http://researchcommons.waikato.ac.nz/

Research Commons at the University of Waikato

Copyright Statement:

The digital copy of this thesis is protected by the Copyright Act 1994 (New Zealand).

The thesis may be consulted by you, provided you comply with the provisions of the Act and the following conditions of use:

- Any use you make of these documents or images must be for research or private study purposes only, and you may not make them available to any other person.
- Authors control the copyright of their thesis. You will recognise the author's right to be identified as the author of the thesis, and due acknowledgement will be made to the author where appropriate.
- You will obtain the author's permission before publishing any material from the thesis.

Occlusion Prediction For Kiwifruit Yield Estimation

A thesis

submitted in fulfilment

of the requirements for the Degree

of

Doctor of Philosophy in Engineering

at

The University of Waikato

by

Matthew Seabright

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

2020

Abstract

Kiwifruit is very important to the New Zealand economy, bringing in NZ$2.39 billion in the 2017/2018 season and projected to grow to NZ$4.5 billion by 2025. Every season, the expected yield from each orchard is estimated. These estimates are used for both on-orchard decisions regarding thinning and spraying, and industry level decisions regarding export pricing, supply agreements, labour requirements, packaging quantities and more. The current method for yield estimation is manual and prone to errors. In the 2016/17 growing season, green kiwifruit estimates were 5.9% lower than actual harvest yield, resulting in over 5 million trays (approximately 18,000 tons) of fruit being disposed of.

An automated system could reduce errors saving the industry significant amounts of money. Such a system could realise other benefits such as providing more granular information on orchard performance to growers and increasing labour efficiency and crop uniformity via targeted labour. Other automated fruit yield estimation systems have been developed but have suffered from sub-optimal performance due to occlusion of fruit. The work presented here introduces a system that has been developed with the aim of minimising occlusion where possible and employs an occlusion prediction model to account for remaining occlusion. The system utilises stereo vision, a 3D lidar unit, an inertial measurement unit and LED lighting mounted on an all terrain vehicle (ATV). Data is captured across 16 maturity areas, consisting of 33.9 Ha of commercial kiwifruit orchards. A simultaneous localisation and mapping system is used to locate the ATV within the orchard. A convolutional neural network detects kiwifruit, which are localised via stereo triangulation. Individual fruit are tracked through overlapping images to reduce occlusion while ensuring no double counting. Fruit are counted and an occlusion prediction model is applied to account for the unseen fruit.

The system is able to predict orchard scale fruit yield with a mean absolute percentage error of 0.68%. This error rate was obtained from an independent test dataset and ground truth data obtained from a commercial kiwifruit packhouse. The system can also generate visualisations of orchards showing all individual detected fruit in 3D, fruit density maps and canopy density maps. The ability of the system to predict yield with high accuracy

and at orchard scale sets it apart from previous work. It is also the largest scale study currently known, consisting of over 15 million fruit.

Embargo

This document is subject to a three year embargo from the date of acceptance. This embargo is to protect the interests of the commercial partner of the work (Robotics Plus Limited).

Acknowledgements

I would like to thank the following people:

- My family; Mum, Dad, Scott and Kat for their support.
- The magnificent Mark Jones for answering all of my questions and teaching me many things.
- All the members of the CoHort team including, Jamie, Henry, Josh, Michael, Hamish, Canaan, Phil, Gordon, Mahla, Bruce, Peter and Paul for their support, advice and friendship.
- Everyone at Robotics Plus, past and present including, Alistair, Steve, Daniel, Flo, Henri, Erin, Luke, Kyle, Matthew 2, Dan, Oscar, Chris, Nick, John, Demler, Mel, Akshay, Bram, Carl, Ben, Will, Shiloh, JG, Andy and Andrew for providing a fantastic workplace and supporting me for the last few years.
- My supervisors; Mike, Lee, Michael and Rachael for their advice, support and the many lunches at Nourish.
- MBIE for providing funding for this work.
- Andrew Scott, the rest of the Gro-Plus team and the orchard owners for giving advice and allowing me into their orchards.
- Authors and contributors of open-source applications/libraries that were relied upon heavily. To name a few, ROS, OpenCV, Tensorflow, Mask R-CNN, PCL, the Linux kernel, Inkscape, GIMP, LaTeX, Python, NumPy, Scikit-learn.

Thank you all very much!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Kiwifruit is New Zealand's largest horticultural export (2016–2018 [1–3]) bringing in NZ$2.39 billion in the 2017/2018 season [4]. The industry is projected to grow even larger with revenue expected to rise to NZ$4.5 billion by 2025. New Zealand kiwifruit is exported to 59 countries and makes up a third of the countries total horticulture export revenue [5].

There are a two main issues stifling the growth of the New Zealand kiwifruit industry. The first is labour shortages, particularly in the kiwifruit harvesting season. The second is the errors seen with the current manual yield estimation system that are causing significant costs for the industry. The current manual system is to send orchard workers into orchards, who then count fruit by hand.

An automated yield estimation system can provide more thorough and accurate estimates of yield than the current manual system. It can also provide a higher degree of information granularity and new ways for growers to visualise the performance of their orchards. This extra information could allow more efficient and specific management decisions to be made, increasing production. A high performing system may also enable techniques to increase labour efficiency.

Other researchers have built automated yield estimation systems (Chapter 2) but have not been able to produce consistently accurate results. The main factor preventing high accuracy is fruit occlusion. Fruit that are not visible

Figure 1.1: The giant kiwifruit located near the town of Te Puke in the Bay of Plenty, New Zealand. The Bay of Plenty is the main kiwifruit growing region of New Zealand, containing 80.7% of New Zealand's crops [4]. The giant slice of kiwifruit shows the importance of kiwifruit to the region's economy.

are not able to be counted. Correcting for the under-counting is not trivial as the ratio of occluded to unoccluded fruit is not consistent from plant to plant or orchard to orchard. This thesis explores the use of an occlusion prediction model to dynamically predict the occlusion ratio and improve yield estimation accuracy.

1.1 Yield Estimation

One of the many steps in the kiwifruit production process is yield estimation. Yield estimation is the counting of buds, flowers or fruit to predict the number of fruit that will be present at harvest time. It is conducted at multiple stages throughout the growing process. The two main counts are completed in December (shortly after fruitset), and February (when fruit are still small).

The counts produced via yield estimation are used throughout the supply chain to aid in a variety of planning and decision making processes. Orchard

managers use the data for on-orchard decisions like thinning and spraying strategies. Packhouses use the information for projecting labour, packaging and coolstore requirements. Zespri, the company that exports the majority of New Zealand's kiwifruit overseas, use the data to formulate export pricing strategies, supply agreements and plan shipping logistics.

The current method for yield estimation is to send people into orchards who count buds/flowers/fruit in small sections of the orchard. Depending on the preferences of the orchard manager, between 1% and 2% of the crop is counted (some orchard managers may choose to not count at all). Counts are then extrapolated to the full area of the orchard. For this method to be accurate, the counting must be conducted with care and the area surveyed must be representative of the orchard as a whole. However, this is often not the case. On an industry wide scale, yield estimations are compiled for use by Zespri. These estimates can be more than 10% away from the actual harvest yields (Figure 1.2). Over the past three growing seasons, the most accurate February yield estimation for either green or gold fruit was over 4.5% away from actual yield[1].

When yield estimates are incorrect, inefficiencies in the supply chain and sub-optimal pricing strategies reduce grower profits. For example, when the total crop is underestimated, fruit may be 'crop managed'. The term 'crop managed' is used to describe fruit that are destroyed, sold as stock feed or otherwise not exported as they normally would be. Zespri employs this strategy in some seasons due to a variety of factors and economic reasons. In the 2016/17 and 2017/18 seasons, over 6 million total trays of green fruit were crop managed [6, 7]. This crop management resulted in a decrease in grower payments (OGR) of NZ$0.34 per tray or 7.2% in the 2016/17 season [6]. In 2016, a Zespri spokeswoman was quoted as saying; "Crop management isn't a new thing - the industry has used it in three of the past eight seasons" [8]. Although errors in yield estimation are not always solely responsible for crop management, the

[1]By personal communication

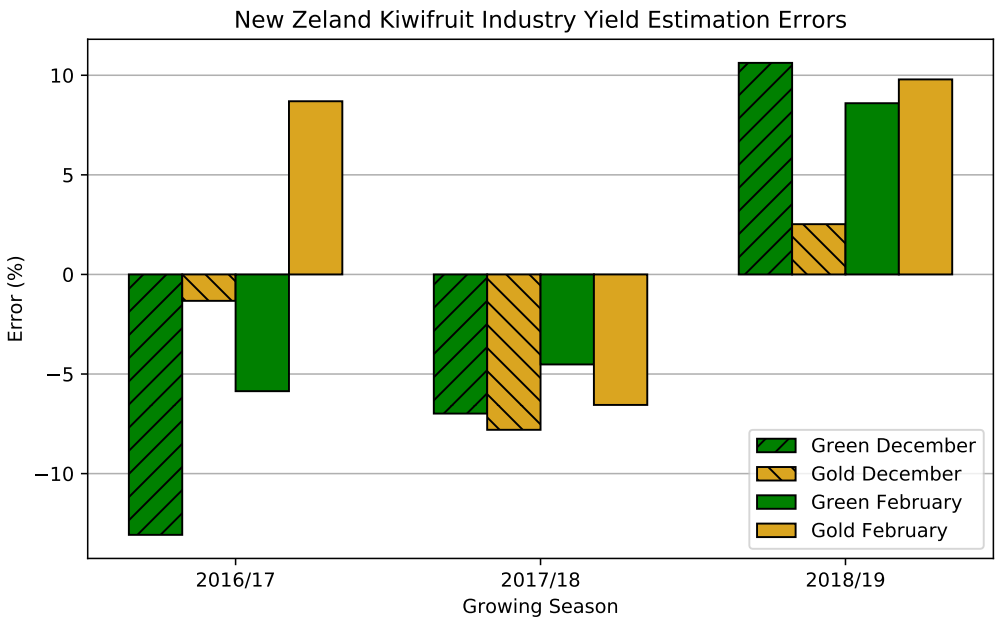


Figure 1.2: The yield estimation errors for green and gold fruit over three growing seasons. Yield estimations are performed in December and February. The errors are relative to the harvested yield.

following was stated in the Trevelyan's grower newsletter in September 2018; "Going forward, Zespri has indicated that more accurate industry crop estimation will ensure they can better manage market demand-to-supply and would likely reduce their requirement to introduce crop management" [9].

1.2 Labour Shortage

In 2018, the New Zealand government declared a 1200 person labour shortage in the kiwifruit industry [10]. The number of seasonal workers required to harvest and process the crop is expected to increase 45% between 2017 and 2027 [11]. The labour issues will continue to worsen if labour saving solutions are not found.

Yield estimation has a negligible labour requirement compared to tasks such as harvesting and pruning [11]. However, yield estimation can be used to maximise labour efficiency in these labour intensive tasks. For example, a natural extension to yield estimation is enabling targeted thinning. Thinning

is the process of removing a subset of buds, flowers or fruit to maximise the orchards productivity. However, variability within an orchard means that the one area may require significant thinning, while another requires no thinning. Targeted thinning would, hypothetically, direct orchard workers to the areas of the orchard that require thinning. It would then inform the workers exactly how much thinning is required to reach the desired bud, flower or fruit density. This augmentation of human labour would both reduce overall labour requirements and increase crop uniformity, resulting in higher yield and lower costs.

Currently, harvesting and thinning accounts for 45% of on-orchard labour requirements, while pruning accounts for 47% [11]. Automated solutions for harvesting are being investigated by multiple groups around the world [12–17]. Finding viable automated solutions to these problems would significantly reduce the required labour force. Many of the technologies required for yield estimation (particularly object detection and localisation on an orchard scale) are also required for automated harvesting, thinning and pruning systems. Thus, yield estimation can be seen as a stepping stone toward automated harvesting and pruning solutions.

1.3 Background

This section is intended to provide an overview of kiwifruit growing practices and the New Zealand kiwifruit supply chain. The contents of this section are assumed prior knowledge for the remainder of the document.

1.3.1 Growing Kiwifruit

There are two varieties of kiwifruit commonly grown in New Zealand, Hayward and Gold3 (Figure 1.3). The Hayward variety is commonly referred to as ‘green’ kiwifruit. Green kiwifruit have a furry brown skin, green flesh and a sweet tangy flavour. The Gold3 variety (often abbreviated to G3) are

Figure 1.3: Left, two Hayward (green) kiwifruit. Right, two Gold3 (gold) kiwifruit.

commonly referred to as 'gold' kiwifruit and are marketed by Zespri under the 'SunGold' name. Gold kiwifruit have a smooth brown skin, yellow-gold flesh and a sweet flavour. There are other varieties of kiwifruit grown in New Zealand (such as Green14 and Hort16A), but Hayward and Gold3 make up 94.6% of the total Zespri crop, as of 2018 [4]. This document refers to the Hayward and Gold3 varieties 'green' and 'gold' kiwifruit respectively.

Kiwifruit orchards are split into sections called blocks. A block is a continuous area of kiwifruit planting, ranging in size from a fraction of a hectare, to multiple hectares. Blocks are often separated by shelter belts, which are high thin trees that shelter the kiwifruit plants from wind. Kiwifruit orchards are also split into maturity areas. Often, multiple blocks will make up a maturity area, but they can be just one block. Typically, blocks are grouped into a

Figure 1.4: An orchard shortly before harvest. Trunks and posts are arranged in rows. The canopy contains the fruit and foliage in a flat structure approximately 1.8 m from the ground.

maturity area based on factors such as geographic similarities and the age of the plants. Maturity areas can contain a maximum of 60,000 trays of kiwifruit (see Section 1.3.2 for further explanation). All of the fruit in a maturity area, should reach maturity at approximately the same time, as they can only be harvested once a sample of the fruit have passed a maturity test. If sections of a maturity area mature later than other, potential earnings can be lost while waiting for fruit to mature as fruit harvested earlier can earn a higher payout for the grower. Orchards range in size from less than a hectare, with a single maturity area consisting of one block, to hundreds of hectares with many maturity areas, each split into multiple blocks.

Modern kiwifruit orchards use the pergola growing system. The pergola growing system has the fruit growing in a flat, horizontal canopy approximately 1.8 m from the ground (Figure 1.4). Plants and posts are alternated to form rows. The width of rows varies between orchards with most between 3 m and 6 m wide.

There are three main structural components of the pergola growing system, posts, beams and wires (Figure 1.5). Posts are placed vertically into the ground

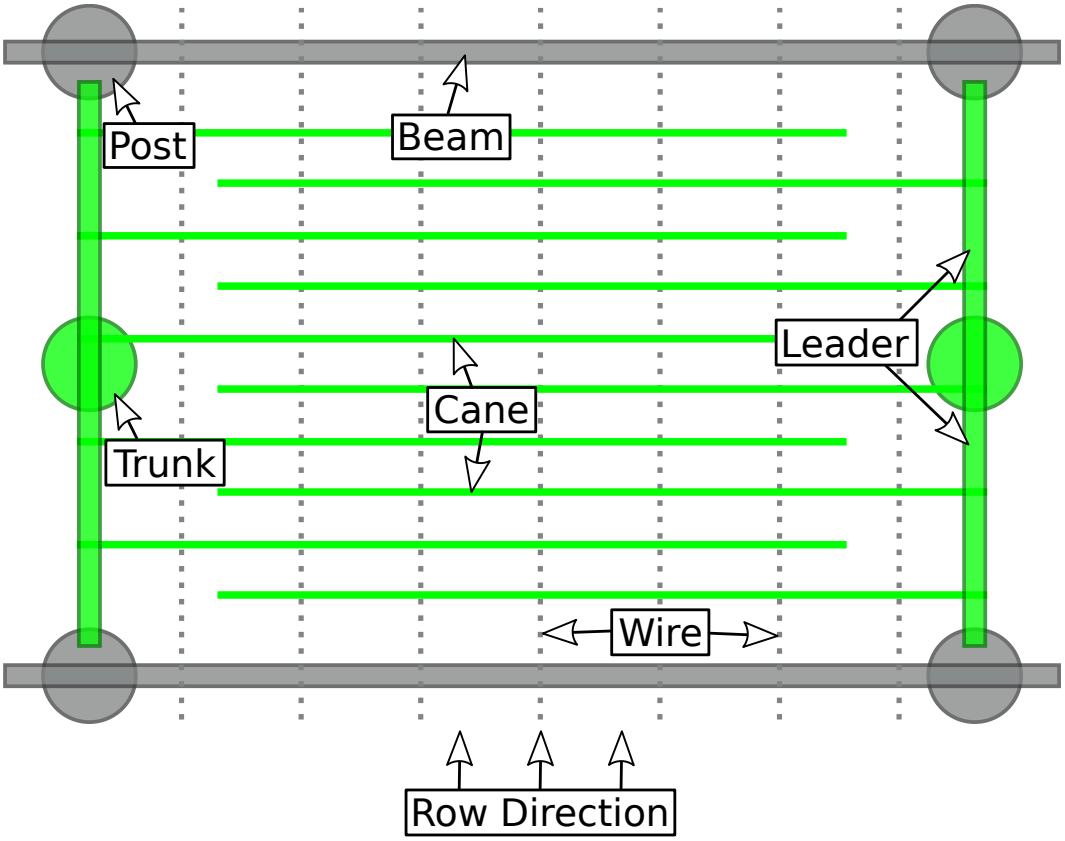


Figure 1.5: A diagram showing the structure of the pergola kiwifruit growing system from a top down view. The row runs vertically through the diagram. Green features represent the plants, grey is the orchard structure. The area between four posts is called a 'bay'.

and support the canopy. Beams sit on top of the posts and run across the width of rows. Beams are usually made of either galvanised steel or wood. Wires run the length of the rows, sitting on top of the beams, and have a spacing of 100 mm to 500 mm between them.

The trunk of the plants goes from the ground, up into the canopy (Figure 1.6). From the top of the trunk, two leaders extend in opposite directions, down the length of the row. The leaders support canes, which are tied to the wires and run across the row from each side. Fruit and leaves grow on small shoots that come off the canes.

Kiwifruit plants each have a sex. Only the female plants produce fruit. The male plants produce pollen that is required to pollinate the female flowers and allow them to grow into fruit. Growers plant approximately one male for

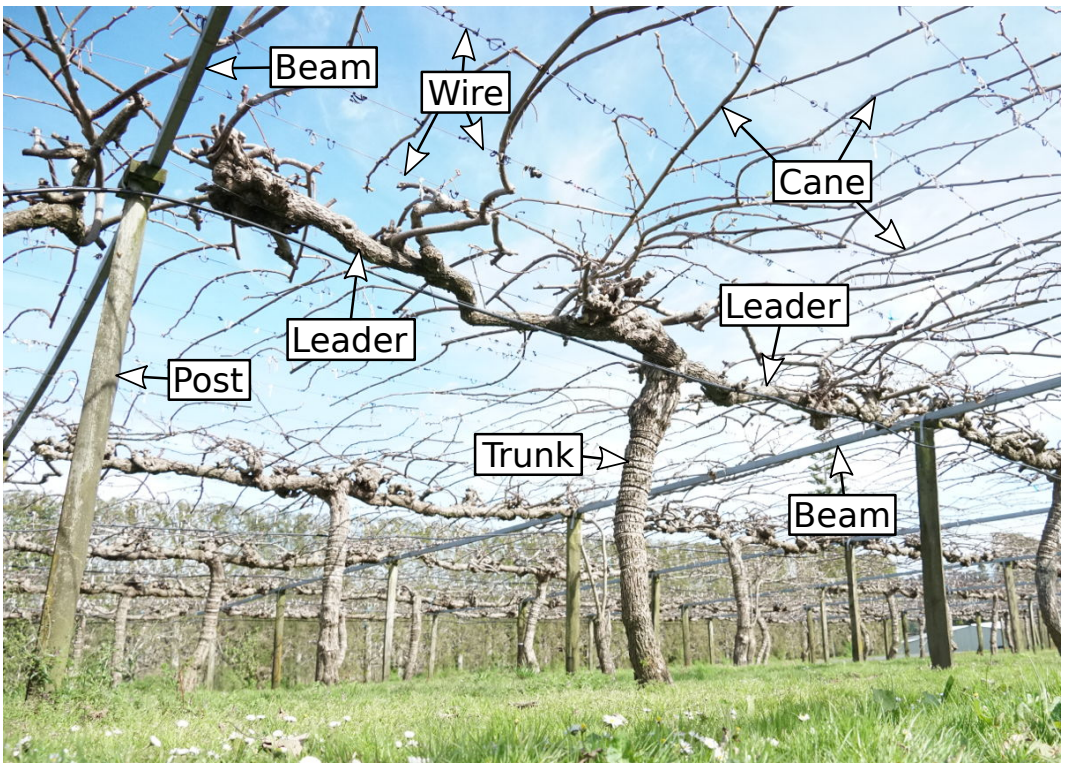


Figure 1.6: A kiwifruit plant in an orchard in early spring, before foliage starts to grow.

every eight females, although this ratio varies significantly between orchards. Some orchards are female only, relying entirely on supplementary pollen, while some others are male only, used to supply pollen to other orchards.

