets picked in first half of row = 80. Total baskets in second half of row = 67.5, full row = 147.5

To generate an accurate estimate of the total fruit picked over the 147 baskets, a statistical analysis is done. Creating a probability estimation to determine the average number of fruits collected in each basket via a normal distribution, in the range of 120-156. Using the observed data, creating a cumulative distribution function (CDF) of the normal distribution as a statistical method of yield estimation. Using the R software, we can generate a normal distribution.

Average Fruits per basket = 139

X is the number of fruits in each basket. μ is the mean = 139 n is # of observations = 16

Standard deviation $\sigma = \sqrt{\left(\frac{\sum(x-\mu)^2}{n}\right)} \quad \sigma = 9.67$

Estimate number per basket = (CDF (156 – CDF (120)) * 80 * 139

Using R a fruit yield estimate is calculated, with a margin of error at 95% confidence interval, and creating the normal distribution graph.

> mean_value <- 139

> sd_value <- 9.67

> basket_count <- 80

>

> cdf_120 <- pnorm(120, mean = mean_value, sd = sd_value)

> cdf_156 <- pnorm(156, mean = mean_value, sd = sd_value)

> probability_range <- cdf_156 - cdf_120

> print(probability_range)

[1] 0.9359111

>

> estimated_total <- probability_range * basket_count * mean_value

> print(estimated_total)

[1] 10407.33

>

> # Calculate the standard error

> standard_error <- sd_value / sqrt(basket_count)

> print(standard_error)

[1] 1.081139

>

> critical_value <- qnorm(0.975, mean = mean_value, sd = sd_value)

> print(critical_value)

[1] 157.9529

>

> margin_of_error <- standard_error * critical_value

> print(margin_of_error)

[1] 170.769

>

> # Calculate the lower and upper bounds of the confidence interval

> lower_bound <- round(estimated_total - margin_of_error)

> upper_bound <- round(estimated_total + margin_of_error)

>

> # Display the estimated total and the confidence interval

> cat("Estimated Total: ", round(estimated_total), "\n")

Estimated Total: 10407

> cat("95% Confidence Interval: [", lower_bound, " - ", upper_bound, "]\n")

95% Confidence Interval: [10237 - 10578]

[1] 10407.33

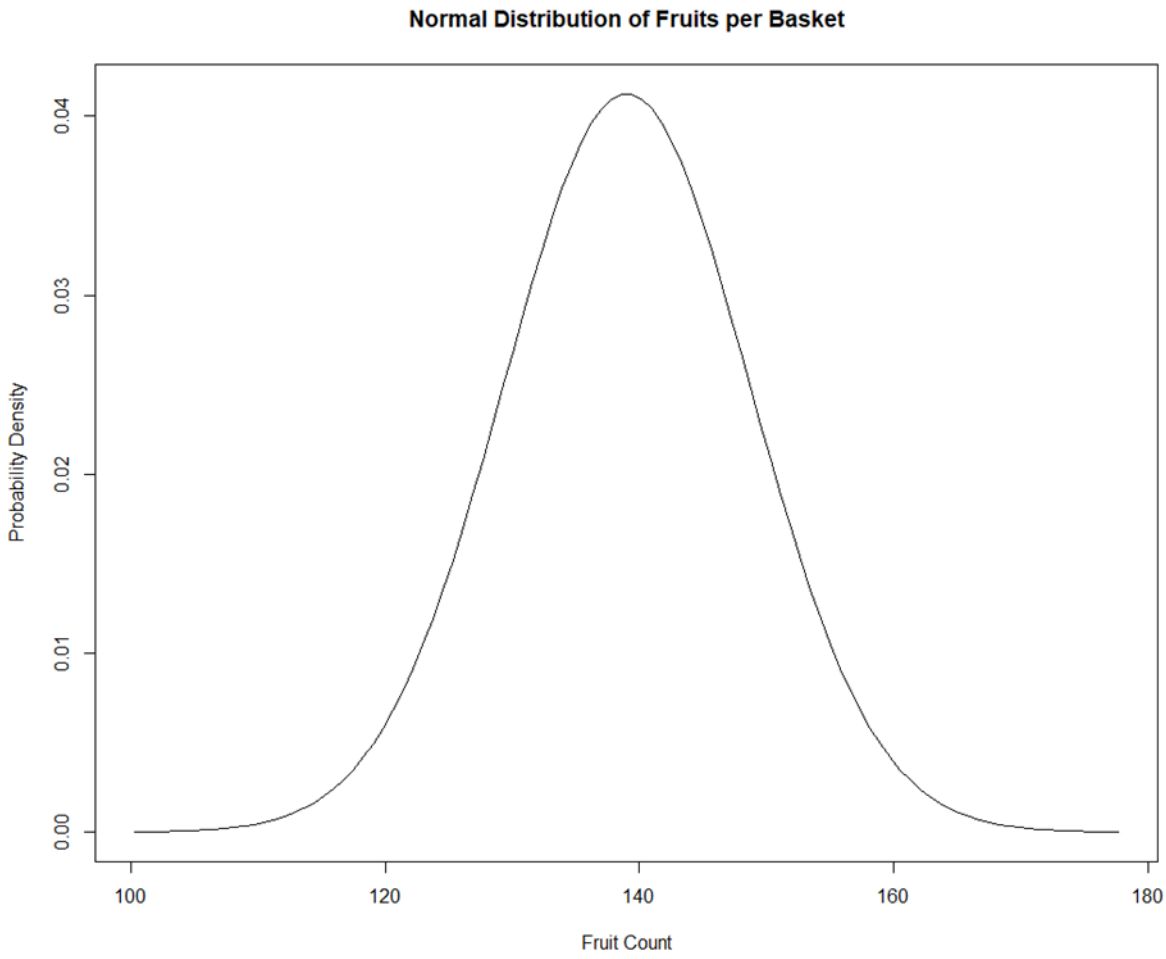


Figure 60: Normal distribution of fruits per basket / yield estimate

Using this statistical analysis, we find that the total number of kiwis in the first half of the orchard row is, 10407 with an error of 170 giving a range between [10237 - 10578] with the <95% confidence interval.

To further analyse the accuracy of the system the full orchard row yield estimate is also calculated. Using the same method, only changing the number of baskets collected to include all the baskets counted. Thus, the *basket_count* variable is changed to 147 instead of 80. Performing these calculations shows a full row yield of 19123, with a 95% confidence interval of [18997 – 19249]

Table 21: various Orchard yield estimation techniques vs actual orchard yield count

	Real Time (YOLOv7-Tiny)	12.5 FPS YOLOv7-Tiny	24FPS YOLOv7	Actual count
Half row yield	**8782**	8220	8766	**10407+-170**
Full row yield	---	15577	16739	**19123+-126**

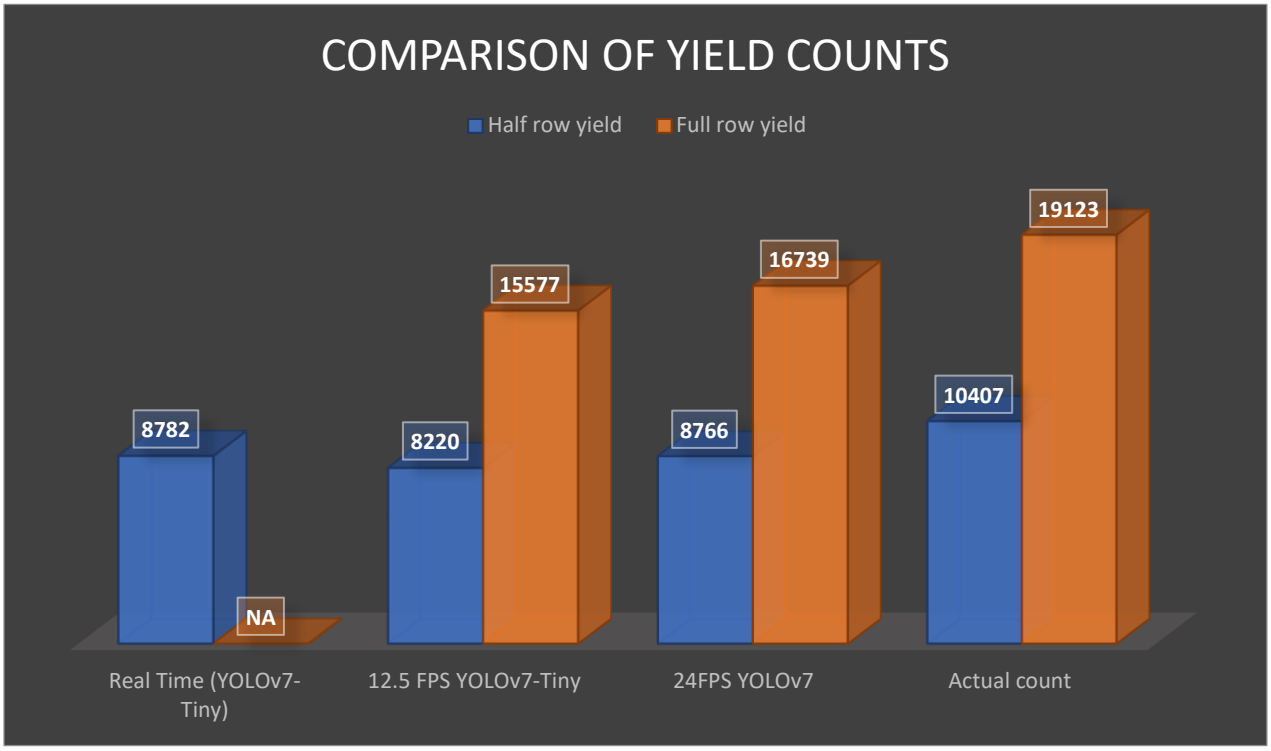


Figure 61:Comparison of yield estimation techniques

Comparing the various methods of orchard yield estimation shows some interesting results, both in actual yield result and method of acquisition. Looking at the 8782 - kiwi fruit yield count obtained from the real time processing drone flight, we can see that the method proposed by this paper has some merit, with the actual count being close to 10407 for the same section. Expressed as a percentage the result is 84.3% accurate with a 1.93% variance. Looking at the mAP scores of YOLOv7-Tiny in Table 16, shows a map score of 84.42. this correlates very well with the accuracy of the model when tested in the field. Similarly, when testing YOLOv7-Tiny on a 12.5 FPS video on a video taken on the day of picking shows a result of 8220 for the first half of the row and 15577 for the full row. When compared to the actual count of 10407 +- 170 and 19123 +- 126, shows an accuracy of 78.98% +- 2.06% and 81.45% +- .8% respectively for the half and full row.

Using the YOLOv7 model, gives a less accurate result of the first half row equal to 8766 or 84.2% +- 1.9%, but become more accurate over the full length of the orchard row with the result of 16739 or 87.5% +- .75%. Comparing these results to the mAP for this particular, YOLOv7 model, a mAP of .898 or a general accuracy of 89.8% which makes the results of 84.2 and 87.5 expected. Additionally, the accuracy of the track and count algorithm must also be taken into consideration. The results of the track and count algorithm show that the fine tuning of the Deep SORT algorithm directly influence the accuracy of the overall system, with finetuning of the system mostly focused on the YOLOv7-Tiny model, the discrepancy in the accuracy of the two systems becomes more reasonable. Additionally, the accuracy of the model is affected by the storm which occurred after the full system test, and before the counting of the kiwi fruit via picking and statistical analysis. This undoubtably gives the drone test flight bias to a higher than actual count when compared to the picking method. While the total number of fallen fruits during the storm remains unknown making the analysis difficult. Overall, the system does show to be capable of performing Kiwi fruit orchard yield estimation at a high accuracy Figure 61, Table 21.

Figure 62: Orchard before storm [Top], After storm [Bottom].

Chapter 6

Discussion

6.0 Chapter Overview

This chapter discusses the results of the experiments, both for experiments directly linked to orchard yield estimation, as well as the results and performance of sensors used for the overall system to function as designed. Additionally, the chapter highlights shortcomings within the overall system, as well as personal opinions on how errors were introduced into the system, how to mitigate these issues, leading to further improvements to the system as well as suggestions on how I would personally tackle the problem of kiwi fruit orchard yield estimation differently in the future.

6.1 Orchard Yield Estimation Results

Overall, the results from both the real time processing flights and post processing produced results of high accuracy for a prototype system. These prove a successful implementation of the solution to the problem of kiwifruit orchard yield estimation. The YOLOv7 CNN architecture proved to be an exceptional object detector, as well as the Jetson Nano proving to be a high quality, and powerful microprocessor. The methodology used can accurately detect and count kiwi fruits with an accuracy relative to the mAP of the CNN architecture, suggesting that over time this method of orchard yield estimation will only improve. Some limitations are introduced when considering the lateral horizontal and vertical movement of the drone due to drift, wind gusts or obstacles such as leaves and vines hanging from the orchard. Additional errors are introduced when sun strike on the camera is high, and when kiwifruit is positioned directly behind one another making them hidden to a front facing camera angle. The system also relies on a clear orchard floor, although mowing in not required, the hinderance of tall grass completely prevents the system from functioning. Additionally, the low battery life available for the jetson nano presents additional limitations for the length of orchards that can be counted, combined with the slow flight speed required to gain an acceptable result due to low FPS. Overall, the use of the DeepSort tracking algorithm proved highly effective and allowed for a high level of fine-tuning dependant on the relative speed of the camera and the FPS. Finding the ideal weighting for each variable in the DeepSort algorithm, although time consuming does result in accurate and repeatable results.

6.2 CNN Model Validation

The YOLOv7-Tiny model has a mAP of .8442 when trained on the Kiwi Fruit data set, with the IoU range of .05:0.1:0.95 +.005. While the IoU range is larger than normal, due to the nature of the project, where overlapping and partially hidden fruit are common, the wider range used to calculate the mAP become more appropriate. Additionally, during the experiment a less common IoU value of .1 is used, this further reflects the need to extend the mAP IoU calculation range, as this is where the model became most accurate. Overall, the model is suitable for the application, but further improvements are possible. To achieve this more images of sunstruck kiwifruits would need to be added to the training data, with more training data overall being beneficial. This would lower the need for the use of a low IoU value during inference. Ideally the model would perform best at a IoU of 0.5, showing an equal ability to detect true positives and true negatives at an equal rate, and should be aimed for in future iterations of the system, and may require a new camera angle. Additionally, the use of a CNN was chosen as a fundamental method of object detection not purely based on accuracy but verticality. The use of a SVM could be a better option for a system that only detects one or two classes. The reason CNNs were chosen as the fundamental method of object detection is due to their ability to detect many classes with high accuracy. With this project being a proof of concept in terms of the YOLO architecture, later iterations should focus on making full use of the architecture and add gold, green and red kiwifruit classes. Additionally further classes can be added for fallen or “ground” kiwifruit which could be of use to farmers in their pursuit of PA. Overall, the model performed well, but a higher mAP can be achieved and should be aimed for.

6.3.0 Areas of Improvement

Although all the systems used in the drone work to a level capable of producing a high accuracy result, many individual components, changed could further improve accuracy, flight time, safety, ease of use of the overall system. The following sections highlight important areas that could use improvement assuming this style of system was to be taken further in development.