In New Zealand, the kiwifruit growing cycle is the following: In September and October, new buds emerge and grow. In November, the buds open revealing flowers. The female flowers are pollinated with pollen from the male flowers. In December the pollinated flowers begin to develop into fruit. These fruit grow in size over the following months and reach maturity between March and June, when they will be harvested. After the harvest, winter pruning and other maintenance is performed at various stages before the next growing season begins.

1.3.2 New Zealand Kiwifruit Supply Chain

There are three main types of entity that make up the New Zealand kiwifruit supply chain. The first is the grower, who produces the fruit and manages the

orchard. The second is the packhouse, which grades and packs the fruit. The third is Zespri, who export and markets the majority of fruit overseas.

The packhouse grades each piece of fruit as class 1, class 2 or reject. Class 1 fruit are the highest quality and are generally exported by Zespri. Class 2 fruit are lower quality fruit which go into the New Zealand and Australian markets. Rejects are fruit that do not meet the class 1 or 2 criteria because of defects or size.

Kiwifruit are generally packed into trays. All trays contain approximately 3.6 kg of fruit. The number of fruit on a tray is determined by the count size, which is a measurement of fruit size. Standard count sizes range from 18 to 41. The count size is the number of fruit of that size that fit onto a tray. For example, a count size 25 tray will have 25 fruit on it and weight approximately 3.6 kg. Fruit production is typically measured in trays, as opposed to raw fruit numbers as trays accounts for fruit size.

1.4 Research Questions

This thesis addresses two main issues with fruit yield estimation systems, detailed in Chapter 2:

Appearance variation Variation in fruit size, shape and colour, lighting and distance to camera make detection of fruit difficult.

Fruit occlusion Occlusion due to leaves, branches, orchard structure and other fruit cause some fruit to be partially or completely invisible to an imaging system.

Appearance variation has largely been overcome in recent years with the use of advanced convolutional neural networks (CNN) such as SSD, YOLO, and Faster R-CNN coupled with high performance graphics processing units (GPU).

Fruit occlusion has been an issue that many researchers have attempted to overcome by using correction factors. These correction factors are usually obtained via regressing the number of fruit detected against harvest counts. This provides significant improvements in accuracy and eliminates under-counting bias, but does not account for all the variation between orchards/vineyards/-plants.

Ideally, each orchard/vineyard/plant would have its own correction factor. These individualised correction factors would be adjusted based on the properties of the orchard/vineyard/plant they belong to. For example, an orchard with high fruit density should have a higher rate of occlusion, and hence a higher correction factor, when compared to an orchard with a lower fruit density. Figure 1.7 demonstrates how fruit density effects occlusion rate. The same can be said for fruit clustering, in that an orchard where fruit are heavily clustered would have a higher rate of occlusion, when compared to an orchard with uniform fruit distribution. Foliage density and other factors could also effect the rate of occlusion. These factors can be measured whilst counting fruit, allowing an occlusion prediction model to produce individualised correction factors providing more accurate yield estimations.

The hypothesis that is tested by this work is the following: An occlusion prediction model will produce dynamic correction factors calculated based on measurable characteristics of an orchard. These individualised correction factors will provide more accurate predictions of yield than a static correction factor.

To test this hypothesis, a new yield estimation system is designed, implemented and tested. The yield estimation system is used in kiwifruit orchards to predict harvest yield on an orchard scale. To minimise the effects of occlusion, multiple viewpoints are used by having significant overlap between images taken from a moving platform. To negate the double counting introduced by using multiple viewpoints, a novel fruit tracking system, capable of localising individual fruit within an orchard context, is employed. The fruit

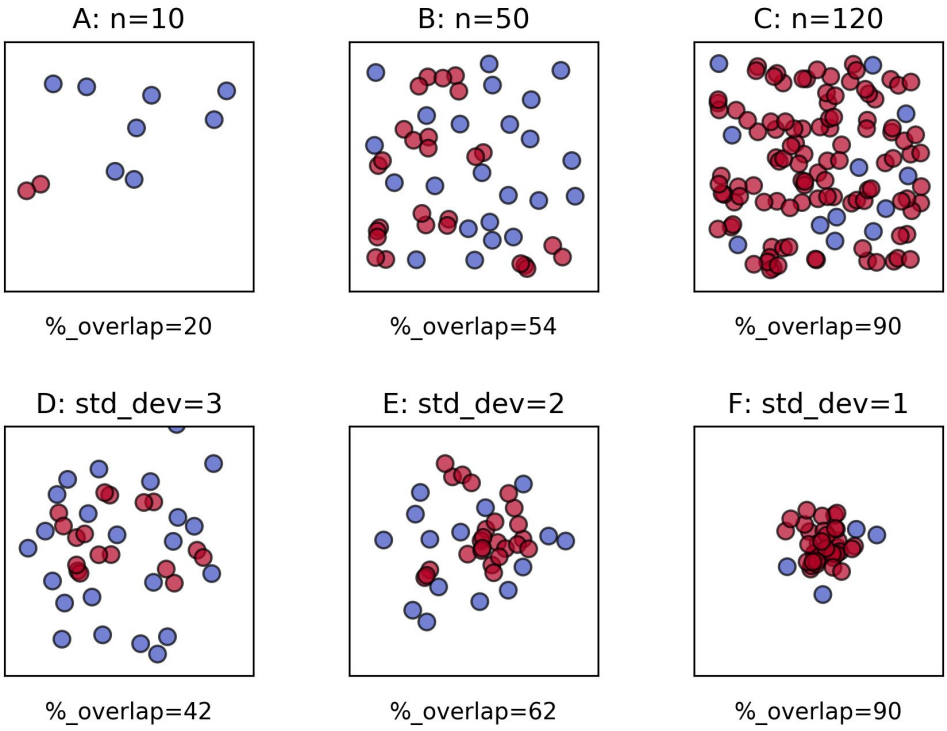


Figure 1.7: A demonstration of how fruit density and fruit distribution effect occlusion rates. Circles are coloured red if they overlap another circle, blue if they do not. The top row shows an increasing number of randomly placed circles. Each panel in the bottom row has 40 circles distributed in a normal distribution about the centre, with decreasing standard deviation.

tracking system utilises stereo vision and lidar based simultaneous localisation and mapping (SLAM) to position each fruit within the orchard. Parameters that correlate with occlusion rate are measured and used for the occlusion prediction model. Final yield estimates are calculated and compared to ground truth data.

1.5 Thesis Structure

This thesis begins with background on the kiwifruit industry and how yield estimation is important to the process. The research aims of the work are then outlined along with an overview of how other researchers have approached solving similar problems. The bulk of the document is divided into the various components that make up the overall yield estimation system. Figure 1.8 shows each component and how data flows between them. Conclusions about the work and how the research questions have been answered follow. A thorough outline of potential changes, modifications and extensions that could be applied to this system or others, concludes the document.

1.6 Publications

The following is a list of publications resulting from this work:

- M. Duke, J. Barnett, J. Bell, M. H. Jones, P. Martinsen, B. McDonald, M. Nejati, A. Scarfe, P. Schaare, M. Seabright, H. Williams, J. Lim, and H. Ahn, Automated Pollination of Kiwifruit Flowers, in *7th Asian-Australasian Conference on Precision Agriculture (7ACPA)*, (Hamilton, New Zealand), pp. 15, November 2017. Author's contributions: Design of electronics, software and mechanical systems, field testing.

- M. Seabright, L. Streeter, M. Cree, M. Duke, and R. Tighe, Simple Stereo Matching Algorithm for Localising Keypoints in a Restricted Search Space, in *2018 International Conference on Image and Vision*

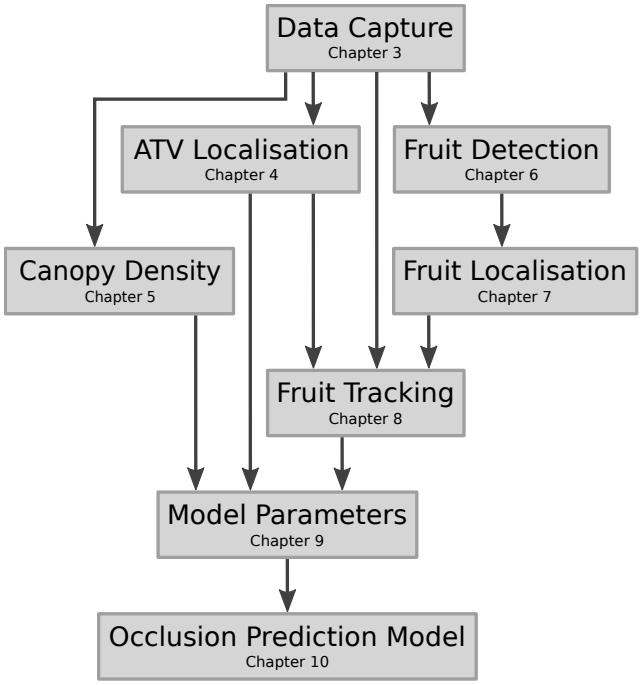


Figure 1.8: Diagram showing the flow of data through the various components of the yield estimation system. The chapter in which each component is described is shown.

Computing New Zealand (IVCNZ), vol. 2018-Novem, (Auckland, New Zealand), pp. 16, IEEE, November 2018. Author's contributions: Design, implementation and testing of stereo matching algorithm.

- H. A. Williams, M. H. Jones, M. Nejati, M. J. Seabright, J. Bell, N. D. Penhall, J. J. Barnett, M. D. Duke, A. J. Scarfe, H. S. Ahn, J. Lim, and B. A. MacDonald, Robotic kiwifruit harvesting using machine vision, convolutional neural networks, and robotic arms, *Biosystems Engineering*, vol. 181, pp. 140156, May 2019. Author's contributions: Design of electronics, software and mechanical systems, field testing.

- H. Williams, M. Nejati, S. Hussein, N. Penhall, J. Y. Lim, M. H. Jones, J. Bell, H. S. Ahn, S. Bradley, P. Schaare, P. Martinsen, M. Alomar, P. Patel, M. Seabright, M. Duke, A. Scarfe, and B. MacDonald, Autonomous pollination of individual kiwifruit flowers: Toward a robotic kiwifruit pollinator, *Journal of Field Robotics*, pp. 1-17, January 2019. Author's contributions: Design of electronics, software and mechanical

systems, field testing.

- H. Williams, C. Ting, M. Nejati, M. H. Jones, N. Penhall, J. Lim, M. Seabright, J. Bell, H. S. Ahn, A. Scarfe, M. Duke, and B. MacDonald, Improvements to and largescale evaluation of a robotic kiwifruit harvester, *Journal of Field Robotics*, vol. 0, July 2019. Author's contributions: Software for robotic arm control, system testing, end effortor design.

- M. Jones, J. Bell, D. Dredge, M. Seabright, A. Scarfe, M. Duke, and B. MacDonald, Design and Testing of a Heavy-Duty Platform for Autonomous Navigation in Kiwifruit Orchards, *Biosystems Engineering*, vol. 187, 2019. Author's contributions: Design and building of hardware and software for power genreation system.

Chapter 2

Literature Review

A modern machine vision based fruit yield estimation system consists of multiple subsystems. The first is an imaging system to collect images of plants. This imaging system can consist of a single hand-held camera, an autonomous robot carrying many sensors, or anything in between. The second subsystem is a detection system to detect the fruit in the images. These detection systems range from simple colour thresholding systems, to convolutional neural networks (CNNs). Modern systems then include a localisation subsystem to position each fruit either within the context of a single plant, or within an entire orchard/vineyard context. Localisation systems vary from overlap based algorithms to GPS based systems with stereo vision. The final section is a yield estimation model that transforms the number of fruit seen (and possibly other information) into an yield. The approach of other researchers in each of these areas is analysed to inform the design of the kiwifruit yield estimation system.

2.1 Machine Vision Based Fruit Yield Estimation Systems

Machine vision fruit yield estimation systems have been a focus of research for two decades. Systems have been developed for counting many types of

fruit including apples [18–28], grapes [29–31], mangoes [32–37], kiwifruit [38], capsicum, [39, 40], almonds [22, 41] and dragonfruit [42]. Over the last five years, focus has been shifting from increasing the accuracy of fruit detection, to increasing the accuracy of yield estimation.

2.1.1 Imaging Sensors

For detecting fruit on plants, the sensor of choice is the colour camera with the majority of studies using industrial or digital single-lens reflex (DSLR) cameras [18–25, 29, 30, 32–35, 38, 42–70] whereas Aquino et al., Gong et al. and Qian et al. used cellphone cameras [26, 71, 72]. A yield estimation system based on a cellphone is an enticing prospect for growers as it would allow quick and easy measurement without requiring extra equipment. However, the lack of computational power and imaging consistency are significant disadvantages to this approach.

Stajnko et al. used a thermal camera to capture the thermal gradient between apples and leaves in the afternoon [27]. However, detecting fruit near the inner sections of the tree, was difficult due to the fruit being shaded from the sun by leaves. Underwood et al. used a vertically oriented lidar to estimate the canopy volume of almond trees [41]. By combining the canopy volume estimations with a colour camera, better yield estimation accuracy was achieved than by using the camera or lidar in isolation. Stein et al. also used a vertical lidar and a camera [36]. However, they used the lidar to mask out the areas of each image not containing the tree being measured. The mask was created by first segmenting each tree in the lidar point cloud, then projecting the resulting segmentation onto the images. Sa et al. used both a colour and an infra-red camera to provide an extra input channel to their detection system for sweet pepper other fruit [39, 40]. Wang et al. compared the performance of a depth camera (RGB-D), a stereo vision system and a time of flight (ToF) camera for on-tree estimation of mango size [37]. The authors recommend RGB-D cameras based on there low cost and high performance, although the authors

note the performance of the camera was poor in direct sunlight.

2.1.2 Imaging Conditions

The ideal yield estimation system could be used at any time of day as it would be robust to varying lighting conditions. In reality, many researchers have chosen to image at night with artificial lighting to ensure consistency [19, 20, 24, 31, 33, 35, 38, 40, 43, 46, 58–60, 64, 65, 73]. This negates the changes in light levels, colour temperature and background colour, aiding detection. However, a system that can only be used at night has severely limited practical utility. Using artificial lighting during the daytime can decrease the variability of lighting conditions. Some researchers employ artificial LED or halogen lighting for day time imaging [19, 20, 28, 29, 40, 46, 47, 58, 59]. Some even use the specular reflection from their lighting systems as a feature to aid detection [19, 20, 46, 47, 58, 59].

Avoiding false detection caused by fruit in adjacent rows has led some researchers to include an artificial background behind plants or fruit (Figure 2.1) [18, 23, 30, 53, 72]. Dorj et al. went a step further and manually masked out background pixels from their images [50, 51]. With some crops, it is possible to build an imaging platform that straddles a row of plants [28, 39, 40]. This approach provides a consistent backdrop and rejects natural light, giving greater consistency to images (Figure 2.2). It also allows imaging from both sides of the plant in a single pass.

Cameras and other sensors can be mounted and moved through orchards/vineyards in multiple ways. Some researchers used hand-held sensors [26, 41, 50–52, 60, 71, 72]. Holding cameas by hand is a low cost option that gives flexibility in its usage, however imaging consistency depends on the operator. Mounting the sensors on a tractor or other farm vehicle gives a higher degree of consistency compared to hand-held and is hence a popular option for researchers [19, 20, 28, 29, 33, 46, 47, 64]. An operator is still required to navigate the vehicle along the prescribed path and control the speed of the vehicle, but

Figure 2.1: Grapes on the vine with black paper used as an artificial backdrop. (Source: Aquino et al. 2017 [72]. Used with permission.)

Figure 2.2: The yield estimation hardware used by Gongal et al. straddles a row of apples, blocking out light, providing a consistent backdrop and allowing imaging from both sides of the plant. (Source: Gongal et al. 2016 [28]. Used with permission.)

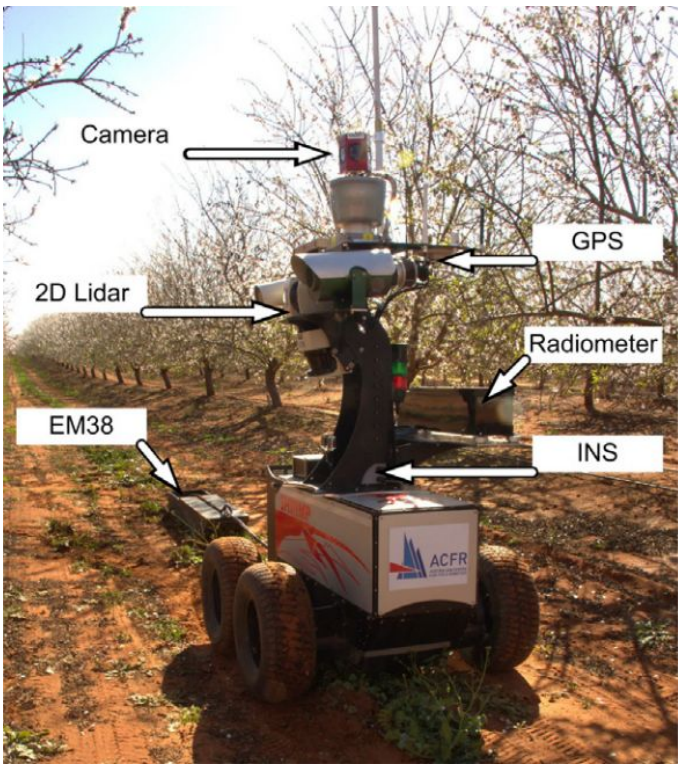


Figure 2.3: The 'Shrimp' mobile robot used by researchers at the University of Sydney [22, 36, 41, 55, 56]. (Source: Hung et al. 2015 [55]. Used with permission.)

the orientation and height of the sensors is fixed. With a vehicle, more sensors can be used and more plants can be covered in a given time compared to hand-held sensors. For the greatest consistency, an autonomous platform can be used such as the 'Shrimp' (Figure 2.3) [22, 36, 39–41, 45, 55, 56, 74]. Jones et al. presented an autonomous, multi-purpose, mobile platform (AMMP) specifically designed for use in kiwifruit orchards (Figure 2.4) [75]. The AMMP has been used for both harvesting and pollination of kiwifruit, but not yield estimation [15, 16, 76, 77]. Automated control over the path and velocity ensures consistent positioning of the sensors.

2.1.3 Detection Methods

Convolutional neural networks (CNN) and high performance graphics processing units (GPU) are now the standard for many computer vision tasks. It is no surprise to see these techniques applied to fruit detection systems. Before

Figure 2.4: The autonomous, multi-purpose, mobile platform presented by Jones et al. [75]. The AMMP is shown with pollination equipment mounted.

the rise of CNNs, detection methods were generally hand engineered or used machine learning and traditional image processing techniques.

2.1.3.1 Hand Engineered

Before the popularisation of convolutional neural networks for object detection, fruit detection algorithms were either crafted manually or used machine learning based classifiers with manually defined feature detectors. Most use a colour filter designed to separate fruit pixels from background pixels as the first step. Researchers have used many colour spaces including RGB [23,28,71], HSV [19,20,52,62,62,63], YCbCr [50,51] and L*a*b [37,38,43,48] with some using a combination of multiple colour spaces [26,32,33,64,65]. Because of variation in fruit colour, light levels, lighting colour temperature and fruit maturity, colour thresholds have to be kept wide to avoid false negatives. However, in some cases, a lack of colour differentiation between fruit and foliage/branches, means that wide colour thresholds will cause a high false positive rate. For example, Wijethunga et al. report a segmentation system that uses a minimum distance classifier on the a and b channels of the L*a*b colour space for use

detecting kiwifruit [38]. They show the system misclassifying some leaves and wooden orchard structure as kiwifruit, due to the similarity in colour.

Nuske et al. presented an algorithm for detection of white grapes, where there is little to no colour difference between the grapes and the leaves [29]. A radial symmetry transform was used to identify potential locations of grapes in images. Each potential point was then characterised using a 34-dimensional vector consisting of colour and texture (Gabor filters) information. The k-nearest neighbours algorithm was then applied to classify each point as a grape or background. This approach is less reliant on consistent colours than a simple colour filter method and was able to achieve a very high grape detection precision of 98% across three grape varieties. However, recall was much lower at 64%, caused by a high proportion of grapes not being identified by the radial symmetry transform (61% of false negative rate).