6.3.1 Microprocessor

Although a mAP of 0.844 and an FPS of ~12.5 is a very good result, the overall result falls just short of a truly real time solution operating at 25-30 FPS. Such a result can in my opinion be achieved, with a simple addition of a Coral Edge TPU device, or a change of the microprocessor. A More powerful variant of the jetson nano such as the Nvidia Orin NX, which when compared to the jetson nano with 0.5TFLOPS of AI performance the Orin NX has ~ 21TOPS of AI performance. Although far out of budget for this project, the Orin NX would be an ideal candidate as it features the same footprint, and comparable architecture / operating system to the Jetson Nano. With such an upgrade allowing for a full 30FPS real time operation of the YOLOv7-Tiny model, additionally it may be able to handle a higher resolution, wider angle camera thus further increasing the accuracy of the system.

6.3.2 Camera

The GoPro-Hero3 Silver proved to be a good and reliable camera choice for this project, with a count of more than 20 drone crashes during multiple test flights throughout the project, the Go Pro survived all the crashes with just a few scratches. Its ability to change several settings related to resolution, field of view, megapixels, FPS, and low light. The ability to change the resolution down from 4k to 1080p, adjusting the field of view such in encompass the correct width of the orchard, as well as the FPS and low light settings to further optimise the system. The only downside of the camera is the inability to handle bad lens flare (when the sun strikes the lens) this causes a loss of contrast and a washed-out appearance. This was more noticeable on sunny days as caused issues with the accuracy of the object detection system. So, although the GoPro was a good choice, a higher quality camera able to better deal with changing light conditions would significantly improve the system overall.

6.3.3 Obstacle Avoidance / mm-Radar

The use of the mm-Radars/TFMini-s used for hight estimation and obstacle avoidance, provided both a cheap, long range, quality solution that was relatively easy to implement when compared to the other components on the drone. The use of the mm-Radar for hight estimation was one of the best decisions in this project. Due to the lack in accuracy of the altimeter at low latitudes, and the necessity to keep the drone at a steady hight result in the

necessity for a range finder to carryout hight estimation. Although other more capable solutions are on the market the TFMini-s performed well in this case. Additionally with the orchards on a slight angle, flying the drone under the orchards without it was more than problematic. Additionally, due to the nature of the sensor pointing down there didn't seem to be much noise introduced into the sensor. Using a second TFMini-s mm-radar for object avoidance also worked as expected, with the drone programmed to stop when within 1m of an obstacle. However, this can be somewhat annoying to use while the drone is in manual flight/ flown via remote control. This is due to noise introduced by the sun, low hanging leaves or the most common issue leaves from the ground that had been lifted into the air. Due to the use of a large drone the amount of air moved by the drone is also large. This causes any dead leaves that are not wet (and there is a good amount on the floor during picking season). These leaves cause the drone to momentarily stop whenever a leaf happens to fly in front of it for just a moment. These leaves also cause more issues discussed later. Due to this when flying remotely the obstacle avoidance sensor is a hindrance, and when not flying remotely it would be advisable that the floor is racked before flight under the orchards.

6.3.4 5V5A Regulator

The custom 5V5A linear regulator designed for the project worked as expected, cutting power from the LiPo before reaching the minimum 5V required to power the Jetson Nano. The use of the linear voltage regulator was not an ideal choice and was only used out of accessibility and necessity for the project. Future iterations should use a more efficient, high quality buck converter, capable of providing a smooth 5V5A, thus majorly reducing weight of the circuit board due to not requiring a large heatsink, as well as prolonging the battery life, making the system more capable.

6.3.5 Drone Platform

The Hexadrone platform was chosen for its inherent stability, capability to lift heavy loads, relatively small size, and successful use in similar projects in the past. The use of the Tarot 680 carbon fibre Frame did prove to perform well in these areas, as well as providing a large surface area for mounting of the many modules needed for a successful flight. However, there are likely better frames to choose for such a project in the future. The largest issue the tarot frame faces was the inability for truly fasten the motor mounts to the motor arms in a way that

completely prevents moving/rotating of the motor after a slightly hard landing or crash. This issue causes the user to constantly tune and retune the drone for the stability needed for such accurate flight. A better option for a similar project would be to use a standard four arm drone but utilising eight motors. For example, the Droidwork Skyjib X4 featuring 4 arms, with 2 motors mounted to each arm, with a smaller overall footprint, larger lifting power, and space to carry 2 large LiPos at a time. Such a drone would be more capable of flying under orchards, for longer periods of time with added weight of a higher resolution camera assuming price is a non-issue. Figure 63, [75].

Figure 63: Droidwork Skyjib X4 with dual batteries. [74].

6.3.6 FPV System

The FPV system used in the drone worked as expected, providing the user with a clear screen from which to remotely fly the drone from, the camera used for the FPV system is small, light and of high quality and does not require any improvement. The main advantage of the FPV system is that it allows the user to always view the drone from a front facing view. When flying the drone without a FPV camera, flying the drone can become confusing quickly as the pitch and yaw no longer correlate to the drone movement as the drone changes rotation. This is one important reason to implement a FPV system. Additionally, although autonomous flight via the Pixhawk is an option, remote flying allows for far superior accuracy of drone positioning in the orchard. The FPV system used in this project however has one major flaw, that was not considered during purchase. This is that the screen used to display the FPV output does not

run on battery power. Due to the orchard being in the middle of a farm the FPV system was not usable when doing flights under the orchard thus rendering it useless. Due to this, this aspect of the drone system that did not function in practice when compared to test flight at home. Additionally, instead of a screen to display the FPV output, a better option may be the use of battery powered FPV goggles. This would reduce the sun glare from the screen and may provide a better overall experience.

3.3.7 Autonomous Flight

While the autonomous flight/ mission planner itself is a usable and more than capable system, the GPS and surrounding GPS infostructure require both upgrading and further development such to allow GPS guided autonomous flight through the orchards. The current GPS setup onboard the drone system uses 2 GPS modules, one located in the Pixhawk Flight controller, and the second in the GPS/compass module mounted on the GPS stand. To further improve this system GPS triangulation should be implemented. GPS triangulation consists of mounting three identical GPS/compass modules in a triangle configuration as seen in Figure 5 by [9]. Such a setup may allow for an acceptable GPS tolerance within 2.1m. Furthermore, if this method of yield estimation is attempted in many parts of Europe, Asia, or America, it is likely that the region will be covered by SBAS, allowing for high accuracy GPS localisation via the use of a F9P GPS module. Although not tested the ability to perform GPS guided navigation through a kiwi orchard by a drone is very likely possible and should be investigated further.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This master's project aimed to demonstrate the effectiveness of a new kiwi orchard yield estimation technique, focusing on the utilization of neural networks. To assess the effectiveness of this new technique, the project can be divided into three main aspects: the accuracy of a neural network in identifying kiwi fruit, the drone's suitability as a robotic platform under a kiwi orchard, and the microcontrollers' capacity to perform real-time CNN object detection.

After experimentation, it can be concluded that all three fundamental aspects of the project can work in unison at an acceptable level to achieve real-time kiwifruit orchard yield estimation. Analysing these three aspects individually allows us to better understand the system's strengths, weaknesses, and areas that require improvement. Experimental results revealed that while many convolutional neural networks possess the accuracy required for kiwifruit object detection, their shortcoming lies in the speed of the networks. The YOLOv7 neural network, although a new iteration of the YOLO architecture, proved sufficient in both accuracy and speed for this project.

The Hexadrone platform used is capable of carrying all the necessary components but has some limitations. Primarily, the size of the Hexadrone restricts lateral movement under the orchard, resulting in a less safe and slower flight to prevent crashes. Additionally, the drone's size can cause leaves and other debris to be lifted into the air, reducing the stability of the flight—a crucial aspect of this project. Despite these shortcomings, the drone platform can perform yield estimation, albeit at a slower rate than desired.

The microprocessor used provided sufficient processing power for real-time object detection and can be upgraded via tensor cores to further enhance processing power without significantly adding to the weight. It can be considered a successful implementation for the project. Although all three aspects have room for improvement that could significantly enhance system performance, the current combination of software, hardware, and the robotic platform provides a proof of concept for further development.

7.2 Recommendations

To achieve an enhanced overall result, it is suggested to pursue two different approaches for further development. Firstly, improving the current system involves using a coaxial quadcopter to reduce the footprint while maintaining the drone's lift capacity. Secondly, employing a more powerful microcontroller capable of providing a higher FPS and, consequently, a faster flight speed would enhance both the accuracy of detection and the distance the drone can cover, such as with the NVIDIA Jetson Orin. Although YOLOv7 has proven to be a capable CNN, the latest YOLO iteration, namely YOLOv8, may offer further improvements in accuracy and inference time for the object detection system.

The camera used for object detection should be upgraded to reduce sun glare and improve visibility, especially during overcast days. This enhancement would significantly increase the accuracy and usability of the overall system. Finally, while the mm-radars have proven effective for height estimation and object avoidance directly in front of the drone, an improvement could be the use of a 360-degree Lidar, capable of reducing the lateral drift of the drone under the orchards.

This method of orchard yield estimation could lead to several fundamental changes based on the findings of this thesis. Firstly, to enhance the versatility, manoeuvrability, and capabilities of the drone platform, a smaller, more agile drone, specifically a freestyle drone, could be implemented. These drones are compact enough to be handheld and designed for navigating tight corners, flying under trees, and around obstacles. Using such a drone might enable users to fly under orchards without the need to clear tall grass or remove broken branches, minimizing the lift force required to prevent ground debris from becoming airborne.

Although this platform would not support an onboard microprocessor, this limitation could be overcome by using a portable laptop capable of receiving a live video feed from a secondary camera mounted at an upward-facing angle. This setup aims to further reduce occlusion of kiwifruits. Additionally, the laptop could perform real-time object detection given it possesses the necessary hardware capabilities. While this system differs significantly from the one used in this project, the fundamental concept of real-time kiwifruit orchard yield estimation via a drone is still achieved, potentially offering improved results.

Although such a method of performing kiwi fruit yield estimation, namely via a small drone, relaying video data to a server or personal computer to process the data in real time or even at a later stage was considered, this path was not taken intentionally. This is due to a few important reasons. Firstly, such a method of implementation is simply too basic and trivial for

a master's thesis. Secondly having a computer/ server process the data introduces complexities in terms of real-life implementation. Using such a method means farmers/customers must have a larger investment to implement the system. Have increased complexity regarding data transfer, requiring cellular network coverage or a strong Wi-Fi signal from a base station. Finaly such a method would results in a longer time to realise the orchard yield count. Having the drone system incorporate all aspects needed on the drone platform results in a cheaper and simpler method for the end user as well as a faster yield count result. The down side is that the design, construction and coding of such a system is more complicated. For these reasons the latter option, which focuses more heavily on mechatronic principles was decided to be the best fit for this projects scope.

References

[1] Kumar, A., H. Sharifi, and K.M. Arif, *Mobile machine vision development for urine patch detection.* 2016 23rd International Conference on Mechatronics and Machine Vision in Practice (M2VIP), 2016(1): p. 1-6.

[2] Cameron, S., *NZ Kiwifruit Growers Inc (NZKGI).* Productivity Commission, 2021: p. 1-7.

[3] Goverment, N.Z. *Agricultural production statistics: June 2020*. 07 May 2021.

[4] *Fact sheet: Cherry Leaf Roll Virus* 2019 [cited 2021; 1]. Available from: https://kvh.org.nz/assets/documents/Biosecurity-tab/Offshore-Risks/CLRV-fact-sheet-March-2023.pdf.

[5] Liu, Y., et al., *Multi-robot coordination in complex environment with task and communication constraints.* 2013. **10**(5): p. 229.

[6] Verma, J.K. and V. Ranga, *Multi-Robot Coordination Analysis, Taxonomy, Challenges and Future Scope.* Journal of Intelligent & Robotic Systems, 2021. **102**(1): p. 1-36.

[7] Dong, X., et al., *Extraction of information about individual trees from high-spatial-resolution UAV-acquired images of an orchard.* 2020. **12**(1): p. 133.

[8] Surekha, P., et al. *An Automatic Drone to Survey Orchards using Image Processing and Solar Energy*. in *2020 IEEE 17th India Council International Conference (INDICON)*. 2020. IEEE.

[9] Chen, C.-J., et al., *Identification of fruit tree pests with deep learning on embedded drone to achieve accurate pesticide spraying.* 2021. **9**: p. 21986-21997.

[10] Stefas, N., H. Bayram, and V.J.I.-P. Isler, *Vision-based UAV navigation in orchards.* 2016. **49**(16): p. 10-15.

[11] Team, H.S. *TAROT T960 HEXACOPTER KIT*. 2022; Available from: https://hobbystation.co.nz/tarot-t960-hexacopter-kit/

[12] Donmez, C., et al., *Computer vision-based citrus tree detection in a cultivated environment using UAV imagery.* Computers and Electronics in Agriculture, 2021. **187**.

[13] *DJI Spreading Wings 1000+ with A2 Flight Controller.*

[cited 2021; Available from: https://www.digitalcamerawarehouse.com.au/dji-spreading-wings-1000-with-a2-flight-controller.

[14] Koirala, A., et al., *Deep learning – Method overview and review of use for fruit detection and yield estimation.* Computers and Electronics in Agriculture, 2019. **162**: p. 219-234.

[15] Dutcosky, R., *Support Vector Machine (SVM) Practical Implementation.* 2020.

[16] Ray, S. *Learn How to Use Support Vector Machines (SVM) for Data Science.* 2017, September 12; Available from: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/.

[17] Haug, S. and J. Ostermann. *A Crop/Weed field image dataset for the evaluation of computer vision based precision agriculture tasks*. in *ECCV Workshops (4)*. 2014.

[18] Yunong Tian, E.L. and L.Y. , Zize Liang, *An image processing method for green apple lesion*

detection in natural environment based on GA-BPNN and

SVM. International Conference on Mechatronics and Automation, 2018.

[19] LüQiang, C.J., Liu Bin, Deng Lie, Zhang Yajing, *Identification of fruitand branchin natural scenes for citrus harvesting robot using machine vision and support vector machine.* International Journal of Agricultural and Biological Engineering, 2014. **7**(2): p. 7.

[20] Shetty, S., *Application of Convolutional Neural Network for ImageClassification on Pascal VOC Challenge 2012 dataset.* Manipal Institute of Technology, 2012: p. 6.

[21] *An Insight into YOLO Deep Learning*. 2020, July 4; Available from: https://indiantechwarrior.com/an-insight-into-yolo-deep-learning/.