Some researchers have utilised the strong specular reflections seen when applying artificial lighting to fruit to aid detection [58, 59]. Linker and Kelman presented such an algorithm for use in detecting apples at night. Initially, bright spots in the image are identified as the centres of regions of interest. Each region is iteratively expanded by including adjacent pixels with a predefined decreasing level of grey intensity. The geometry of the expanding region is used to classify each region as an apple or background. In 60% of images, the false positive fraction was 5–25%, while in 50% of images, the false negative fraction was 25–35%. False positives mostly occurred on leaves (90%). False negatives were mainly due to partial occlusion (60%) and apples that appeared too small in the images (25%).

Diago et al. presented a system that classifies pixels using the Mahalanobis distance [30]. The system is applied to detecting grapes, leaves and branches in vineyards. The user must first select pixels representative of each class from a series of images. The algorithm will then classify pixels in new images based on the Mahalanobis distance of the pixel to the labelled samples. The system achieved an R^2 value of 0.78 between leaf pixels and observed leaf area and

0.76 between grape pixels and harvest yield.

Wang et al. used a cascade classifier with histogram of oriented gradients (HOG) features for mango detection [37]. The system first splits the image into small boxes and calculates the orientation of the brightness gradient through the box. The gradient orientations of a series of boxes are then used to classify the area as a mango or not using a cascade classifier. As the authors were estimating fruit size (as opposed to counting mangos), the algorithm was tuned to have a very low false positive fraction by adding an ellipse fitting step. The system was able to detect mangos with zero false positives, but a low true positive fraction (77%) over a series of validation images.

For use in a kiwifruit harvesting robot, Scarfe developed a detection system [13]. First a colour filter is applied to segment fruit coloured pixels then a Sobel filter is applied to detect edges. Next, areas of the image that have more than a set number of fruit coloured pixels around the perimeter of a fixed sized circle are identified. Of those areas, those that do not contain enough edge pixels within the circle are disregarded. From there, the distribution of edge pixels is analysed. Only those with the majority of edge pixels distributed nearer to the perimeter of the circle than the centre are kept. Finally, the aspect ratio of the identified region is measured. Those near a 1-to-1 aspect ratio are considered fruit. The system was able to correctly identify 69.55% of the 821 fruit in a set of test images. The system had a low false positive fraction at only 1.83%.

2.1.3.2 Convolutional Neural Networks

Over the last five years, many researchers have based fruit yield estimation systems around convolutional neural networks [22, 24, 35, 36, 39, 56, 60, 68, 70, 74,78,79]. Convolutional neural networks (CNN) such as ResNet [80], VGGNet [81] and ZFNet [82] have proven far more robust to image variation than hand engineered methods. Object detection frameworks such as YOLO [83], SSD [84] and Faster R-CNN [85] can provide accurate locations of multiple instances

of fruit in an image. This has significantly improved detection accuracy and largely solved the problems of image variation.

Bargoti and Underwood compared the performance of a multi-layered perceptron (MLP) and a CNN for detecting apples [56]. The CNN was a custom implementation built around work done on classification of electron microscopy images and urban scenes. With both of the pixel wise segmentation algorithms, they trailed the addition of image metadata such as sun position and tree type. Both networks saw an improvement with the inclusion of metadata, but it was negligible with the CNN. Overall, the CNN outperformed the MLP with pixel-wise F_1-score of 0.79 compared to 0.75 for the MLP. When both algorithms were followed by a watershed detection algorithm to identify individual fruit, the F_1-score of the CNN was 0.86, compared to 0.84 for the MLP.

To continue their earlier work, Bargoti and Underwood applied Faster R-CNN to detection of mangoes, almonds and apples [22]. The ZF and VGG16 networks were both applied and compared within Faster R-CNN with VGG16 performing slightly better. They achieved very good performance with an F_1-score $>$0.9 for both apples and mangoes (their previous method achieved an F_1-score of 0.86 on the same apple dataset [56]). Performance with the almond dataset was lower than apples or mangoes which the authors attribute to the small size of the almonds and large number of almonds in each image. For both mango and apple detection, performance increased when the training images were augmented via scale changes and orientation flips. Detection time averaged 0.13 second per image when using 500×500 pixel resolution.

Sa et al. built a capsicum detection system using Faster R-CNN [39]. Their data had an NIR channel as well as the traditional RGB colour channels. They tested two methods of combining the data for use with Faster R-CNN. The first, which they call 'early fusion', inserted the extra channel at the input of the CNN which was modified accordingly. The second method, 'late fusion', consisted of two independently trained neural networks, one for colour and one for NIR. The outputs of these two networks were combined to give the final

output. They compared these methods to both the RGB and NIR networks on their own. The early fusion network performed slightly worse than the RGB only network with F_1-scores of 0.80 and 0.82 respectively. The late fusion network performed the best with an F_1-score of 0.84.

Rahnemoonfar and Sheppard used simulated images of tomatoes to train a CNN they call DeepCount, to detect tomatoes in real images [79]. The simulated images consisted of randomly generated red-orange blobs on a mottled green-brown background and subjectively, did not look much like tomatoes. The CNN they used is a modified version of Inception-ResNet. They tested the network on images of real tomatoes they found via Internet searches. They authors claim an average accuracy of 91% on the real images compared to manual counts.

Chen et al. used a fully convolutional network (FCN) to detect oranges and apples [24]. The network used was that of Shelhamer et al. [86] which performs pixel-wise semantic segmentation. When detecting apples, the network achieved a true positive rate of 0.96 and false positive rate of 0.33.

Lamb and Chuah used Single Shot Multibox Detector (SSD) to detect strawberries [67]. SSD can place bounding boxes around each instance of a class in an image. It was trained with an Nvidia GTX 1080 Ti, starting with pretrained weights. To increase the speed of inference, images were first preprocessed and only the portions of each image that contained enough red pixels were passed to the CNN. They were able to run their final network on a Raspberry Pi 3B (Raspberry Pi Foundation, Cambridge, United Kingdom) at 1.63 frames per second with an average precision of 0.842, which is very impressive.

Halstead et al. used Faster R-CNN to detect capsicum [74]. To add ripeness detection to the CNN, they tested two methods. The first was to use a class for each of the three levels of ripeness they wanted to detect. The second was to add a ripeness classification layer to the output of Faster R-CNN. The multi-class approach gave poor results with F_1-scores between 0.47 and 0.73

at intersection over union (IoU) 0.4, depending on the class. The ripeness classification layer method performed better with an F_1-score of 0.77 at IoU 0.4. Ripeness evaluation was correct in 63–94% of instances, depending on the ripeness level.

Dias et al. used a hybrid approach to detect apple flowers [70]. Simple linear iterative clustering (SLIC) was used to segment areas of input images. SLIC is a superpixel segmentation algorithm that groups pixels using k-means clustering based on colour and spatial proximity. Each segmented area is fed into Clarifai CNN, which is a variant of ZFNet [82]. Rather than taking the usual output of the network, they take the output of the first fully connected layer, a 4096 dimensional vector. Principal component analysis is then applied to reduce dimensionality to 69, which encapsulates approximately 94% of the original variance. A support vector machine is then used to classify each segment. The authors chose this approach as opposed to a system such as Faster R-CNN as theirs can learn in an unsupervised manner. Their methods achieved an F_1-score of 0.82 on a validation dataset. When applied to peach flowers with no modification, the system achieved an F_1-score of 0.80, showing the network generalises well.

Koirala et al. compared multiple versions of Faster R-CNN, SSD and YOLO, as well as their own network, MangoYOLO, on detecting mangoes [35]. The MangoYOLO network is a compromise between the larger, more accurate but slower YOLOv3 and the smaller, less accurate but faster YOLOv2(tiny). MangoYOLO has 33 layers compared to 106 in YOLOv3 and 16 in YOLOv2(tiny). Their testing showed MangoYOLO to be the most accurate with an F_1-score of 0.968, with Faster R-CNN scoring between 0.929 and 0.945 dependant on configuration. SSD achieved an F_1-score of 0.950 to 0.959 and YOLO between 0.900 and 0.951. When comparing inference speed, YOLOv2(tiny) was fastest with an average of 10 ms per image. MangoYOLO averaged 15 ms, with Faster R-CNN ranging from 37 to 67 ms. Overall MangoYOLO performed the best with both a high F_1-score and low inference time. The authors also compared

the training performance of Faster R-CNN on two sets of images of the same mango orchard. One set of images was taken during the day while the other at night with artificial lighting. Training converged faster when using the night-time image set. The authors attribute this to the lower complexity and greater consistency of the images in the night-time image set compared to the day-time dataset.

Heinrich et al. compared Faster R-CNN, SSD and Region-based Fully Convolutional Network (R-FCN) for detection of grapes [78]. They trained each of the networks with three classes; grape bunches, wooden structure and metal structure. SSD performed very poorly with a mean average precision (mAP) of 0.58 at IoU of 0.5. Faster R-CNN performed best with a mAP of 0.996 at IoU 0.5, with R-FCN scoring 0.991.

Bellocchio et al. proposed a weakly supervised counting approach for multiple crops [68]. They tested both an end to end supervised counting algorithm (S-COUNT) and a weakly supervised counting algorithm (WS-COUNT). S-COUNT is based on the ResNet101 CNN with the final fully connected layer replaced with a 1×1 convolution with eight filters. To train S-COUNT, images are provided with a label representing the number of fruit present in the image. In contrast, WS-COUNT requires only a binary indication of the presence of at least one instance of a fruit for each image. WS-COUNT operates at multiple levels, on the full image, on the image divided into quadrants and on the image divided into 16 parts. Constraints are enforced to ensure the count is consistent between levels. The advantage of both S-COUNT and WS-COUNT over systems such as Faster R-CNN is the minimal labelling effort required, particularly in the case of WS-COUNT. They compared their methods to that of Bargoti and Underwood (Faster R-CNN) [22], Rahnemoonfar and Sheppard (DeepCount, modified Inception-ResNet) [79] and Zhou et al. (PRM, a weakly supervised FCN for instance segmentation) [87]. The Faster R-CNN approach was both the slowest to train and perform inference in most cases. DeepCount and PRM were both very fast to train, taking less than an

hour, compared to 12 hours for Faster R-CNN, with DeepCount also being the fastest to perform inference. Faster R-CNN was more accurate on all datasets tested, while DeepCount and PRM were the least accurate. Both S-COUNT and WS-COUNT demonstrated impressive performance given they require far less labelling effort than the Faster R-CNN or DeepCount approaches.

Williams et al. used Faster R-CNN to detect kiwifruit flowers for the purpose of pollination [76, 77]. The network was trained on 1015 images at a resolution of 1024×600 pixels. The network achieved a precision of 0.91, recall of 0.80 and F_1-score of 0.85 across three test datasets containing a total of 211 images. The detection system was part of an automated targeted wet pollen application system that managed a 61.8% hit rate of flowers when travelling at 1 km/h.

In addition to kiwifruit pollination, Williams et al. also presented a robotic kiwifruit harvester that uses a fully convolutional neural network called FCN-8S [15]. The network was trained on 48 images of 200×200 pixel resolution to perform semantic segmentation of images. The imaging system has a resolution of 1920×1200 pixels, so each image was spit into 12 sections which were each passed through the network individually and the results recombined. Kiwifruit calyxes, wires and branches were detected in images. In an evaluation across a series of images, the system was able to detect 79% of visible fruit. In a continuation of this work, Faster R-CNN was implemented to replace RCN-8S [16]. The network was trained to detect both the fruit and calyxes. The harvesting system used the calyxes as the target as the smaller size of the calyx provides more positional accuracy with the stereo localisation system used. The detection network achieved a precision of 0.95, recall of 0.85 and F_1-score of 0.90 on a series of evaluation images. The network was able to run at 5 Hz on a GPU. The level of performance of Faster R-CNN was shown to be much higher than FCN-8S in both detection accuracy and processing time.

In a 2018 review of deep learning in agriculture, Kamilaris and Prenafeta-Baldú looked at 40 publications [88]. They found that deep learning tech-

niques offered superior performance, while requiring less effort to implement than conventional image processing techniques in most cases. Faster R-CNN and DetectNet CNN were highlighted by the authors as promising networks for yield estimation. However, the authors point out that deep learning methods often require large datasets for training to represent sufficient diversity. Obtaining these large datasets is an obstacle for many researchers.

Koirala et al. performed a review of deep learning methods for fruit detection and yield estimation in 2019 [89]. The authors note that with the high accuracies reported for fruit detection using deep learning methods, research attention should shift to approaches to estimate the total fruit per tree and per orchard.

2.1.4 Occlusion Compensation Methods

The most commonly cited challenge faced by researchers in yield estimation is occlusion. Fruit can be partially or fully occluded by leaves, branches and even other fruit. Fully occluded fruit are not visible to the imaging system and, hence, cannot be individually counted. Partial occlusions give the fruit irregular shapes and sometimes split a fruit into multiple sections. A study conducted by Williams et al. showed that 3.3% of kiwifruit were not visible in images taken from below the canopy [15]. To combat occlusion, researchers have tried many approaches; correction factors, taking multiple images of each plant, tracking fruit through a series of images and more.

The simplest way to account for occlusion is with a correction factor. These are usually obtained by regressing the fruit count produced by the algorithm with the ground truth fruit count. These static correction factors have been used by many researchers [18, 23, 29, 32, 44, 55, 56, 61, 73, 90]. Applying a static correction factor removes any under-counting bias and in most cases, significantly improves yield predictions. However, applying a static correction factor assumes that the occlusion rate is consistent across plants, rows, orchards and vineyards. This assumption does not seem to be true.

To reduce the effects of occlusion some researchers have taken multiple images of the same plant from different angles to see more fruit [50, 51, 59]. Linker took images of apple trees at three different heights from each side of the tree [59]. Multiple models were formed to predict yield from different combinations of images, such as all images from one side, only the lower and upper images and all images from both sides. The best results were achieved when using all of the images, very closely followed by using images from one side. However, the testing was performed on a relatively small dataset (23 trees). The downside of taking multiple images of the same plant is the same fruit being counted more than once.

2.1.4.1 Multi-frame Tracking

To avoid double counting of fruit seen in multiple images, a multi-frame fruit tracking system can be used. A multi-frame fruit tacking system identifies the same fruit in a series of images so it can be counted exactly once. Various implementations have been used by researchers [19, 20, 28, 36, 45, 60, 74, 76, 78].

Moonrinta et al. tracked pineapples between frames taken from a moving platform [45]. Pineapples were identified as the same pineapples in consecutive frames if the detected areas overlapped. Structure from motion was then applied across pairs of images to construct a point cloud of detected pineapples. The authors note that their tracking algorithm needs improvement to better handle merges and fragmentations. Also noted was the need to add GPS and/or SLAM to resolve scale ambiguity and provide a 3D visualisation.

Wang et al. used a stereo pair of cameras mounted to a moving platform with GPS to track apples [19, 20]. Each apple was seen in up to seven images from each side of the row. Fruit were globally localised using GPS and stereo triangulation. A distance threshold was used to determine if fruit seen in different image pairs are the same fruit. GPS drift and stereo triangulation bias were mentioned by the authors as issues, which were overcome by placing landmarks in the images that were located by hand.

Gongal et al. located apples with an over-the-row system with a colour and a time-of-flight camera [28]. The cameras were moved to different heights on both sides of the row to capture multiple viewpoints of each tree. Apples were detected in the colour image and localised using the time-of-flight image and the position of the cameras. A distance threshold was used to merge fruit seen from multiple viewpoints. The authors report a mean absolute percentage error in identifying duplicate apples of 21%, which they attribute to clusters and errors in 3D positioning.

Stein et al. used a single camera on a moving platform with a global positioning inertial navigation system (GPS/INS) to track mangoes [36]. Stereo analysis was conducted between consecutive frames using the GPS/INS to estimate camera movement. The Hungarian method [91, 92] was used for stereo matching. Fruit were not associated in 3D after triangulation, instead a fruit could only be identified as an existing fruit if it is seen in consecutive images. If a fruit is visible, before being occluded and becomes visible again through a series of images, it is counted as two fruit. The authors cite issues with the epipolar geometry not being accurate when the platform hit a bump, causing oscillations of the camera. They also mentioned issues with clustered fruit being incorrectly matched.

Liu et al. tested both a hand-held camera and a camera on a moving platform to track apples and oranges [60]. Camera movement was estimated using optical flow with a Kalman filter. Fruit were stereo matched between consecutive frames using the Hungarian method [91, 92]. The fruit were then localised using structure from motion. A fruit was only counted if it was seen in a series of images.

Halstead et al. and Herinrich et al. both used a similar approach to track capsicum and bunches of grapes respectively from moving platforms [74, 78]. Their algorithms simply check for overlap between bounding boxes in consecutive images. If the overlap is above a threshold, they are considered to be the same fruit.

Williams et al. tracked kiwifruit flowers using a pair of stereo cameras on a moving platform for the purpose of pollination [76]. Flowers were detected in each image pair, then stereo matched using the Hungarian method [91,92] and localised relative to the cameras via stereo triangulation. Movement of the cameras was tracked via odometry from the platform. For the next frame, the new position of the previously seen flowers was predicted using the movement of the camera. Newly localised flowers were then matched to previously seen flowers using the Hungarian method.

2.1.4.2 Multi-fruit Region Classification

One of the issues caused by occlusion is overlapping fruit. When one or more fruit are partially occluded by another fruit, detections systems can classify the entire region as just one fruit. To overcome this issue, Gong et al. used a modified 8-connectedness chain code (M8CCC) to identify the number of fruit in a fruit region [71]. Their M8CCC algorithm encodes the outline of an identified fruit region as a series of numbers. Each number describes the direction of a section of the outline. The number of fruit in the cluster is defined by the number of times a particular number appears in the generated code. The M8CCC algorithm performed well on clusters of fewer than five fruit, but failed in situations where a fruit region was intersected by a leaf or branch.

Chen et al. used a neural network to infer the number of fruit in a cluster [24]. The authors do not report the performance of their count neural network in isolation. When included as part of a full yield estimation for oranges and apples, performance was significantly improved compared to when it was not used.

Tran et al. developed a custom algorithm for identifying the number of dragon fruit in a cluster [42]. A binary image representing detected fruit pixels is modified using a distance transform. The distance transform colours each fruit pixel based on its distance to the nearest non fruit pixel. Adaptive

thresholds are adjusted iteratively to split each region into its constituent fruit. In very limited testing, the algorithm performed well.

2.1.4.3 Other Occlusion Mitigation Techniques

Some researches have employed different strategies to reduce the effects of occlusion. Nuske et al. took images of grape vines using a single camera from a moving platform [46, 47]. The platform had a stereo camera pair facing the ground which was used for visual odometry. Each detected bunch of grapes was projected back onto the fruit wall. The row was split into chunks 0.5 m long. For each chunk, if there were grapes detected in multiple images, only the image with the most detections was used. All other detections were discarded. This method makes the assumption that the image with the most detections is the image least affected by occlusion. It avoids having to do registration of grapes between multiple images, simplifying implementation. The authors noted that the true positive counts are linearly related to actual grape count, whereas false positive count is independent of actual grape count. They used this to form a model to relate their counts to actual harvest weight.

Cheng et al. used a neural network to estimate apple yield [21]. For each tree, a single image was taken. The image was segmented into fruit pixels, foliage and background. The inputs to the neural network were the fruit area, number of fruit, small fruit area and foliage area. The output of the neural network was the total estimated fruit weight. The system performed well with a mean absolute percentage error of 8.9% on a per tree basis.

To estimate the weight of bunches of grapes, Font et al. tested models based on both grape area in images, and estimated grape bunch volume [64]. The bunch volume method assumes that a bunch of grapes can be modelled by the solid formed by revolving the grape area in an image about the vertical axis. Both methods performed similarly in the small amount of testing the authors conducted.

2.1.5 Conclusions

Fruit yield estimation systems have evolved from simple detect and count approaches to complex systems based on CNNs with multi-frame fruit tracking. However, further improvements need to be made to the state of art to provide accurate information to growers on a commercial scale.

Colour cameras are inexpensive and offer high resolution and performance. The addition of other sensors such as lidar or infra-red cameras have shown small increases in yield estimation performance, however, they add significant cost and complexity to the systems. Modern CNNs such as Yolo, SSD and Faster R-CNN have demonstrated impressive performance across a variety of conditions and crops and are the clear choice for vision based fruit detection systems.

Occlusion is the biggest remaining problem faced for fruit yield estimation systems. Taking multiple images of each plant can reduce the effect of occlusion by providing multiple viewpoints, however, this introduces the double counting problem. Multi-frame fruit tracking solves the double counting problem, provided a reliable fruit localisation method is implemented. The stereo vision approach of Wang et al. and Williams et al. where a calibrated stereo pair of cameras is used is a robust approach [19, 20, 76].

2.2 Sparse Stereo Correspondence

The stereo correspondence problem is identifying which objects in one image of a stereo pair correspond to which objects in the other image. When only discrete points need to be corresponded between images, it is sparse stereo correspondence, which is the case for matching detected fruit.

Yuille introduced the stereo ordering constraint for stereo correspondence problem [93]. If object A is to the left of object B in one image of a stereo pair, object A will also be to the left of object B in the second image of the stereo pair. This observation forms the basis of the ordering constraint. The system

allows the correspondence of multiple objects lying on the same epipolar line to be correctly inferred. However, if there is not a 1-to-1 correspondence between images, the system can produce incorrect results. This will occur if there is occlusion in one of the two images causing some objects to be visible in only one of the images. The constraint will also not hold if object B is inside of the 'forbidden zone', an area constrained by object A and the focal point of each camera.

For use in an apple harvester, an area based matching algorithm was used by Si et al. [94]. Detection was performed on each image and binary thresholding applied to mark all apple pixels. Fruit were matched based on epipolar geometry and finding detected fruit with similar area in each image. An ordering constraint was applied to ensure that matched fruit appeared in the same order along the epipolar line in both images. The method achieved a 95% success rate with the authors noting failures mostly caused by apples overlapping in one image but not the other, causing differences in measured area.

To localise tomatoes and apples, a mean disparity approach was used by Plebe et al. and Xiang et al. [95, 96]. First, a matrix was constructed that contained the disparity of every possible matching combination. All values that fell outside disparity thresholds (decided by the distance they expected fruit to be from the cameras) were then removed. The mean disparity was calculated and for fruit with multiple potential matches, the match with the disparity furthest from mean was removed. This was repeated until only one or zero matches were left for each detected fruit. Neither authors quantified the performance of their systems.

Nielsen et al. used a trinocular camera configuration for localising peach blossoms [97]. Their matching algorithm can utilise all three cameras or just two, based on which combination gives the best quality match. To evaluate matches, a fixed size window around a detected blossom was extracted and a pixel-wise comparison conducted against detected blossoms in the other images. They performed the comparison at multiple window locations to find the

location of highest correspondence between the images. Their method is very computationally intensive, taking 60–80 seconds per image pair, however they were using very high resolution images (10 MP) and have a very high number of blossoms per image.

For use in localising litchi, a grey-scale correspondence matching algorithm was employed by Wang et al. [98]. All of the pixels belonging to a single litchi in one image would be swept along the epipolar line in the other image. The similarity of the grey value of the pixels in the two images was evaluated at each point along the epipolar line using a normalised cross-correlation. A match was declared at the point of maximum similarity. Their algorithm correctly matched 98% of unoccluded litchi and 94% of partially occluded litchi. The authors note the main reason for mismatching partially occluded litchi was the shape of the litchi appearing to be different in the two images.

For use in a kiwifruit harvester, a reduced search space template based algorithm was used by Scarfe [13]. A fixed size template was was taken from each detected kiwifruit calyx and converted to grey-scale. For each calyx, a search window was identified in the other image of the stereo pair based on camera geometry and expected object distance. The template was then swept through the search window and the sum of squared differences between the overlapping patch of image and the template calculated for each position. The position with the lowest sum was selected as the matching point, if it met a set threshold. The author cites computation time as 14–26 seconds per image pair. However this was run on a CPU from 2007 and programmed without regard for computation time.

Williams et al. applied the Hungarian method [91,92] for matching of both kiwifruit and kiwifruit flowers in a robotic harvester and pollinator [15, 16, 76, 77]. The success rate is only reported for the harvester, which achieved a very high 99.7% matching rate.

2.2.1 Conclusions

The methods outlined above all produce acceptable matching accuracy when used on their respective datasets. In particular, the Hungarian method used by Williams et al. achieved impressive results [15, 16, 76, 77]. However, those that reported computation times showed that their algorithms are very computationally intensive. This high computation time is due to many comparisons being run on subsets of the images. With the relatively 2D nature of kiwifruit orchards, a simpler algorithm could be applied that forgoes comparing features of the image. Such an algorithm could offer both high accuracy and very low computation times.

2.3 Point Cloud Registration

One of the problems of a multi-frame fruit tracking system is finding the correspondence between fruit seen in different images. A method to solve this is to use a point cloud registration algorithm. In the case of point clouds representing fruit, there is a high likelihood that two point clouds being registered will not have a 1-to-1 correspondence. This is because point clouds will often contain fruit that are not present in the other.

Besl and McKay presented the iterative closest point (ICP) algorithm [99]. The ICP algorithm is a cost function minimisation algorithm. The cost function is the sum of the squared distances between each point in the cloud being registered and the closest point in the cloud being registered to. With each iteration, a new transform is applied to the cloud being registered, which shifts the points, optimising their position relative to the second cloud. The algorithm will find the correct solution if the initial guess is relatively close. However, if the initial guess is not close to the optimal solution, finding a local minimum rather than the global minimum is likely.

To register two or more partially overlapping point clouds, Larkins presented a method based on analysis of estimated surface normals [100]. First,

the surface normal is estimated for each point in each point cloud. Then, for each of the point clouds, a spherical-harmonic transform is applied to the set of all normals. The spherical harmonics for each point cloud are then cross correlated to identify the potential relative rotations of the point clouds. The translation between the two point clouds is then determined using a 3D Fourier phase correlation. The quality of alignment is then assessed and if adequate, an iterative closest point algorithm is applied to improve the quality of registration. This method was shown to be effective at registering point clouds obtained from 3D scans (or depth cameras) to reconstruct 3D objects. However, performance on point clouds not representing surfaces was not investigated.

For registration of a dense point cloud with a sparse point cloud, Agamennoni et al. presented a modified version of the iterative closest point algorithm (ICP) [101]. Their modification changes how data is associated between the two point clouds with a system they call probabilistic data association. Each point in the first point cloud is associated with multiple points in the second point cloud, rather than just one as in the standard ICP algorithm. Each of these associations is weighted to form a probability distribution. They claim their system is more robust to both noise and outliers when compared to standard ICP. When applied to registering points clouds from a 360° lidar to those from a Kinect sensor (Microsoft, Redmond, Washington, USA), the point clouds were registered correctly in all test cases.

2.3.1 Conclusions

Many 3D point cloud registration methods are designed for application on dense point clouds. The point clouds representing fruit positions are sparse point clouds that have between 0 and 100 points per cloud. Dense point clouds obtained from depth cameras, lidar units and dense stereo systems contain thousands or even millions of points. These dense point clouds represent the surfaces of objects as opposed to the locations of discrete objects in a scene as they do with fruit locations. Thus, many of the existing point cloud

registration algorithms are not suitable for this application. However, when registering point clouds representing fruit, the point cloud is not the sole source of information. The appearance of the fruit in the images could also be used to aid the registration algorithm, improving accuracy.

2.4 Simultaneous Localisation and Mapping

Simultaneous localisation and mapping (SLAM) systems can both create a map of an area, while localising an agent within that map. There are many implementations of simultaneous localisation and mapping systems, each with different specializations and capabilities. Various options are investigated to find the most suitable for localisation of a vehicle in an orchard.

HectorSLAM is a SLAM implementation designed to perform real-time estimates using lidar and, optionally, IMU data [102]. The Gaussian-Newton algorithm is used to optimise a rigid transform for scan matching. To avoid local minima, a multi resolution map approach is used.

GMapping is a SLAM implementation that uses a Rao-Blackwellized particle filter with an adaptive resampling technique [103]. GMapping is written to take long-range lidar data and vehicle odometry to make its estimations. Santos et al. performed an independent evaluation of multiple SLAM implementations and found GMappings accuracy and computational load to be similar to that of HectorSLAM [104].

Cartographer is an open-source SLAM implementation developed by Google (Mountain View, California, USA) that can be run in both a 2D and 3D mode [105]. Estimations are based on lidar data with odometry and IMU data being optional (IMU is required for 3D SLAM). Cartographer is split into two systems. The first is a local SLAM system which produces small submaps. These submaps are produced over a relatively short time period to avoid large errors. The submaps are fed into the second system, where they are matched and combined to form the global map and loop closure is performed.

All of the SLAM systems investigated are available with ROS integration, making for easy software integration.

Chapter 3

Data Collection

The aim of the data capture system is to provide high quality data suitable for use in estimating the position of each visible fruit in an orchard. It must be robust and easy to use as the quantity of data required is very large (tens of hours of data collection). The main form of data required is stereo images of the kiwifruit canopy for detecting and localising kiwifruit relative to the cameras. Also required are point clouds from a lidar unit and readings from an inertial measurement unit (IMU) for estimating the position of the data capture system within the orchard.

3.1 Data Capture System Hardware

The hardware of the data capture system consists of a vehicle, the various sensors, laptop and mounting hardware.

3.1.1 Cameras

The canopy of pergola kiwifruit orchards sits between 1700 mm and 2100 mm from the ground. There are fruit outside of these limits: lower fruit are prone to damage from machinery and are thus often graded as rejects by the packhouse; higher fruit are often not harvested as they are both difficult to see and reach. Therefore, the cameras on the data capture system need to be

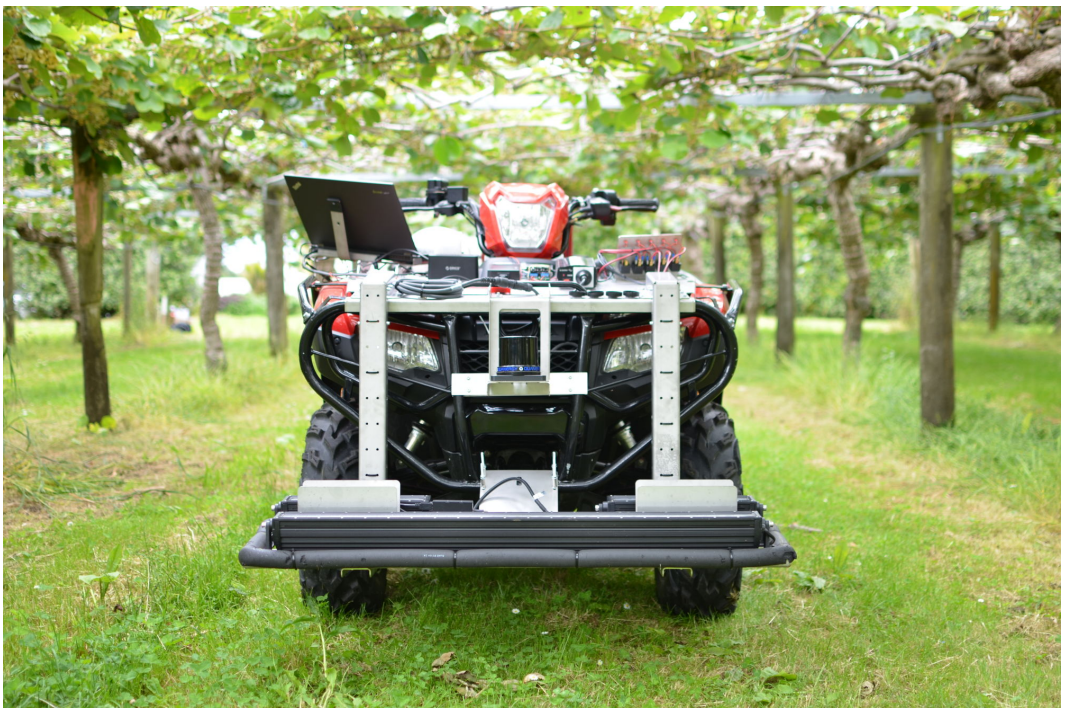


Figure 3.1: The data capture system in an orchard.

capable of imaging fruit 1700-2100 mm from the ground.

The Ace acA1920-40uc (Basler, Ahrensburg, Germany) is a 2.3 MP colour camera with a USB3.0 interface. It is used because it has the following features/attributes:

Global shutter Due to camera movement while capturing an image, a rolling shutter would introduce distortion to the images as different parts of the image are captured at different points in time. A global shutter negates this effect by exposing all pixels of the sensor simultaneously.

Hardware trigger Performing stereo analysis requires that images from both cameras are taken simultaneously. Hardware triggering is used to achieve accurate synchronisation.

GPIO The two general purpose input output pins can be used both as an input or output for the hardware trigger.

Dynamic range The high dynamic range (73.2 dB) makes this model suitable for outdoor conditions.

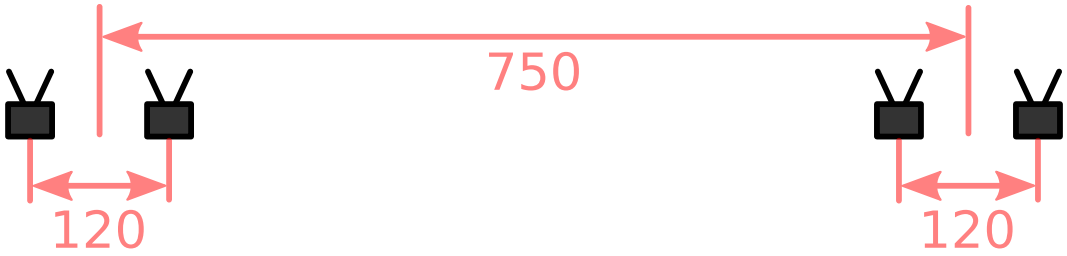


Figure 3.2: Scale diagram showing camera spacing from the front. Stereo baseline is 120 mm, separation between the two stereo pairs is 750 mm.

Resolution The 2.3 MP (1920 x 1200) resolution has been shown to be adequate for detecting fruit [15].

The LM12HC (Kowa, Aichi, Japan), 12.5 mm lenses are used as they provide the desired field of view (FOV) with the selected cameras (see Figure 3.3 below). They have an adjustable aperture from F1.4 to F16 and minimum focal distance of 300 mm. A smaller aperture (higher F number) gives a larger depth of focus while decreasing the amount of light captured by the sensor. An adjustable aperture allows using the setting that gives the desired depth of field, while allowing maximum light to the sensor. More light means shorter exposure times, giving less blur as the images will be taken from a moving vehicle. The FOV of the lens gives a resolution of approximately 1 mm/px at the edge of the image at the highest working distance.

Four cameras are arranged as two stereo pairs to provide sufficient imaging width. The base line for each of the stereo pairs is 120 mm (Figure 3.2). At 120 mm base line, the depth error is less than 10 mm per pixel of disparity error over the working range. The separation between camera pairs is 750 mm (Figure 3.2) giving overlap between the FOVs (Figure 3.3) of the two camera pairs. The overall imaging width is 1840 mm at the lowest working height (1700 mm from the ground).

The mounting position for the cameras should be as close to the ground as possible. Being mounted further from the canopy reduces the change in viewing angle from the centre of the image to the edges (given the same viewing area), aiding kiwifruit detection and localisation accuracy. However, either longer

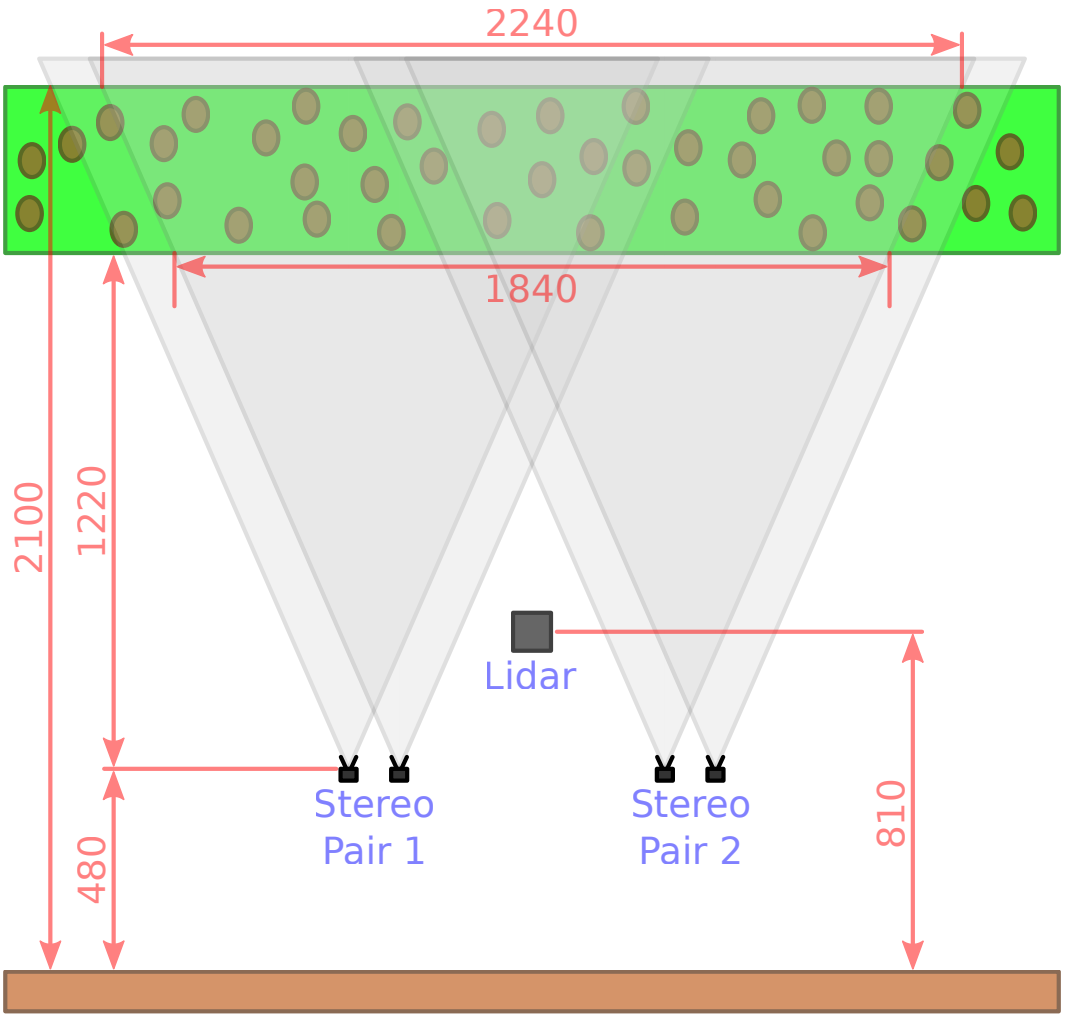


Figure 3.3: Scale diagram showing camera and lidar positioning from the front. Gray triangles represent the field of view of each camera. The brown rectangle represents the ground, the green rectangle represents the canopy. Note there are four cameras arranged in two stereo pairs. Dimensions in mm.

exposure or more light is required for a similarly bright, well exposed image, when compared with a higher mounting location. The cameras must also be high enough to allow adequate ground clearance. A camera height of 480 mm from the ground is used as a compromise between these factors (Figure 3.3).

Imaging every area of the canopy from multiple perspectives can reveal occluded fruit that may not be visible from a single perspective. Therefore, overlap is required between images. Taking an image every 200 mm of travel will mean any area of canopy is seen in 3–4 consecutive images (disregarding additional overlap between camera pairs or passes). When travelling at the target velocity of 5 km/h (1.4 m/s), imaging at 7 frames per second gives an

image every 200 mm.

One of the four cameras is triggered via software at a rate of 7 frames per second. One of the GPIO pins on the software triggered camera is configured to be driven high when the camera is exposing an image. That pin is connected to a GPIO pin on each of the other three cameras, which are configured as a trigger inputs. Triggering all the cameras simultaneously ensures that all of the cameras take an image at the same time, which is vital for processing stereo images.

The focal distance of the lenses is set so that the working range is in focus. The aperture of the lenses is set to 1.4 to give adequate depth of focus. Each of the four cameras are configured identically (apart from one being software triggered) to ensure consistent images. Exposure is set to 3 ms to give well exposed images in orchards with the LED lighting. Automatic white balance and gain adjustment is turned off to avoid inconsistencies.

3.1.2 Lighting

As kiwifruit orchards are outdoors, lighting conditions can vary significantly. External lighting can reduce the effect of natural lighting variations, aiding detection. Three LED light bars are used for the data capture system, one 40" 240 W and two 10" 60 W. The light bars have a colour temperature of 6000 K, similar to that of daylight. The light bars are mounted alongside the cameras.