[22] Ankushsharma. *YOLO V1/V2/V3 Architecture*. 2020, June 25; Available from: https://medium.com/@ankushsharma2805/yolo-v1-v2-v3-architecture-1ccac0f6206e.
[23] Zhang, H. and R.S. Cloutier, *Review on One-Stage Object Detection Based on Deep Learning.* EAI Endorsed Transactions on e-Learning, 2022. **7**(23).
[24] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. , *You only look once: Unified, real-time object detection.* Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016: p. 779–788.
[25] Michael Halstead, C.M., Simon Denman,Tristan Perez,Clinton Fookes, *Fruit Quantity and Quality Estimation using a Robotic Vision System.* 2018: p. 8.
[26] Sa, I., et al., *DeepFruits: A Fruit Detection System Using Deep Neural Networks.* Sensors (Basel), 2016. **16**(8).
[27] Häni, N., P. Roy, and V. Isler, *A comparative study of fruit detection and counting methods for yield mapping in apple orchards.* Journal of Field Robotics, 2019. **37**(2): p. 263-282.
[28] Baird, A. and S.J.a.p.a. Giani, *An Artificial Intelligence System for Combined Fruit Detection and Georeferencing, Using RTK-Based Perspective Projection in Drone Imagery.* 2021.
[29] Ren Wu, S.Y., Yi Shan, Qingqing Dang, Gang Sun, *Deep Image: Scaling up Image Recognition.* 2015.
[30] Shorten, C. and T.M. Khoshgoftaar, *A survey on Image Data Augmentation for Deep Learning.* Journal of Big Data, 2019. **6**(1).
[31] Li, M., et al., *Agricultural Greenhouses Detection in High-Resolution Satellite Images Based on Convolutional Neural Networks: Comparison of Faster R-CNN, YOLO v3 and SSD.* Sensors (Basel), 2020. **20**(17).
[32] T. GAYATHRI DEVI, D.P.N., S. SUDHA, *Image Processing System for Automatic Segmentation and Yield Prediction of Fruits using Open CV.* International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC), 2017: p. 758–762.
[33] Kathleen Anne M, A., Rona Mae G. Babaran, Bryan Jones C. Carzon, Karl Patrick K. Cu, Jasmine M. Lalanto, and Alexander C. Abad., *Autonomous Fruit Harvester with Machine Vision.* Journal of Telecommunication, Electronic and Computer Engineering, 2018. **10**: p. 1-6.
[34] *The DJI Zenmuse Series: Taking Footage to New Heights.* . 2022; Available from: https://drdrone.ca/blogs/drone-news-drone-help-blog/thedjizenmuseseriestakingfootagetonewheights.
[35] Lee, H.Y., H.W. Ho, and Y. Zhou, *Deep Learning-based Monocular Obstacle Avoidance for Unmanned Aerial Vehicle Navigation in Tree Plantations.* Journal of Intelligent & Robotic Systems, 2020. **101**(1).
[36] *Intel D435 Depth Camera, 0.2m Min Depth Distance, 10m Max Range* [cited 2022; Available from: https://nz.rs-online.com/web/p/depth-cameras/1720981.
[37] Darwish, W., et al., *A New Calibration Method for Commercial RGB-D Sensors.* Sensors (Basel), 2017. **17**(6).
[38] Basso, F., et al., *Fast and Robust Multi-people Tracking from RGB-D Data for a Mobile Robot*, in *Intelligent Autonomous Systems 12*. 2013. p. 265-276.
[39] Tu, S., et al., *Passion fruit detection and counting based on multiple scale faster R-CNN using RGB-D images.* Precision Agriculture, 2020. **21**(5): p. 1072-1091.
[40] Fu, L., et al., *Application of consumer RGB-D cameras for fruit detection and localization in field: A critical review.* Computers and Electronics in Agriculture, 2020. **177**.
[41] team, S.f. *TFMini-S - Micro LiDAR Module—SEN-16977*. 2022; Available from: https://www.sparkfun.com/products/16977.
[42] *What is lidar? Learn How Lidar Works.* . 2022; Available from: https://velodynelidar.com/what-is-lidar/.
[43] co, B., *Product Manual of TFmini, Mini LiDAR Module.* 2017: p. 28-30.

[44] *What is Sensor Fusion?* . 2021, October 21; Available from: https://appen.com/blog/what-is-sensor-fusion/.
[45] Song, H., W. Choi, and H. Kim, *Robust Vision-Based Relative-Localization Approach Using an RGB-Depth Camera and LiDAR Sensor Fusion.* IEEE Transactions on Industrial Electronics, 2016. **63**(6): p. 3725-3736.
[46] Huang, X., J.K.P. Tsoi, and N. Patel, *mmWave Radar Sensors Fusion for Indoor Object Detection and Tracking.* Electronics, 2022. **11**(14).
[47] Wardynski, D.J. *What is a Microprocessor and How Does it Work?* . October 10, 2019; Available from: https://www.brainspire.com/blog/what-is-a-microprocessor-and-how-does-it-work.
[48] Halfacree, G., *Google launches Edge TPU dev board, USB accelerator.* 2019.
[49] *M.2 Accelerator with Dual Edge TPU*. 2020; Available from: https://coral.ai/products/m2-accelerator-dual-edgetpu/.
[50] Safa, A., et al., *Exploring Cross-fusion and Curriculum Learning for Multi-modal Human Detection on Drones*, in *System Engineering for constrained embedded systems*. 2022. p. 1-7.
[51] team, e., *RPI4-MODBP-8GB.* 2021.
[52] Khoi, T.Q., N.A. Quang, and N.K. Hieu, *Object detection for drones on Raspberry Pi potentials and challenges.* IOP Conference Series: Materials Science and Engineering, 2021. **1109**(1).
[53] A Astina Joice, P.R., J Deepa, R Arulmari, *Colour discernment of tomatoes using machine vision system with OpenCV*

Python and Raspberry Pi. Indian Journal of Engineering & Materials Sciences, 2022. **29**: p. 502-508.
[54] *Coral Dev Board—4GB RAM Version*. 2022; Available from: https://www.seeedstudio.com/Coral-4GB-Dev-Board-Version-p-4683.html.
[55] Team, M.E., *Seeed Studio NVIDIA® Jetson Nano™ Developer Kits.* 2022.
[56] Jordà, M., P. Valero-Lara, and A.J. Peña, *cuConv: CUDA implementation of convolution for CNN inference.* Cluster Computing, 2022. **25**(2): p. 1459-1473.
[57] Nguyen, A.Q., et al., *A Visual Real-time Fire Detection using Single Shot MultiBox Detector for UAV-based Fire Surveillance*, in *2020 IEEE Eighth International Conference on Communications and Electronics (ICCE)*. 2021. p. 338-343.
[58] Wang, Q., L. Zhao, and Z. Niu, *Portable Kiwi Variety Classification Equipment Based on Transfer Learning.* Journal of Physics: Conference Series, 2021. **1865**(4).
[59] Assunção, E., et al., *Real-Time Weed Control Application Using a Jetson Nano Edge Device and a Spray Mechanism.* Remote Sensing, 2022. **14**(17).
[60] Unal, H.B., et al., *Fruit Recognition and Classification with Deep Learning Support on Embedded System (fruitnet)*, in *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*. 2020. p. 1-5.
[61] Huang, H., et al., *Design of Citrus Fruit Detection System Based on Mobile Platform and Edge Computer Device.* Sensors (Basel), 2021. **22**(1).
[62] Geetha, S., et al., *EMI shielding: Methods and materials-A review.* Journal of Applied Polymer Science, 2009. **112**(4): p. 2073-2086.
[63] *Glycol-Modified Polyethylene Terephthalate (PETG, PET-G)*. 2020 05 30 [cited 2022; Available from: https://www.makeitfrom.com/material-properties/Glycol-Modified-Polyethylene-Terephthalate-PETG-PET-G.
[64] O'Connell, J. *Understand PETG vs. PLA to see if your next project calls for the more durable PETG or the easier-to-use PLA.* 10 february 2023; Available from: https://all3dp.com/2/petg-vs-pla-3d-printing-filaments-compared/.
[65] Zagidullin, R.S., N.I. Zezin, and N.V. Rodionov, *Improving the quality of FDM 3D printing of UAV and aircraft parts and assemblies by parametric software changes.* IOP Conference Series: Materials Science and Engineering, 2021. **1027**(1).
[66] Pierre-Jean Bristeau, P.M., Erwan Salaün, Nicolas Petit, *The role of propeller aerodynamics in the model of a quadrotor UAV.* European Control Conference 2009, 2009: p. 683-688.