3.1.3 Lidar

The M8-1 (Quanergy, Sunnyvale, California, USA) is a 3D, mechanical lidar with 8 layers. It is used because of budgetary constraints, not its features or design. Its FOV is 360° horizontally and 20° vertically, split asymmetrically with the top layer 3° above horizontal and the bottom -17° below horizontal (Figure 3.4). The viewing area makes the Quanergy lidar suitable for mounting on top of the vehicle. However, in a kiwifruit orchard environment, mounting sensors above the vehicle is not practical due to the pergola structure. A symmetrical

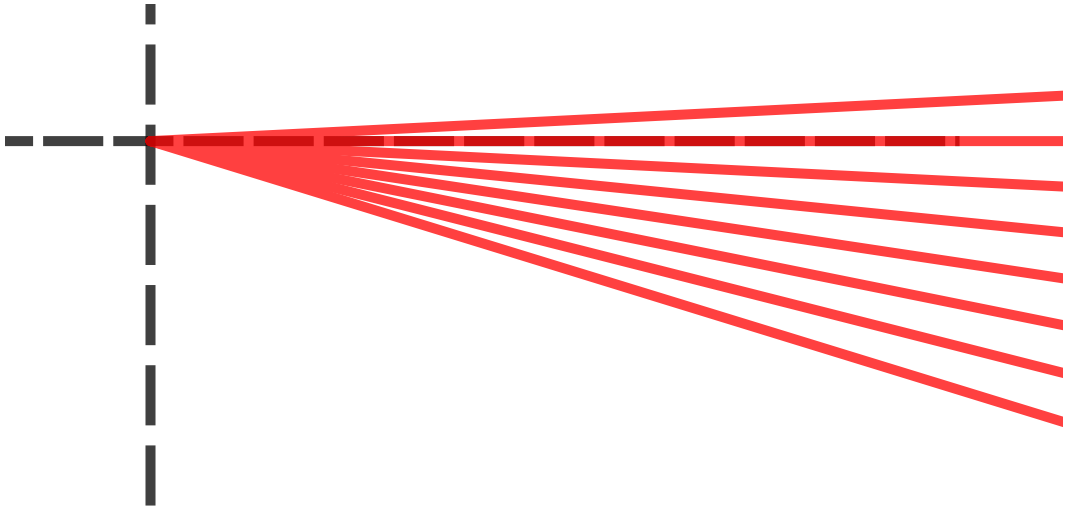


Figure 3.4: The arrangement of lasers in the Quanergy M8-1 lidar shown from a side view. The lasers rotate about the vertical axis. The red lines represent the laser beams. The top laser is oriented 3° above horizontal, the bottom is 17° below horizontal.

vertical FOV, would be more suitable in a kiwifruit orchard to achieve maximum range of vision, despite uneven ground and other obstructions. Other lidar units such as the VLP16 (Velodyne LiDAR, San Jose, California, USA) or OS-1 (Ouster, San Francisco, California, USA) feature symmetrical vertical FOVs and would be better suited to this application. A compromise is to tilt the Quanergy lidar backwards 7°. The tilt gives an optimal FOV in the forward direction at the expense of the rearward direction (Figure 3.5). However, when mounted on the front of the ATV, the rearward direction is blocked by the ATV and frame of the data capture system (Figure 3.6). The FOV is unaffected on the sides (Figure 3.7).

The lidar is set to scan at its maximum of 20 Hz. Using a lidar scan rate of 20 Hz, as opposed to 10 Hz gives double the temporal resolution at the expense of half the angular resolution (returned points per second is constant between scan rates). The lidar returns data in packets containing 360° worth of data. Because the lidar spins and captures points sequentially, each of the points is measured at a different point in time. If the lidar is moving in space, the origin (zero point) of the lidar is also moving. The movement causes the origin to be in a different location for each of the measured points, distorting the resulting

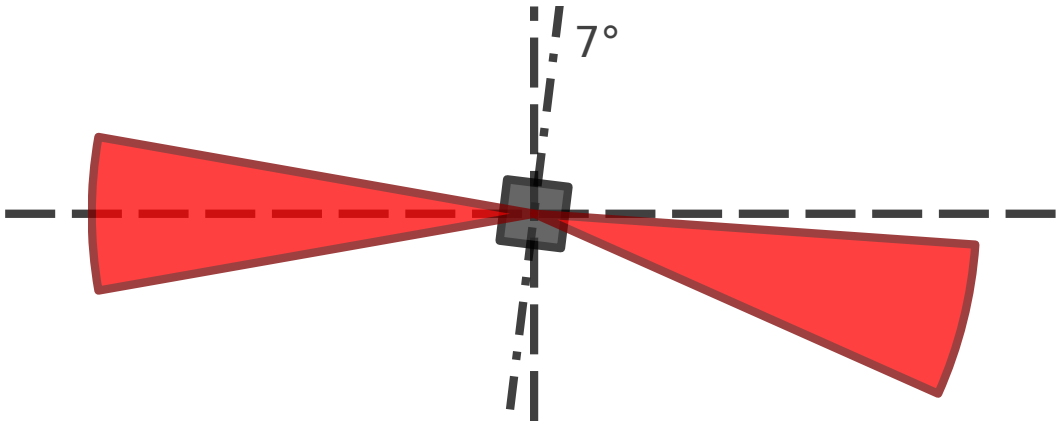


Figure 3.5: Lidar field of view from the side. The direction of travel is to the left of the image. The lidar is tilted backwards 7°. This tilt gives it a symmetrical vertical field of view about the horizontal axis in the forwards direction.

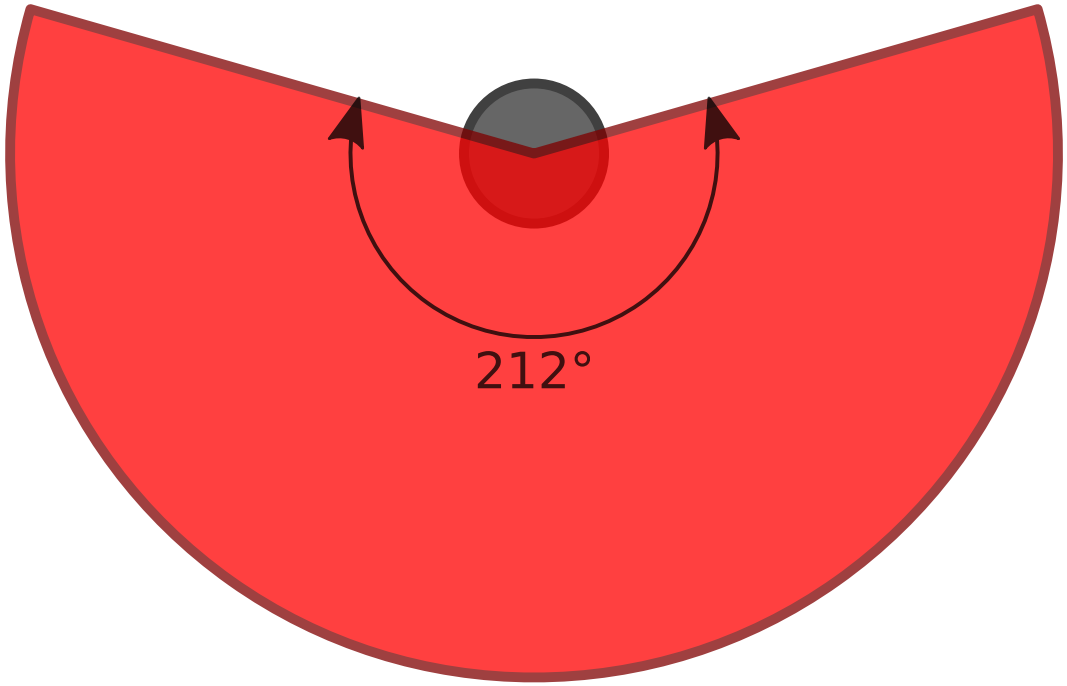


Figure 3.6: Lidar field of view from the top, looking down the axis of the lidar. The direction of travel of the data capture is down in the image. The lidar is blocked in the rearward direction by the frame of the data capture system and the ATV. Total horizontal field of view is 212°.

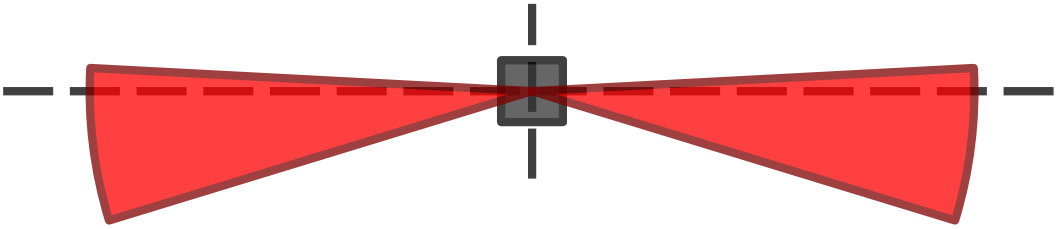


Figure 3.7: Lidar field of view from the front. The field of view to the sides is biased downwards due to the asymmetrical design of the lidar. The field of view extends 3° above horizontal and 17° below.

point cloud. Using a higher scan rate minimises the effect of movement.

3.1.4 Other Sensors

The Life Performance Research LPMS USBAL2 IMU contains a 3-axis accelerometer, 3-axis gyroscope and 3-axis magnetometer. It is used because of its rugged aluminium housing, USB connectivity and availability of a Robot Operating System (ROS) driver. The IMU is set to return measurements at 100 Hz. The reported measurements are; 3-axis linear acceleration, 3-axis angular velocity, 3-axis magnetic field strength and an orientation quaternion.

Fitting a odometry measurement system to the ATV would provide additional feedback for the ATV localisation system (SLAM, see Section 2.4), which could improve localisation accuracy and robustness. However, the additional cost and complexity of adding a reliable system, that doesn't require permanent modification to the ATV is deemed too high.

3.1.5 Mechanical Design

The data capture system is designed around a TRX500 ATV (Honda, Tokyo, Japan). It is used because of budgetary constraints, not because of its feature set.

The frame of the data capture system is designed to be easily removable from, and require no permanent modification to, the ATV. The frame is constructed from laser cut sheet stainless steel and is mounted to the front carrier

of the ATV. It is designed to be assembled using bolts so it can be fully disassembled or modified without any cutting or welding.

The frame consists of five main sections, as shown in Figure 3.8:

Front carrier mount (yellow) The main mounting point of the frame onto the ATV. It is a panel with holes that align with the mounting holes in the front carrier of the ATV. Power supplies and other required hardware are mounted to it.

Struts (red) Two upright struts connect the front carrier mount to the camera tray.

Camera tray (green) Holds the cameras, LED light bars and USB hub. It also has protective shields for the cameras.

Lidar and IMU mount (purple) A bracket to hold the lidar and IMU out the front of the ATV. It is angled back at 7° to optimise the FOV of the lidar.

Laptop mount (blue) A bracket to hold the laptop in a convenient location within reach of the operator.

3.1.6 Assembled Data Capture System

The assembled data capture system can be seen in Figures 3.9 3.10 3.11.

The following is a full list of the hardware on the data capture system:

Cameras 4x Ace acA1920-40uc, 2.3 MP, colour cameras (Basler, Ahrensburg, Germany).

Lenses 4x LM12HC, 12.5 mm, F1.4, C-mount lens (Kowa, Aichi, Japan).

Lidar M8-1, 8 line, 360° FOV lidar (Quanergy, Sunnyvale, California, USA).

IMU LPMS-USBAL2, 9-axis accelerometer, gyroscope and magnetometer (Life Performance Research, Tokyo, Japan).

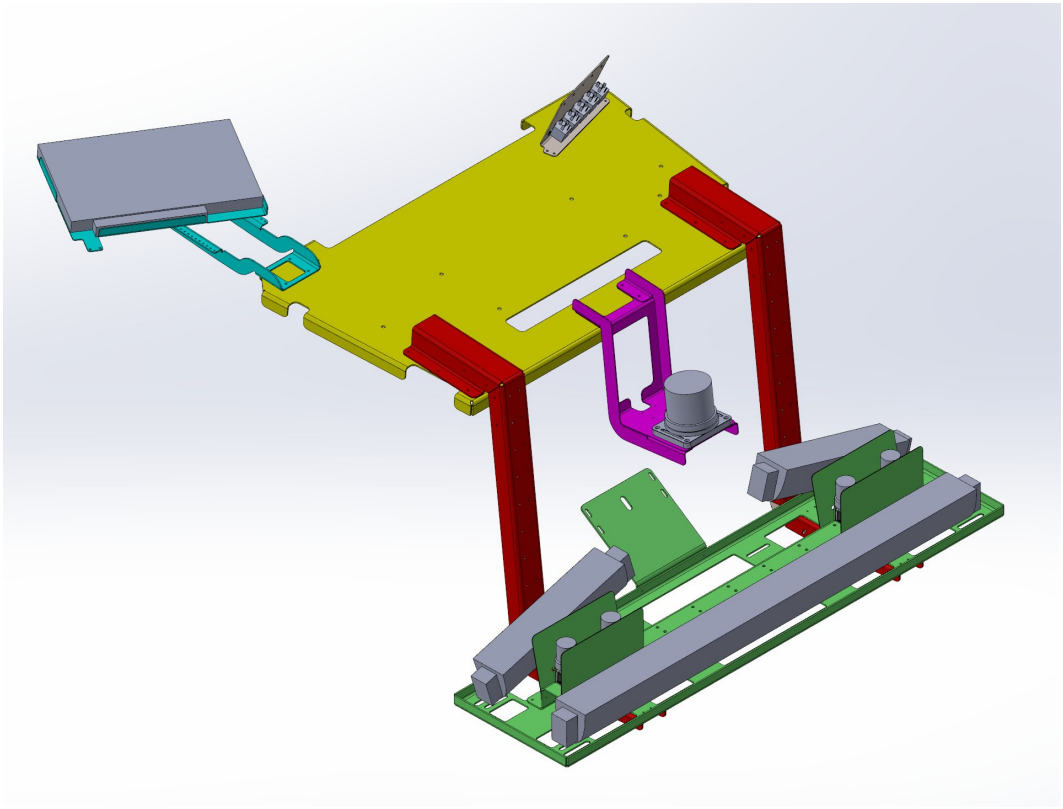


Figure 3.8: A computer aided design (CAD) rendering of the sections of the frame. Yellow is the front carrier mount. Red are the struts. Green is the camera tray and camera shields. Purple is the lidar and IMU mount. Blue is the laptop mount. Gray is the laptop, lights, cameras, lidar and light switches.

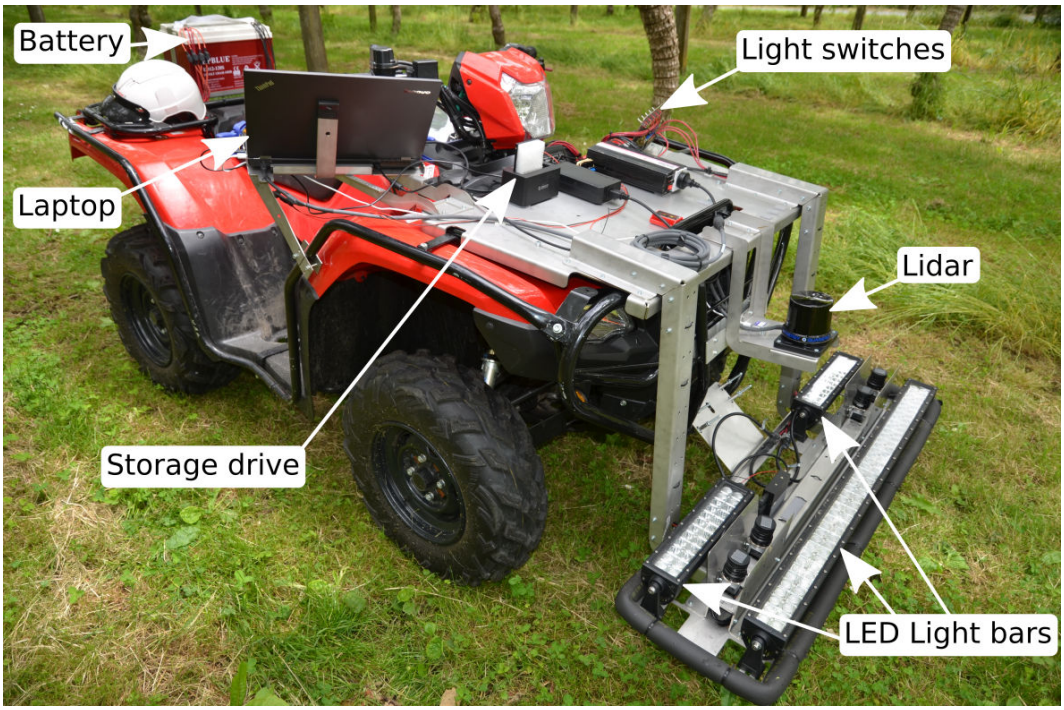


Figure 3.9: Overview of the components of the data capture system.

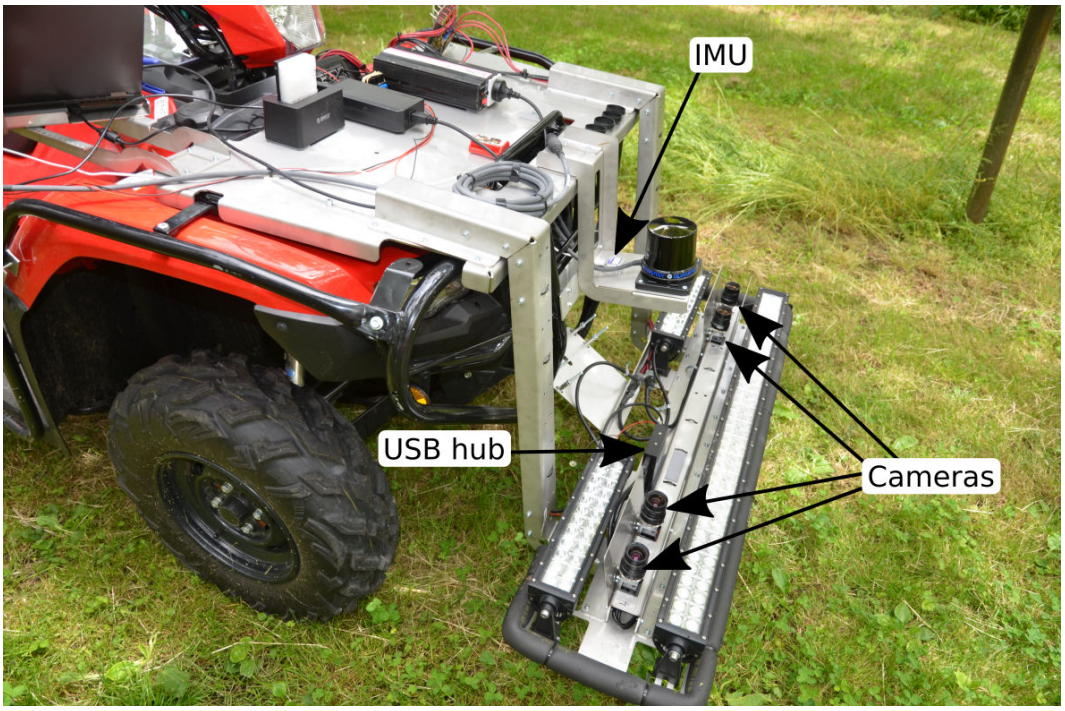


Figure 3.10: Overview of sensors on data capture system.

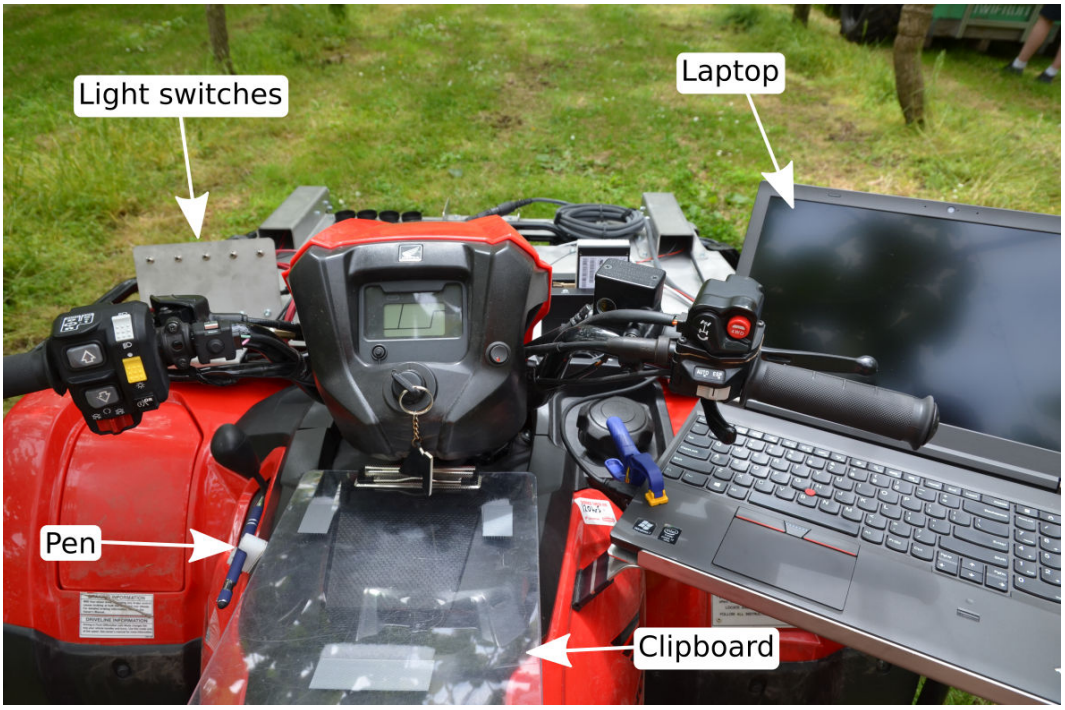


Figure 3.11: View from the operators seat of the data capture system.

ATV 2015 TRX500, 500 cc, IRS (Honda, Tokyo, Japan).

Laptop Thinkpad W541 Laptop (Lenovo, Beijing, China).

SSD SSD Plus 240 GB, 2.5" SSD (SanDisk, Milpitas, California, USA).

External drive bay USB 3.0 SATA docking station (Orico, Shenzen, China).

Light bars 1x 40" 240 W, Pro Line LED light bar, 2x 10" 60 W, Pro Line LED light bar (LEDWAREHOUSE, Nevada, USA).

USB hub 4 Port, powered, USB3.0 hub (Basler, Ahrensburg, Germany).

Battery EPBLUE ED12-120S 12 V, 120 Ah battery (East Power Battery, Shenzhen, China).

Inverter Generic 300 W, 12 V DC to 230 V AC inverter (for laptop charger).

DC-DC Converter 9–36 V to 24 V DC, 20 W, power converter (CUI inc., Oregon, USA) (for lidar).

Other Electronics Various cables, connectors, switches and fuses.

3.2 Data Capture System Software

The rosbag feature of ROS [106] is used to record the collected sensor data. A rosbag records ROS messages, which each consist of a single sensor reading, such as an image or a lidar scan, along with a timestamp. The resulting output bag file can later be replayed using the `playbag` function of ROS or the messages can be read directly via the rosbag C++ or Python APIs.

A Python script adds a label to the data at the start and end of every row. The label contains both the row number and the pass number, for example, `begin_row_4_pass_1` and is recorded as part of the rosbag. The script requires the rider to press the space bar when entering or leaving a row and it automatically increments the row number.

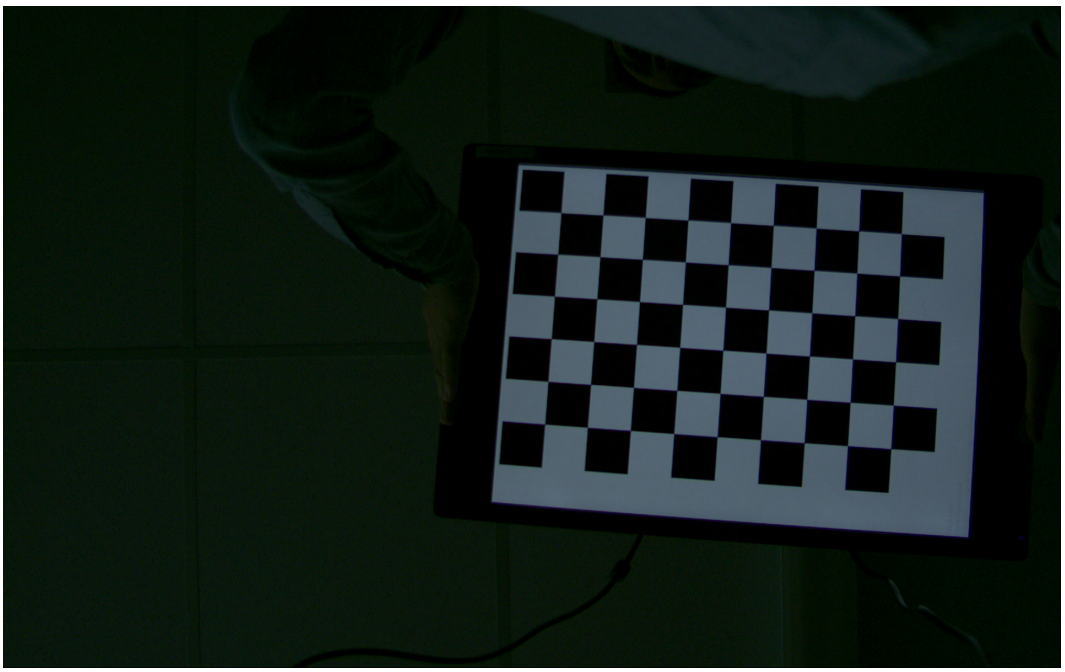


Figure 3.12: An example of an image used for camera calibration.

The laptop screen displays a visualisation of the data being collected in real time. RVis (a ROS data visualisation tool) is used to show a representation of the data capture system with the point cloud produced by the lidar displayed around it. Images from each of the four cameras are shown sequentially so any debris or water on the lenses or any other issues can be identified and fixed. The current row and pass information is also displayed so any errors are noticed.

3.3 Camera Calibration

The cameras are calibrated using the OpenCV checker-board pattern [107]. Images are taken of the checker-board in a range of positions and orientations throughout the FOV of the cameras (Figure 3.12). The OpenCV functions `calibrateCamera` and `stereoCalibrate` are used to produce the relevant matrices describing each camera and the geometry of the two camera pairs.

3.4 Data Collection Method

Orchards mature at different times, based on geographical, weather and other factors. Ideally, all orchards would be visited at the same point in their maturity cycle to ensure consistency. However this is not possible due to issues such as weather, transport logistics and orchard access restrictions. All practical efforts are made to minimise the range of times between bud break/fruit set and data capture.

Before starting each block, the orchard name and block number is entered into the data capture software. The system is aligned with the start of the first row. The data capture software is started via a `roslaunch` command. The lidar is given time to start up (approximately 10 seconds) and the data checked via the on screen visualisation. The space bar on the laptop is pressed to insert a row and pass label into the data. The ATV is ridden down the first row at 5 km/h (1.4 m/s) as measured by the speedometer on the ATV (which reports in 1 km/h increments).

The turning radius of the ATV is too large to comfortably turn from one row directly into the next without doing a three point turn. Therefore, the system is taken down every second row, before returning to cover the skipped rows (Figure 3.13).

The imaging width of the data capture system is approximately 2 m (dependent on canopy height), which is less than the width of each row. Each row is navigated three times, each referred to as a 'pass' from henceforth. One pass is taken down each side of the row and one down the middle (Figure 3.13) to give full coverage in rows up to 5.4 m wide.

A clipboard with an orchard map, paper and a pen is carried on the ATV to record notes. Noteworthy things include:

- Any mistakes in data labels caused by human error;
- Weather conditions
- Stoppages in data capture for any reason;

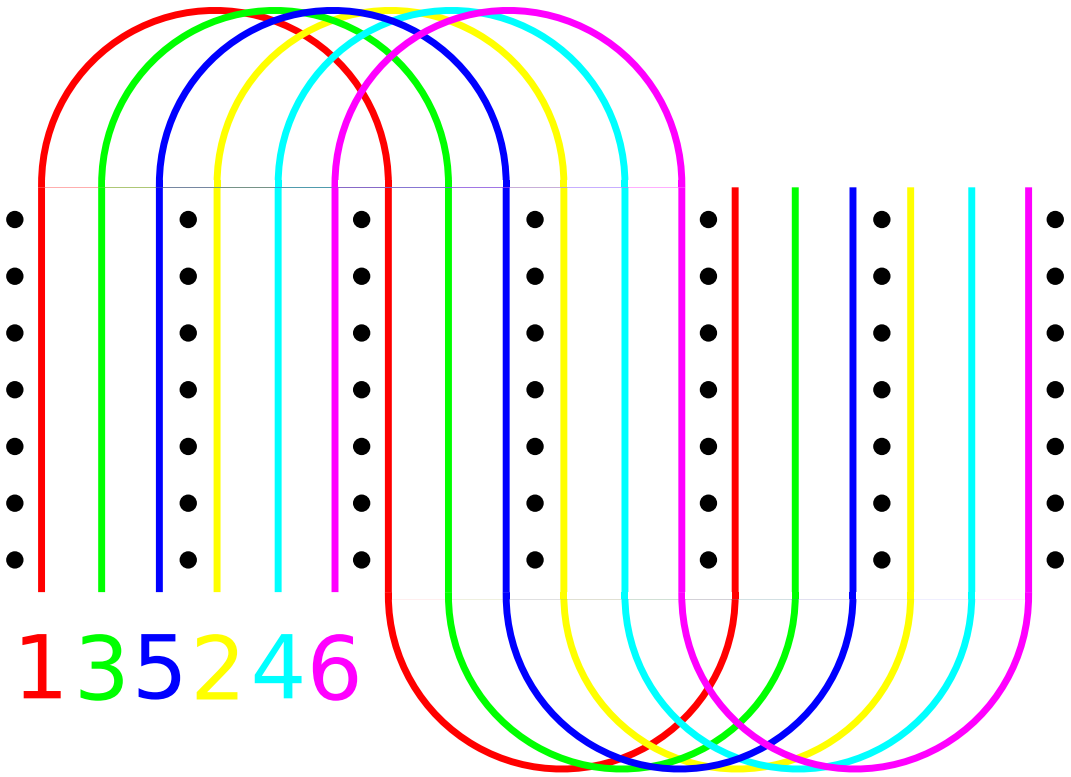


Figure 3.13: Order of passes. Black dots represent the posts and trunks bordering each row. Numbers represent the order of passes and the starting location for each pass.

  - Issue with the ATV
  - Curious orchard managers/owners/workers
  - Cleaning water/debris off sensors
- Anything out of the ordinary;
  - Particularly long grass
  - Strange or collapsed orchard structure

3.5 Collected Data

Data is collected twice in the growing season. Once at bud stage and once at fruit stage. Different orchards were covered in the two data capture runs.

Bud data was collected between October 13th and November 7th, 2017 (Figure 3.14). Seven each of green and gold maturity areas were included. A growing area of 20.4 Ha was covered across 14 maturity areas (Table 3.1). All orchards are managed by GroPlus Ltd and use the pergola growing system. Each orchard was visited between 34 and 43 days after bud break.

Fruit data was collected between January 24th and February 6th, 2018 (Figure 3.15). Only green orchards were included (see Section 4.2.2). A growing area of 33.9 Ha was covered, spread across 16 maturity areas (Table 3.1). All orchards are managed by GroPlus Ltd and use the pergola growing system. Each orchard was visited between 64 and 69 days after fruit set.

The fruit data is split into training, validation and testing datasets (Table 3.2). Ten maturity areas are randomly placed in the training dataset, with three each in the validation and testing datasets. Only data from the training dataset is used for development of all systems and algorithms. The validation dataset is used only for setting of final yield prediction model parameters. The testing dataset is used only to test the accuracy of the full yield estimation system once all other research and development is complete.

Figure 3.14: An example of an image from the bud dataset. The small round objects are the buds.

Figure 3.15: An example of an image from the fruit dataset.

	Buds	**Fruit**
Maturity Areas	14	16
Orchard Area	20.4 ha	33.9 ha
Orchard Blocks	34	47
Hours of Captured Data	33	52
Coverage Rate	0.62 ha/h	0.65 ha/h
Images	3.3 million	5.1 million
Lidar Scans	2.4 million	3.7 million
Total Data	2.5 TB	3.9 TB

Table 3.1: Statistics of captured data.

Maturity Area	**Dataset**	**Capture Date (days after bud break)**	**Size**
G_A	Validation	2018-01-24 (68)	3.29 Ha
G_B	Training	2018-01-24/25 (68/69)	3.67 Ha
N_A	Training	2018-01-26 (64)	2.07 Ha
R_A	Testing	2018-01-27 (64)	2.88 Ha
S_A	Training	2018-01-29 (65)	1.70 Ha
M_A	Validation	2018-01-29 (64)	1.23 Ha
H_A	Testing	2018-01-30 (64)	1.69 Ha
A_A	Training	2018-01-31 (64)	1.57 Ha
P_A	Training	2018-02-04 (67)	0.41 Ha
P_B	Validation	2018-02-02 (65)	2.52 Ha
P_C	Training	2018-02-04 (67)	2.44 Ha
K_A	Training	2018-02-06/07 (65/66)	1.74 Ha
K_B	Training	2018-02-08 (67)	2.84 Ha
K_C	Testing	2018-02-06/07 (65/66)	2.63 Ha
K_D	Training	2018-02-06 (65)	1.11 Ha
K_E	Training	2018-02-06 (65)	2.12 Ha

Table 3.2: Information on captured fruit data. Note, orchard names are obfuscated for privacy reasons.

3.5.1 Data Capture System Issues

Using the data capture system in the field highlights a number of issues with its design.

The camera tray is not visible from the riding position making it difficult to judge its position relative to posts, trunks or deadmen (knee height posts in the ground at the ends of rows). In 85 hours of data capture, there was only one significant crash, but it caused minimal damage.

The screen of the laptop extends outside the normal width of the ATV leaving it venerable to damage from trunks and low hanging canes. The screen sustained damage on multiple occasions, therefore, a protective plate was added.

The seat height of the ATV is relatively high. The seat itself is modified to lower the height of the rider. However, in orchards with low canopies or with low hanging canes or fruit, the riders head is in danger. Therefore, a helmet is required.

The ATV does not have cruise control. Maintaining a steady velocity is the job of the operator. The result is some fluctuations in velocity and after many hours of data capture, hand fatigue.

3.6 Ground Truth Data

To obtain ground truth data, all visited orchards are harvested, graded and packed by a commercial packhouse using industry standard techniques and processes. No special treatment is given to the fruit that are part of this study as the orchard workers and packhouse operators are not aware of the study. The packhouse produces a packout report (as they do for all orchards), from which the ground truth data is taken. The packout report provides a summary of the fruit that reach the packhouse. One packout report is produced per maturity area (some orchards consist of multiple maturity areas). For class one fruit (export quality fruit), the number of trays of each count size is reported. For class two (local market fruit), and waste fruit, the total weight of fruit is

reported. An example packout report is shown in Appendix A.

Using data from the packhouse allows yield estimation trials to be conducted on a scale that would be otherwise impractical. However, there are steps in the growing process between yield estimation and packing. For example, some orchard managers may thin fruit (remove unwanted fruit) during this time. Also, orchard workers are instructed to throw away any damaged fruit while harvesting, meaning they will not be counted by the packhouse. Additionally, a proportion of the fruit is left on the plants after harvesting as it is not visible to the orchard workers due to occlusion. Any variation between orchards introduced by these steps cannot be accounted for. Therefore, the overall accuracy achievable by the yield estimation system is limited by this variation.

To compare the accuracy of the final fruit yield estimation system to the current manual method, the manual counts are required. An attempt was made to obtain the manual counts performed by professional yield estimators after the growing season had finished. However, it was discovered that this data was not recorded by orchard managers. This oversight makes useful comparison of the two methods impossible. Attempts were also made to obtain data on the economic model of the current manual method of counting. Upon contacting multiple growers and orchard managers, it was discovered that there is little consensuses on prices, time taken and proportion of fruit counted. This uncertainty combined with the prototype nature of the presented automated system make economic comparison unfeasible.

Figure 3.16: Data collection going well.

Chapter 4

ATV Localisation

Knowing the position and orientation of the ATV within the orchard at all times simplifies the multi-frame fruit tracking system. Other researchers developing fruit yield estimation systems have relied on GPS for vehicle/camera localisation [18–20, 36, 40, 41, 44, 56, 63, 73, 108, 109]. However, the pergola growing system and tall shelter belts of modern kiwifruit orchards can cause occlusion issues with GPS, making it an unreliable solution [75]. Implementing a simultaneous localisation and mapping (SLAM) algorithm is an alternative solution that can both map an orchard and localise the ATV within that map.

The inputs to a SLAM algorithm are a series of measurements from sensors such as lidar, depth cameras, inertial measurement units (IMUs), ultrasonic sensors, radar and vehicle odometery. The output is a map of the traversed area, often in the form of an occupancy grid, and a trajectory of the vehicle through the map. These outputs are computed by matching each new sensor measurement with the previously seen measurements using cost functions and statistical techniques such as Kalman filters and particle filters. An example output can be seen in Figure 4.1. An example of the lidar input can be seen in Figure 4.2

The ATV data collection system has a 3D lidar and an IMU, but no odometry feedback. Not having odometry restricts the choice of SLAM algorithm to those implementations that can be run using lidar and IMU data alone.

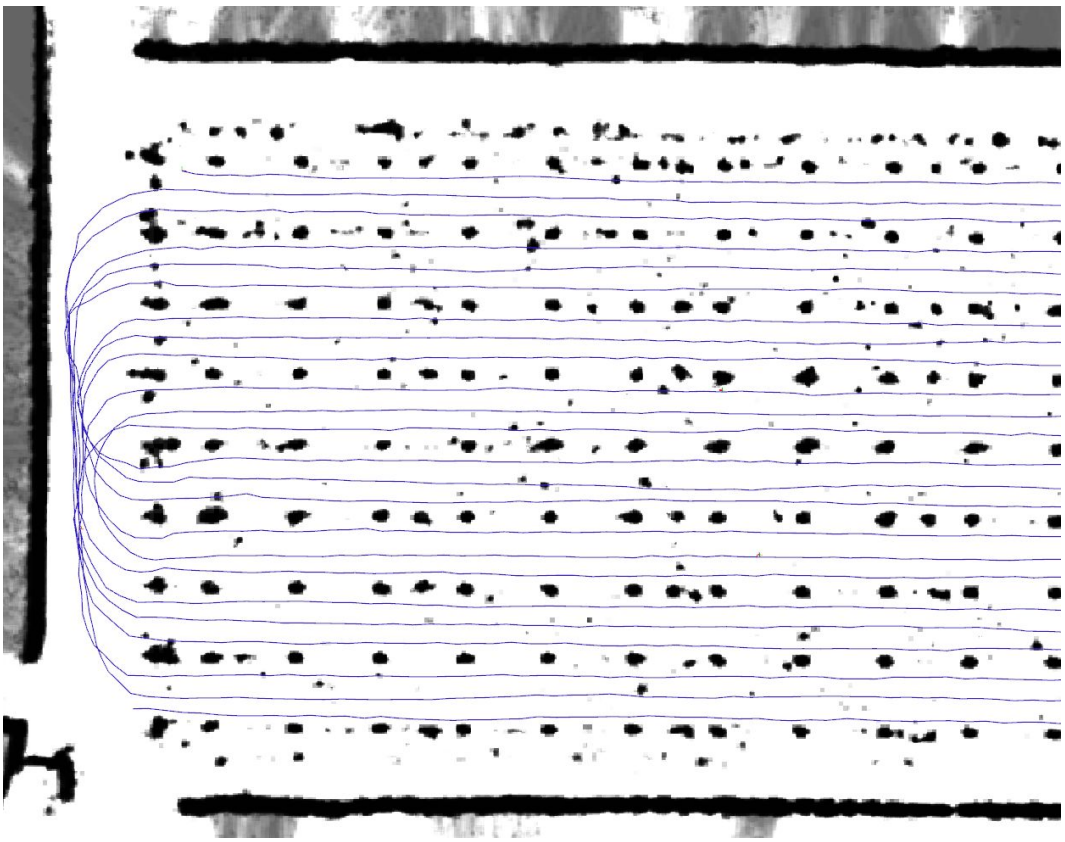


Figure 4.1: An occupancy grid produced by SLAM representing a top down view of one end of an orchard block. Black areas represent occupied areas, which in this case are the trunks, posts, low hanging parts of the canopy and shelter belts of an orchard. The blue line is the computed trajectory of the ATV through the map, which begins in the top left and finishes in the bottom left. Note the image has been cropped on the right for clarity.