[67] Team, A.D. *Advanced Compass Setup*. 2022; Available from: https://ardupilot.org/copter/docs/common-compass-setup-advanced.html.
[68] Santos, F.A., et al., *Low velocity impact response of 3D printed structures formed by cellular metamaterials and stiffening plates: PLA vs. PETg.* Composite Structures, 2021. **256**.
[69] Team, A.D., *Magnetic Interference—Copter documentation.* 2022.
[70] Tyflopoulos, E. and M. Steinert, *Topology and Parametric Optimization-Based Design Processes for Lightweight Structures.* Applied Sciences, 2020. **10**(13).
[71] *RTK GPS Correction (Fixed Baseline)*. 2022; Available from: https://ardupilot.org/copter/docs/common-rtk-correction.html.
[72] *How SouthPAN benefits New Zealand*. 2022; Available from: https://www.linz.govt.nz/products-services/geodetic/southpan/how-southpan-benefits-new-zealand.
[73] Hu, C., et al., *Automatic detection of pecan fruits based on Faster RCNN with FPN*

in orchard. International Journal of Agricultural and Biological Engineering, 2022. **15**(6): p. 189-196.
[74] Wang, C.-Y. and A.B. , and Hong-Yuan Mark Liao, *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object*

detectors. 2022: p. 15.
[75] H.Balta, J.B., S.Govindaraj, K.Majek, P.J.Musialik, D.Serrano, k. Alexis, R.Siegwart, G.D.Cubber., *Integrated Data Management for a Fleet of Search-and-rescue Robots.* Journal of Field Robotics, 2016. **34**: p. 539-582.

Appendix

R calculations:

> mean_value <- 139

> sd_value <- 9.67

> basket_count <- 147

> cdf_120 <- pnorm(120, mean = mean_value, sd = sd_value)

> cdf_156 <- pnorm(156, mean = mean_value, sd = sd_value)

> probability_range <- cdf_156 - cdf_120

> print(probability_range)

[1] 0.9359111

>

> estimated_total <- probability_range * basket_count * mean_value

> print(estimated_total)

[1] 19123.47

>

> # Calculate the standard error

> standard_error <- sd_value / sqrt(basket_count)

> print(standard_error)

[1] 0.7975682

>

> critical_value <- qnorm(0.975, mean = mean_value, sd = sd_value)

> print(critical_value)

[1] 157.9529

>

> margin_of_error <- standard_error * critical_value

> print(margin_of_error)

[1] 125.9782

>

> # Calculate the lower and upper bounds of the confidence interval

```
> lower_bound <- round(estimated_total - margin_of_error)
> upper_bound <- round(estimated_total + margin_of_error)
>
> # Display the estimated total and the confidence interval
> cat("Estimated Total: ", round(estimated_total), "\n")
```

Estimated Total: 19123

```
> cat("95% Confidence Interval: [", lower_bound, " - ", upper_bound, "]\n")
```

95% Confidence Interval: [18997 - 19249]

Generate Graph:

```
> x <- seq(mean_value - 4 * sd_value, mean_value + 4 * sd_value, length.out = 100)
> y <- dnorm(x, mean = mean_value, sd = sd_value)
> plot(x, y, type = "l", xlab = "Fruit Count", ylab = "Probability Density", main = "Normal
Distribution of Fruits per Basket")
```

```
> mean_value <- 139
> sd_value <- 9.67
> basket_count <- 80
>
> cdf_120 <- pnorm(120, mean = mean_value, sd = sd_value)
> cdf_156 <- pnorm(156, mean = mean_value, sd = sd_value)
> probability_range <- cdf_156 - cdf_120
> print(probability_range)
[1] 0.9359111
>
> estimated_total <- probability_range * basket_count * mean_value
> print(estimated_total)
[1] 10407.33
>
```

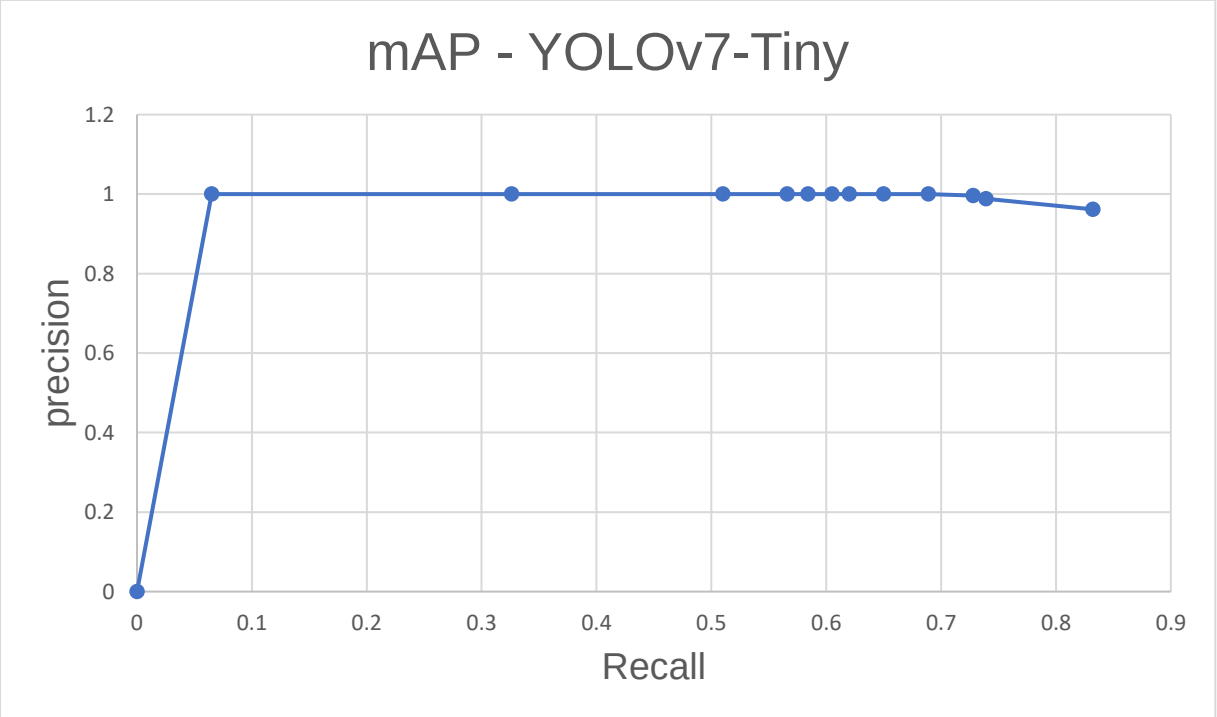
```
> # Calculate the standard error
> standard_error <- sd_value / sqrt(basket_count)
> print(standard_error)
[1] 1.081139
>
> critical_value <- qnorm(0.975, mean = mean_value, sd = sd_value)
> print(critical_value)
[1] 157.9529
>
> margin_of_error <- standard_error * critical_value
> print(margin_of_error)
[1] 170.769
>
> # Calculate the lower and upper bounds of the confidence interval
> lower_bound <- round(estimated_total - margin_of_error)
> upper_bound <- round(estimated_total + margin_of_error)
>
> # Display the estimated total and the confidence interval
> cat("Estimated Total: ", round(estimated_total), "\n")
Estimated Total:  10407
> cat("95% Confidence Interval: [", lower_bound, " - ", upper_bound, "]\n")
95% Confidence Interval: [ 10237  -  10578 ]
```

Image Number	#kiwis /IMG	YOLOv7 Conf 0.5	True +	False +	False -	Average confidence	precision	recall	F1
1	46	29	29	0	17	82.6	1	0.63043478	0.773333
2	59	43	42	1	17	90	1	0.71186441	0.831683
3	31	21	21	0	10	88.2	1	0.67741935	0.807692
4	9	5	5	0	4	88.2	1	0.55555556	0.714286
5	14	7	7	0	7	83.3	1	0.5	0.666667
6	51	24	24	0	27	85.4	1	0.47058824	0.64
7	43	16	16	0	27	85	1	0.37209302	0.542373
8	21	13	13	0	8	91	1	0.61904762	0.764706
9	31	16	16	0	15	88.9	1	0.51612903	0.680851
10	29	18	18	0	11	85.3	1	0.62068966	0.765957

Image Number	#kiwis /IMG	YOLOv7-Tiny 0.5	True +	False +	False -	Average confidence	precision	recall	F1
1	46	28	28	0	18	83.7	1	0.60869565	0.756757
2	59	42	42	1	17	92.4	0.976744	0.71186441	0.823529
3	31	24	24	0	7	85.6	1	0.77419355	0.872727
4	9	7	7	0	2	81.2	1	0.77777778	0.875
5	14	7	7	0	7	90.3	1	0.5	0.666667
6	51	28	28	0	23	88	1	0.54901961	0.708861
7	43	17	17	0	26	86.5	1	0.39534884	0.566667
8	21	14	14	0	7	93	1	0.66666667	0.8
9	31	18	18	0	13	81.4	1	0.58064516	0.734694
10	29	17	17	0	12	88.7	1	0.5862069	0.73913

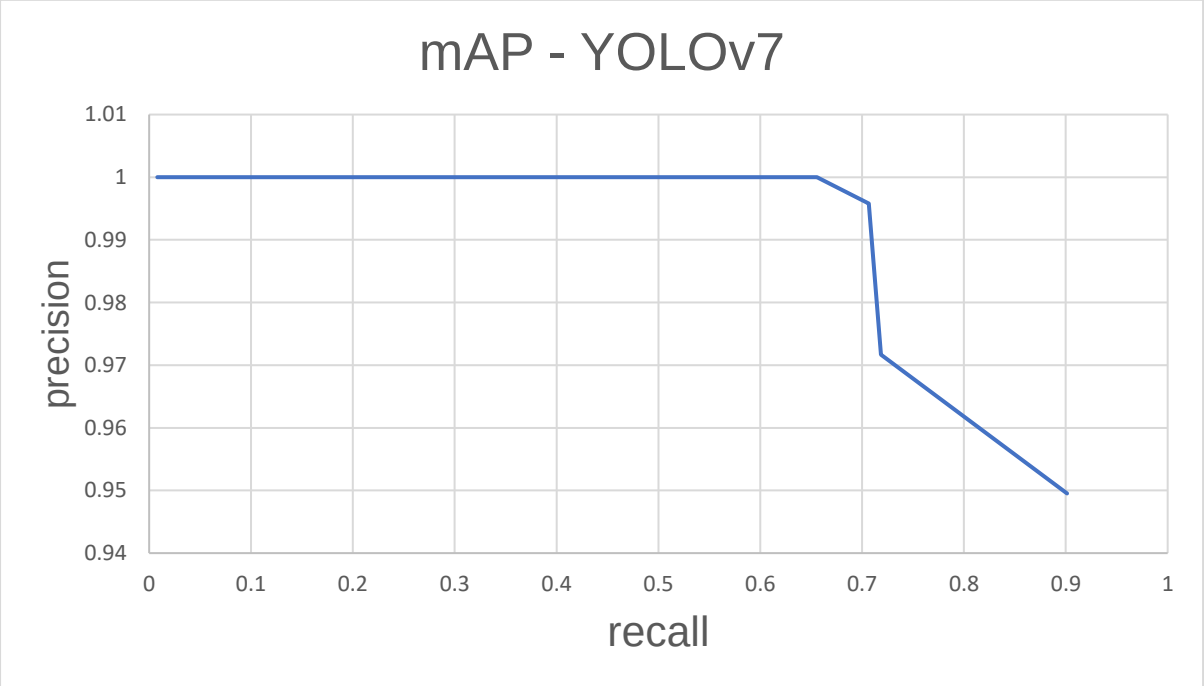
Image Number	#kiwis /IMG	FRCNN Conf 0.5	FRCNN True +	False +	False -	Average confidence	precision	recall	AP = 0.699 F1
1	46	19	19	0	27	94.7	1	0.41304348	0.584615
2	59	20	19	1	40	100	0.95	0.3220339	0.481013
3	31	17	16	1	15	97.7	0.941176	0.51612903	0.666667
4	9	5	5	0	4	99.8	1	0.55555556	0.714286
5	14	6	6	0	8	99	1	0.42857143	0.6
6	51	15	15	0	36	84.1	1	0.29411765	0.454545
7	43	14	13	1	30	82.7	0.928571	0.30232558	0.45614
8	21	16	16	0	5	97.4	1	0.76190476	0.864865
9	31	15	15	0	16	93.3	1	0.48387097	0.652174
10	29	17	17	0	12	78.3	1	0.5862069	0.73913

Yolov7-tiny conf	#KIWIs	True +	False +	False -	Prediction conf	precision	recall	F1
.005	334	278	11	56	61.63	.962	.832	.892
.05	334	247	3	87	75.5	.988	.739	.846
.1	334	243	1	91	81.17	.996	.728	.816
.2	334	230	0	104	83.5	1	.689	.816
.3	334	217	0	117	84.2	1	.650	.787
.4	334	207	0	127	85.1	1	.620	.765
.5	334	202	0	132	86.4	1	.605	.754
.6	334	195	0	139	88.9	1	.584	.737
.7	334	189	0	145	90.5	1	.566	.722
.8	334	170	0	164	92.4	1	.510	.675
.9	334	109	0	225	95.3	1	.326	.492
.95	334	22	0	312	97.1	1	.065	.124
.995	334	0	0	334	0	0	0	0
Average	334	177.6	1.25	156.4	78.6	.919	.576	.741



mAP of YOLOv7-Tiny = .8442

YOLOv7 Conf	#kiwi/img	True +	False +	False -	Average prediction conf	precision	recall	F1
.005	334	301	16	33	58	0.949527	0.901198	0.924731
.05	334	240	7	94	71.4	0.97166	0.718563	0.826162
.1	334	236	1	98	71.4	0.995781	0.706587	0.82662
.2	334	219	0	115	75.1	1	0.655689	0.792043
.3	334	211	0	123	75.1	1	0.631737	0.774312
.4	334	198	0	136	75.1	1	0.592814	0.744361
.5	334	192	0	142	83.3	1	0.57485	0.730038
.6	334	188	0	146	87.5	1	0.562874	0.720307
.7	334	182	0	152	87.5	1	0.54491	0.705426
.8	334	158	0	172	87.5	1	0.478788	0.647541
.9	334	83	0	215	90	1	0.278523	0.435696
.95	334	3	0	331	0	1	0.00898	0
.995	334	0	0	334	0	0	0	0