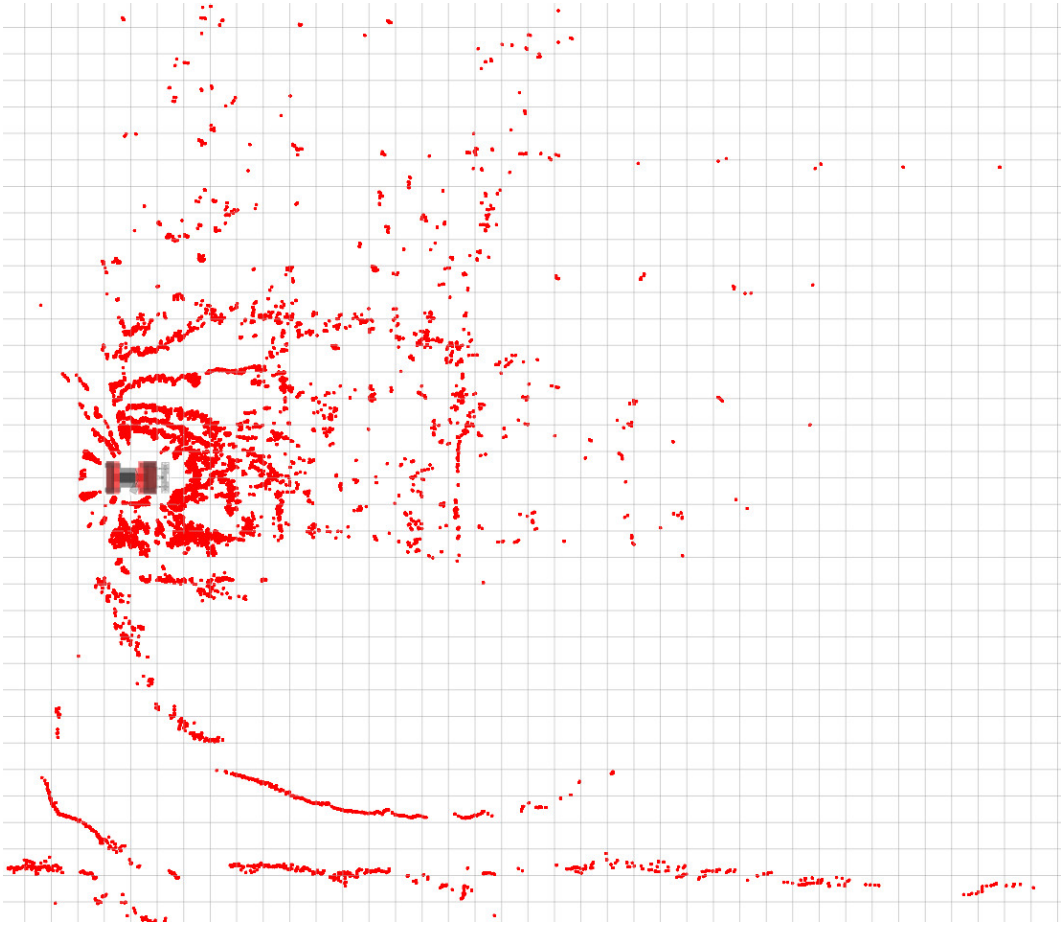


Figure 4.2: A top down view showing the points from a single scan from the lidar. The data collection system can be seen centre left and is travelling towards the right of the image. The line of points along the bottom of the image is a shelter belt. The points surrounding the data capture system are mostly from the ground. The approximate grid of points seen in the upper right of the image are the posts and trunks. The reference grid spacing is 1 m.

Cartographer [105] is selected for use because it is does not require odometry, is the most modern of the systems and is capable of 3D SLAM.

Cartographer is used in 2D mode, which means that it assumes the environment is flat. This is not strictly true in kiwifruit orchards as many are on uneven terrain. However, the reduction in computational complexity outweighs the minor reduction in accuracy.

4.1 Point Cloud Filtering

In each scan produced by the lidar, there are many points that correspond to either the ground or the canopy. When using Cartographer in 2D mode, these points are not useful to the SLAM system as they represent planes that are parallel to the plane of travel. Although these ground and canopy points do not appear to significantly effect the accuracy of the maps, including them slows computation. Therefore, an algorithm to filter these unnecessary points is implemented.

The filtering system consists of two filters both designed to identify points representing vertical features (posts, trunks, shelter belts etc.). Both of the filters are tuned to reduce false negatives (classifying a vertical feature as horizontal). The first filter is a surface normals filter which uses an estimated surface normal to classify each point. The second is a nearest neighbour filter which classifies points based on the number of nearby points. The outputs of the two filters are concatenated to produce the final filtered point cloud (Figure 4.6) that is fed into Cartographer.

4.1.1 Surface Normals Filter

The operating principal of the surface normals filter, is that the normal vector to a vertical surface will be oriented horizontally and vice versa. To estimate a surface normal for each point the `NormalEstimationOMP` function in Point Cloud Library [110] is used. The `NormalEstimationOMP` function analyses

all points within a specified search radius of the point being analysed, fits a surface to the points, and computes the normal to that surface. Only the z (vertical) component of each estimated surface normal is used for classification. The z component will be near zero if the estimated surface normal is oriented horizontally. If the z component of a surface normal lies within a set interval, the point is classified as representing a vertical feature and is kept.

To find an appropriate search radius for the surface normal filter, the point cloud is visualised with each point coloured based on the vertical component of the estimated surface normal. The search radius is then varied to maximise the difference between horizontal surfaces (ground and canopy) and vertical surfaces (posts, trunks and shelter belts). The optimal search radius is found by subjective evaluation to be 0.8 m.

To set the limits for the surface normal filter, the estimated surface normals for 5000 complete lidar scans (approximately 16 million points) are plotted on a histogram (Figure 4.3). There is a cluster of points with surface normals pointing upwards (right side of histogram) which are from the ground. The points with a surface normal pointing downwards (left side of histogram) are from the canopy. There are fewer points from the canopy than the ground because the field of view of the lidar is biased downwards (Chapter 3). Limits are set at -0.8 and 0.7, to minimise false negatives.

As an object is moved further away from the lidar, the number of lidar points on that object decreases. With the Quanergy M8 lidar (as used on the data capture system), the resolution decreases in the vertical direction much quicker then in the horizontal direction (Figure 4.4). In the vertical direction there are 8 layers spaced at 2.85° whereas in the horizontal direction a measurement is taken every 0.12°. At a distance of 15 m from the lidar, the distance between layers (vertical resolution) is 0.75 m, which is approximately half the height of the canopy in some places. Therefore, in some cases, only points from one layer of the lidar will correspond to trunks and posts further than 15 m away. A surface normal estimated using only points from one

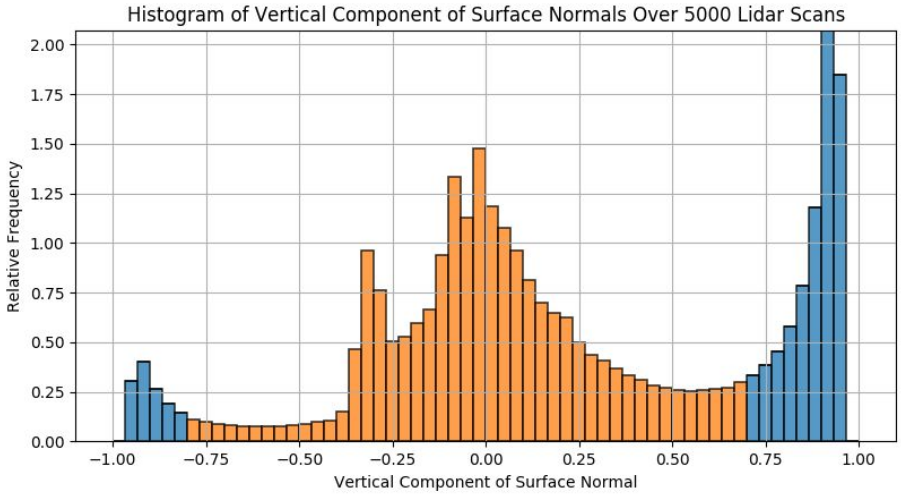


Figure 4.3: Histogram of the vertical component of estimated normal vectors for 5000 lidar scans. The small spike on the left is from the canopy (surface normal points downwards). The large spike on the right is from the ground. Orange bars are wihtin the set limits to be considered to belong to vertical surfaces, blue bars are not.

lidar layer, will always be near vertical as all the points lie on a plane that is near horizontal. Hence, the accuracy of the surface normals filter decreases significantly with range. To overcome this, a second filter is added.

4.1.2 Nearest Neighbour Filter

When only one horizontal lidar layer hits an object, variations in the surface of the object will cause smaller changes in measured distance when the object is oriented perpendicular to the lidar compared to at acute angles. Therefore, tight clusters of lidar points at a medium to large distance from the lidar (>12 m) are more likely from a vertical surface than a horizontal surface.

The nearest neighbour filter is an adaptive cluster detector. It is adaptive because the threshold for classification of a cluster is dependant on distance from the lidar.

For each point further than 12 m from the lidar, the number of points within a set radius of that point are counted. If the number of nearby points is greater than or equal to a threshold, all of those points are classified as

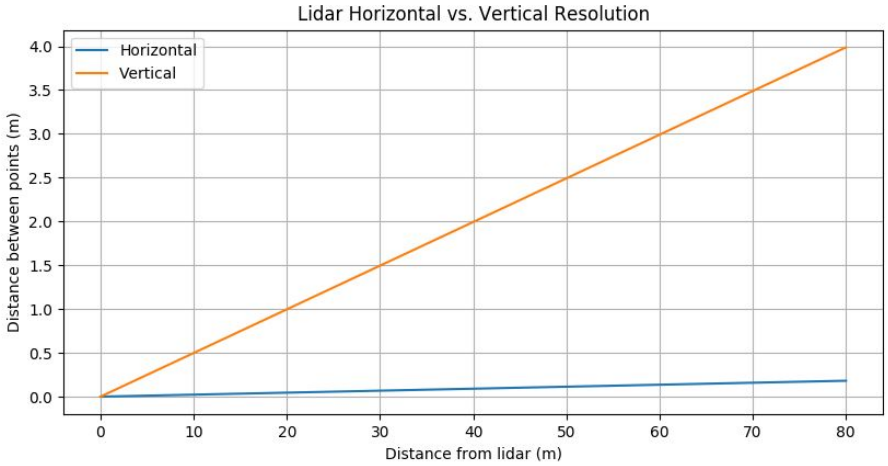


Figure 4.4: Horizontal versus vertical resolution for the Quanergy M8 lidar. The horizontal resolution is much higher than the vertical resolution.

representing a vertical surface. The search radius is 0.25 m, found by ubjective evaluation, and is orchard independent. The formula for the threshold for cluster classification is shown in Figure 4.5.

At 42.5 m from the lidar, the horizontal resolution of the lidar is 100 mm. This is approximately half the width of posts and trunks, meaning, a post or trunk at this distance may only have one lidar point hitting it. Therefore, any points further than 42.5 m from the lidar will be classified as vertical to avoid false negatives.

The combination of the two lidar filters significantly reduces the number of points, decreasing processing time. An example of the lidar filtering can be seen in Figure 4.6.

4.2 SLAM Parameter Tuning

An orchard environment does not contain flat, solid surfaces like walls or floors. Instead they consist of uneven ground covered in long grass, kiwifruit foliage and shelter belts. These surfaces cause noise in lidar measurements as a slight perspective change can be the difference between the lidar sensing a shelter belt or seeing right through it. Therefore, the SLAM output quality in an orchard

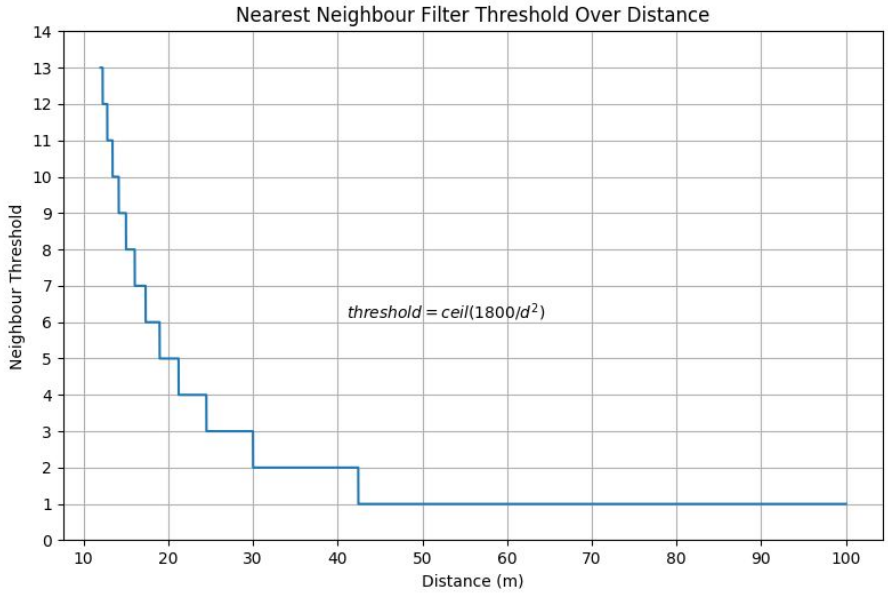


Figure 4.5: The neighbour threshold for the nearest neighbour filter over distance. Points closer than 12 m are excluded from this filter. Points further than 42.5 m will always be classified as vertical as the threshold is 1 point (the point being analysed is included in the count). The `ceil` (ceiling) function rounds a number up to the nearest integer.

will be inferior to that from a structured urban or indoor environment. Tuning of the SLAM system is also more difficult.

Cartographer has many parameters that must be tuned to suit the environment and sensors being used. These range from the resolution of the maps produced to the maximum number of iterations for various optimisation steps to the weights used in cost functions. To find the optimal parameters for Cartographer, SLAM is run on all of the orchard blocks in the training sets of both the fruit and bud datasets. The quality of SLAM output is evaluated for each of the orchards. The process is then repeated with new parameters until a parameter set that produces acceptable results over all of the training orchard blocks is found. More information on the tuning of Cartographer can be found in the tuning guide[1].

[1]https://google-cartographer-ros.readthedocs.io/en/latest/tuning.html

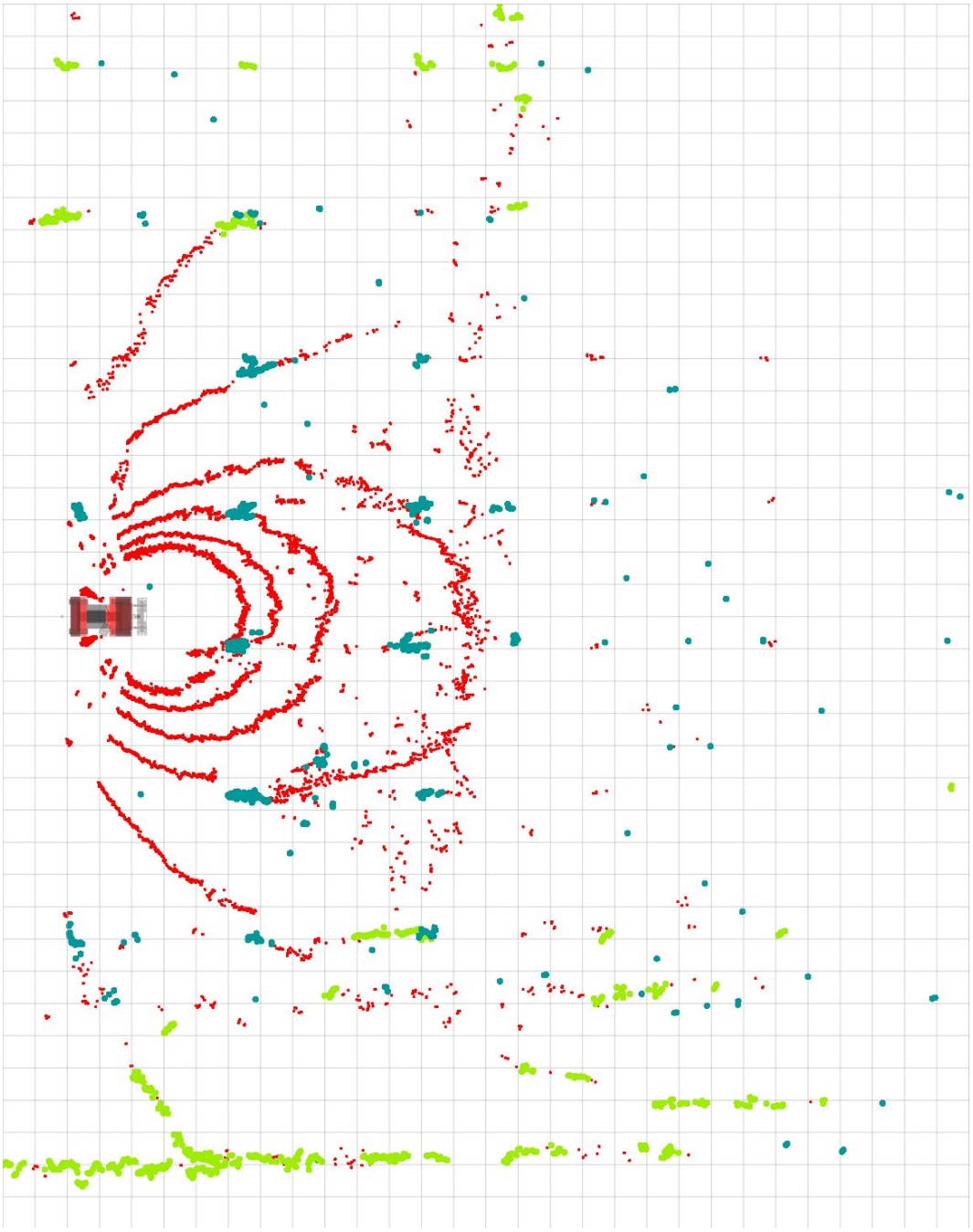


Figure 4.6: A top down view of a single lidar scan. The ATV is shown in the middle left. Red points are those rejected by the filtering system. Blue points have been identified as representing vertical features by the surface normals filter. Green points have been identified as representing vertical features by the nearest neighbour filter. The line of green points along the bottom are from a shelter belt bordering the orchard block. Most of the other green and blue points represent trunks and posts. Reference grid spacing is 1 m.

4.2.1 SLAM Quality Evaluation

Due to the lack of GPS or any other external reference that can be used as ground truth, evaluation of the SLAM output is subjective and conducted by inspection. To reduce the subjectivity of evaluation, criteria ware developed to classify SLAM output as acceptable or unacceptable. An acceptable SLAM output will have the following attributes:

- Trajectory lines are parallel to each other, evenly spaced and do not intersect each other or the rows of trunks and posts. Exceptions to this are when the ATV is turning at the end of rows, or when obstacles in the orchards are avoided.

- Objects in the orchard (trunks, posts, shelter belts etc.) appear well defined and black in colour in the occupancy grid (grey represents uncertainty) without 'ghosting'. Ghosting is when an object is represented more than once in the SLAM output.

An example of acceptable SLAM output can be seen in Figure 4.7. An example of small errors in trajectory, ghosting and hence, unacceptable SLAM output are shown in Figure 4.8. An example of large errors in both trajectory and ghosting is shown in Figure 4.9.

4.2.2 Shade Cloth Occlusion

Some growers install shade cloth between some of the rows (usually every 2–3 rows) of their orchards (an example can be seen in Figure 4.10). The shade cloth creates a series of 'hallways', each a few rows wide, where the lidar cannot see into the adjacent hallway. As the ATV is driving down each hallway, the SLAM system must dead reckon the ATV's position based only on reference points in the current hallway. Small errors accumulate down the length of the hallway and, if large enough, cannot be corrected by the loop closure system built into Cartographer. For orchards with short rows (less than 100 m), acceptable SLAM output is achieved with well tuned SLAM parameters.

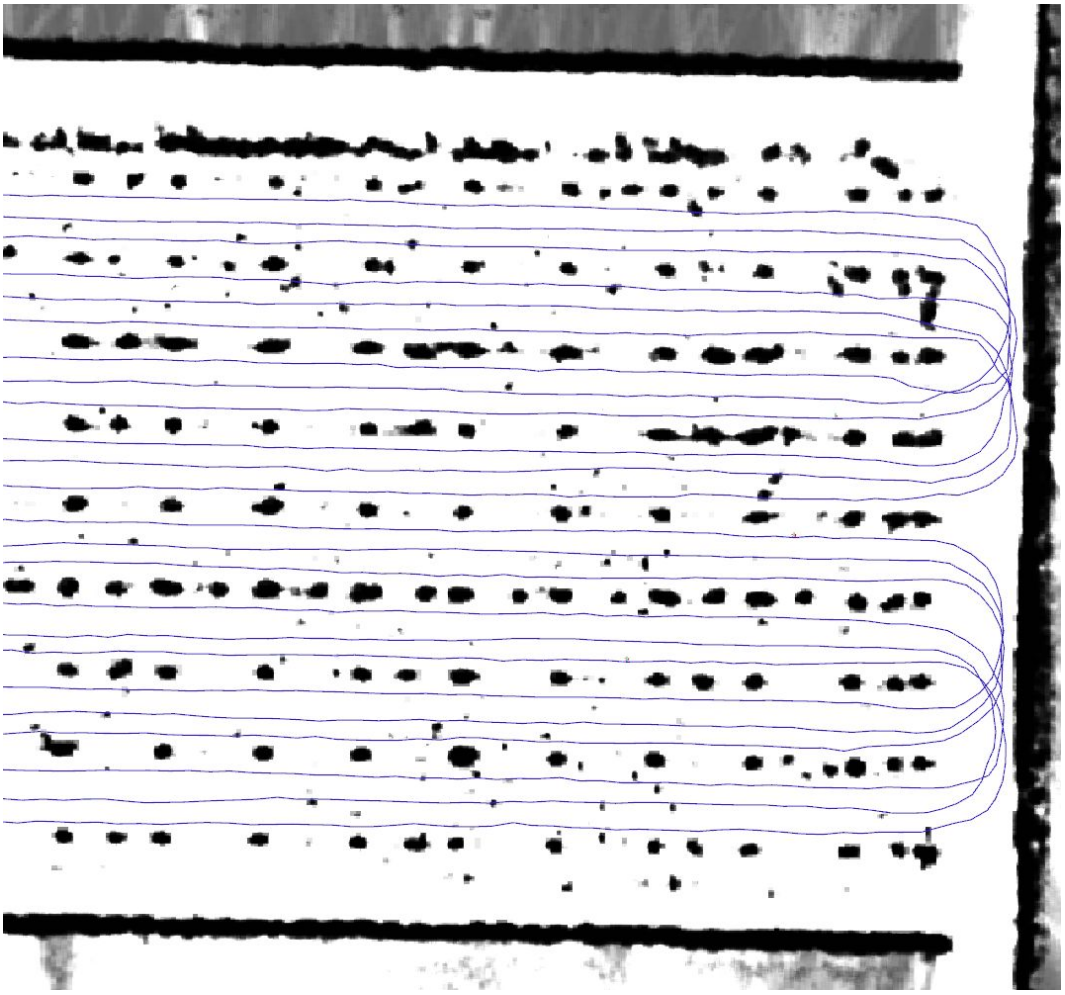


Figure 4.7: An example of acceptable SLAM output. Trajectory lines are parallel to each other, evenly spaced and do not cross over each other (apart from when turning at the end of rows). Posts, trunks and other features are well defined with no ghosting. Note only a section of the orchard is shown for clarity.