YOLOv7 mAP = 0.898

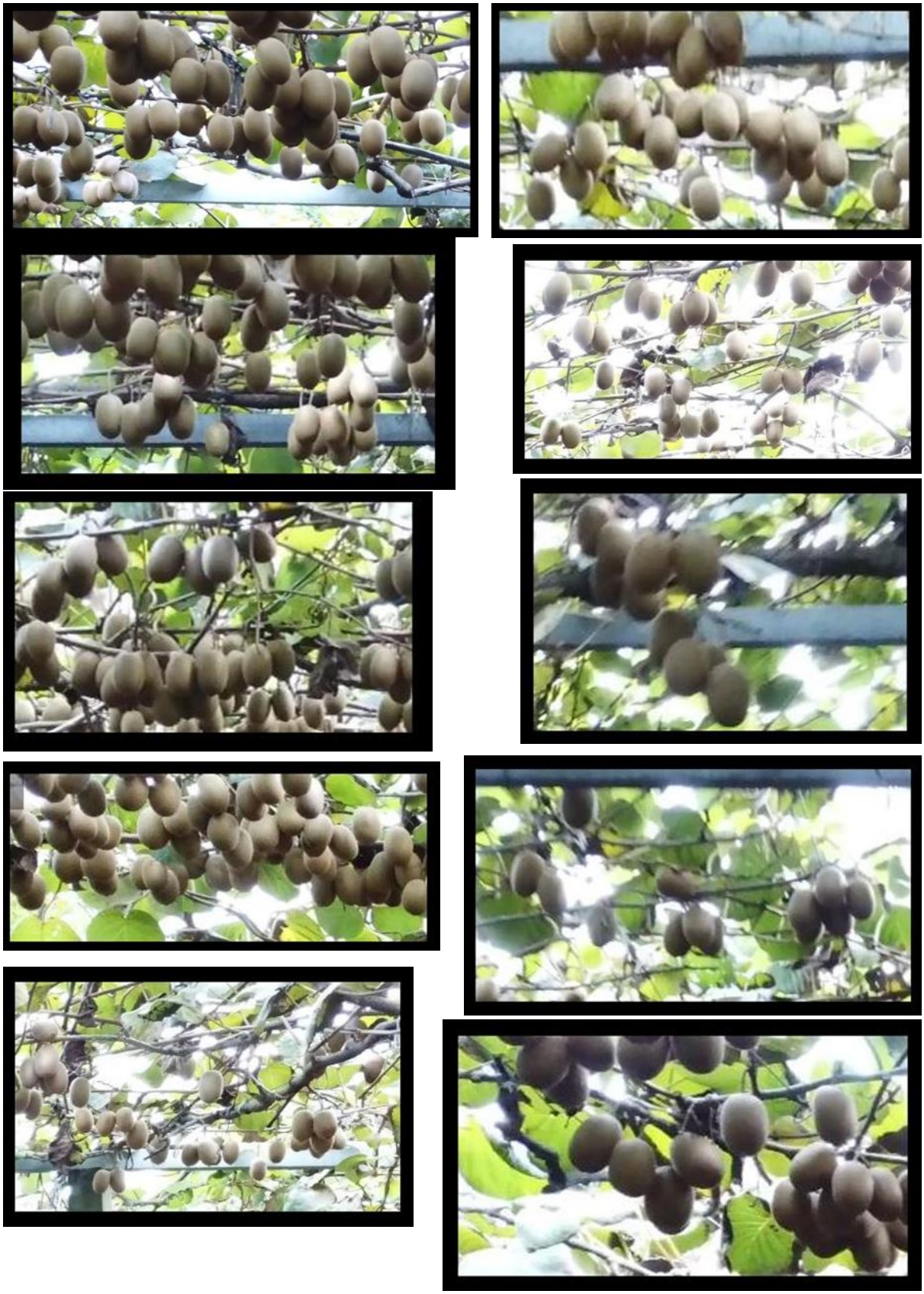


Figure 64: mAP Test Images (Padding Removed)

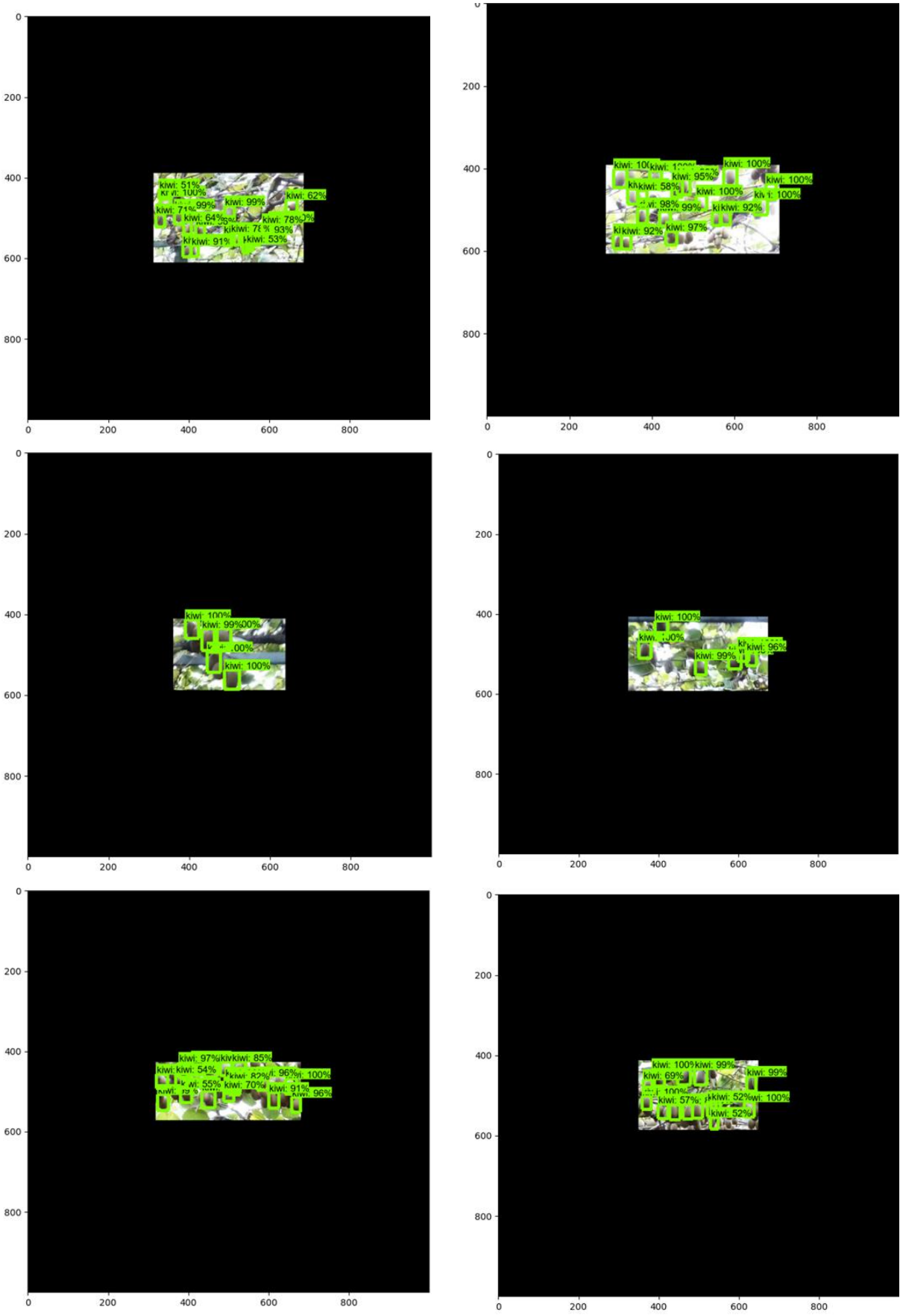


Figure 65:selection of FasterRCNN mAP test images. (Sunstruck image top right as shown for YOLOv7 in Chapter 4 for comparison.)

Figure 66: Early drone test flight

```
#!/bin/bash

trap 'echo "script terminated by leon." ; deactivate ; exit 1' SIGINT

source /home/leon/.virtualenvs/yolov7/bin/activate

cd /home/leon/yolo/yolov7

python detect_count_and_track.py --weights YOLOv7Tiny.pt --conf 0.1 --img-size 640 --source 0 --view-img --no-trace &

#loop until user presses 'q' key

while true

do

        echo "press 'Q' to quit."

        read -n 1 input

        if [[ $input == "q"]]

        then

                echo "Exiting script."

                deactivate

                exit 0

        fi
```

Figure 67: Jetson nano .sh script with Q command

Figure 68: Drone View from above the orchards