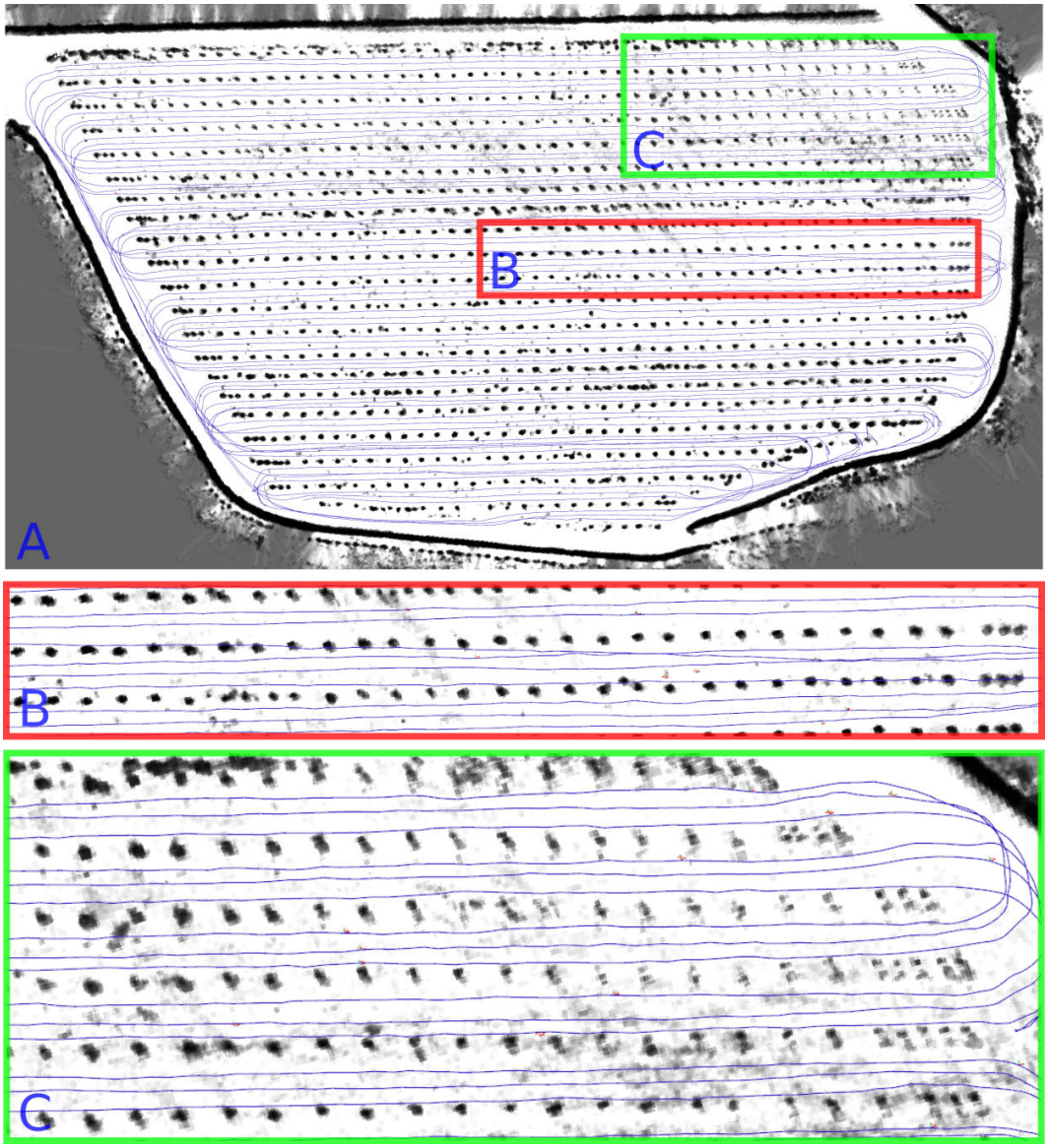


Figure 4.8: An example of unacceptable SLAM output. Panel A shows the entirety of the SLAM output. Panel B shows a section containing trajectories that are not parallel to each other and cross over each other and a row of trunks. Panel C shows a section with ghosting, particularly in the right side of the image. Many of the features are grey in colour and can be seen twice.

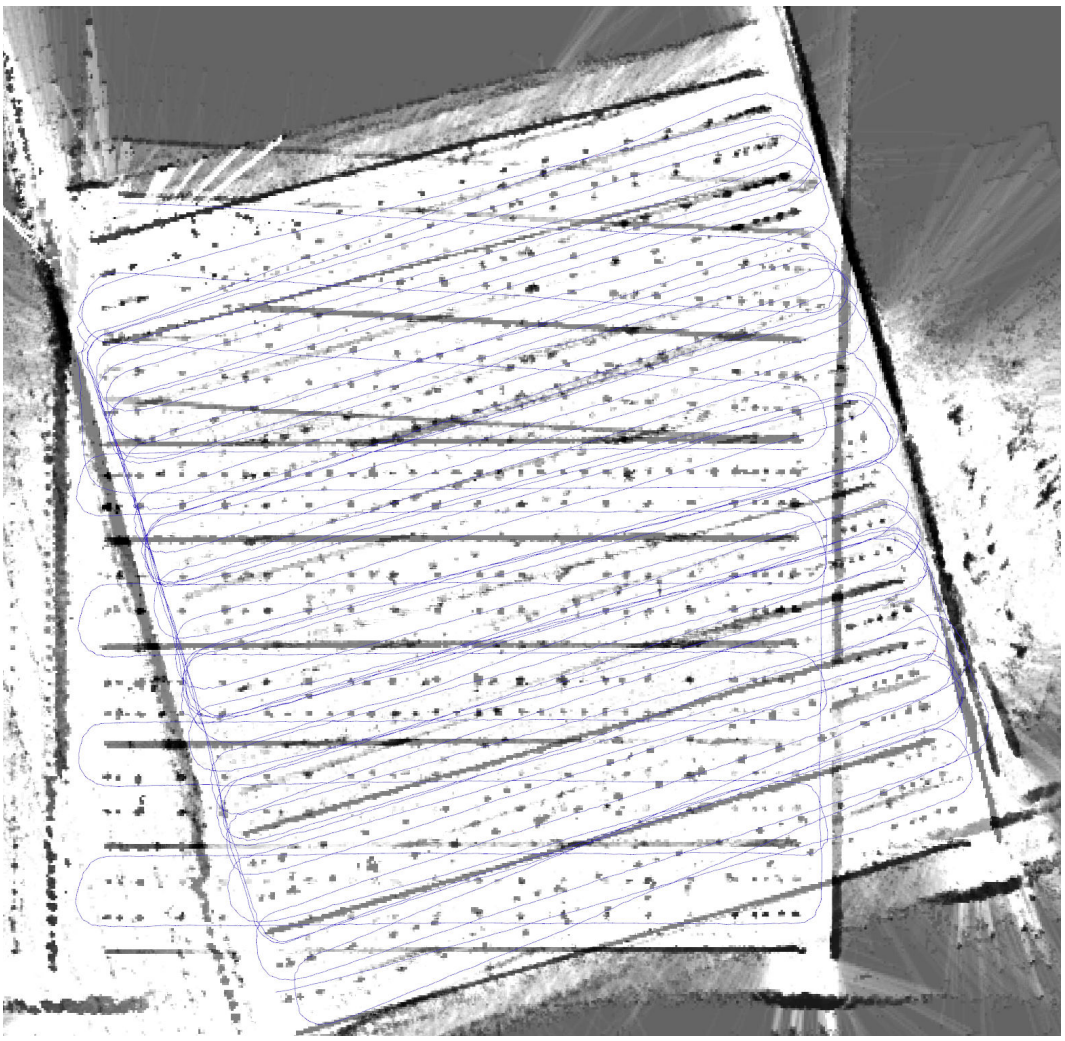


Figure 4.9: An example of unacceptable SLAM output. Severe ghosting can be seen with the entire map seen twice due to a rotation error. Trajectories are not parallel to each other and overlap in multiple locations.

Figure 4.10: An example of an orchard with shade cloth between every second row. The shade cloth is the white material.

However, in orchards with longer rows (100–240 m) acceptable SLAM is not achieved.

The shade cloth occlusion issue is discovered between bud data collection and fruit data collection. Therefore, only orchards free of shade cloth are visited for fruit data collection.

Experiments comparing the effects of shade cloth occlusion with the VLP16 (Velodyne LiDAR, San Jose, California, USA) lidar to the Quanergy M8 lidar, used for the rest of data collection, are conducted (Figure 4.11). Additionally, experiments are conducted with a lidar mounted to the rear of the ATV to provide more data to the SLAM system. However, these experiments are interrupted by unrelated hardware issues and could not be resumed due to limited availability of the additional lidar sensors. The experiments did show that the VLP16 lidar is less effected by shade cloth occlusion than the M8 (Figure 4.12). The additional reference points from beyond the shade cloth should result in higher quality SLAM output. However, because of the interruptions to the experiments, definitive conclusions cannot be drawn, and this problem is left for future work.

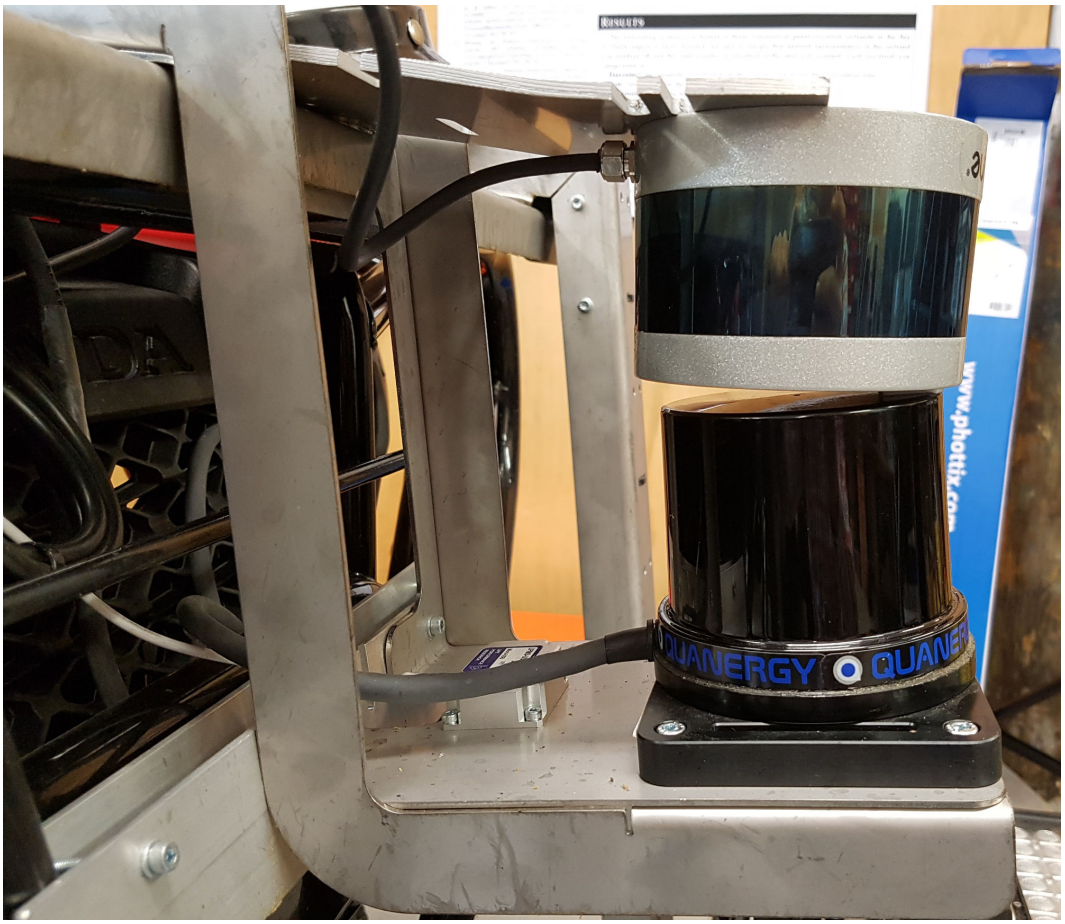


Figure 4.11: Side view of the Velodyne VLP16 lidar mounted above the Quanergy M8 lidar. Experiments are conducted to evaluate the effect of shade cloth occlusion with different lidar sensors. Note, the M8 is tilted backwards 7° whereas the VLP16 is mounted horizontally due to the symmetrical vertical field of view of the VLP16.

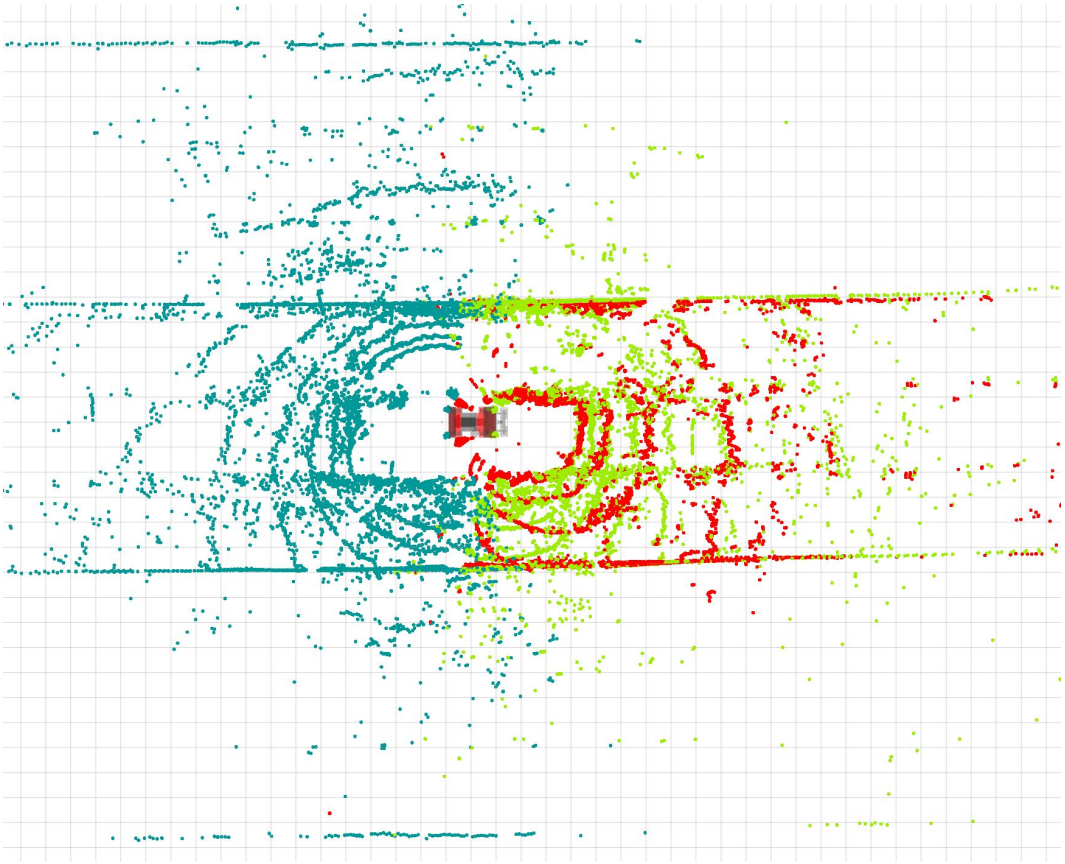


Figure 4.12: Top down view of lidar point clouds with multiple lidar sensors. The ATV is shown near the centre of the image. Red points are from the Quanergy M8 lidar used for all data collection. Green points are from a Velodyne VLP16 mounted above the Quanergy lidar. Blue points are from a second Velodyne VLP16 mounted on the rear of the ATV. The shade cloth can be seen in the lidar points above and below the ATV. Reference grid spacing is 1 m.

Parameter	**Primary**	**Alternate**
use_imu_data	**true**	**false**
num_accumulated_range_data	**4**	**5**
motion_filter.max_time_sections	**5**	**0.5**
motion_filter.max_distance_meters	**1**	**3**
motion_filter.max_angle_radians	**10°**	**3°**
submaps.num_range_data	**450**	**75**
optimize_every_n_nodes	**500**	**50**
num_subdivisions_per_laser_scan	1	1
use_online_correlative_scan_matching	true	true
submaps.resolution	0.3	0.3

Table 4.1: The key configuration parameters used for Cartographer. Values that are different between the primary and alternate configurations are shown in bold.

4.2.3 Final SLAM Configuration

A single set of parameters that produces acceptable SLAM output across all of the training orchard blocks (excluding those with shade cloth between rows) could not be found. Instead a primary and an alternate parameter set are implemented. The primary set produces acceptable SLAM output in 40 of the 43 training blocks. The alternate parameter set produces acceptable SLAM in 39 of the 43 training block, but more importantly, produces acceptable SLAM output on the 3 blocks that the primary parameter set did not. The key parameters and differences between the primary and alternate parameter sets are shown in Table 4.1. An analysis of the results could not identify a single reason why one parameter set performed well in most orchards, but not in others.

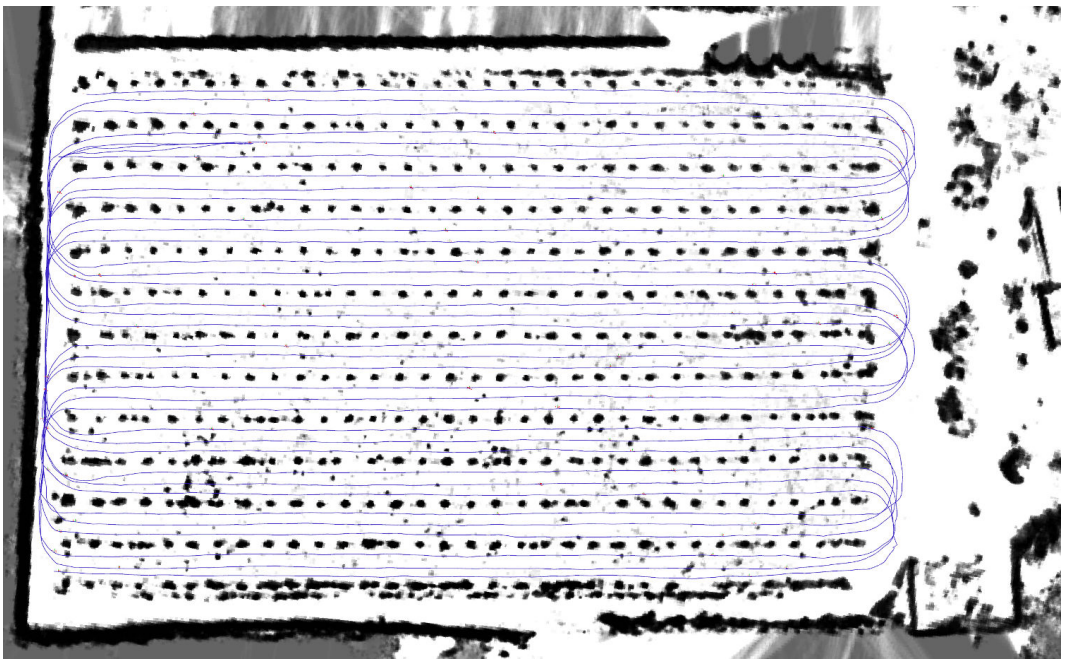


Figure 4.13: An example of acceptable SLAM. The black dots represent trunks and posts. The lines bordering the orchard on the top, bottom and left are shelter belts. On the right are trees, a shed and a house.

4.3 SLAM Evaluation

Cartographer is run on each of the orchard blocks from the fruit dataset (training, test and validation datasets) using the primary parameter set. The quality of the SLAM output is evaluated using the criteria detailed in Section 4.2.1. For those blocks where acceptable SLAM output is not produced, Cartographer is run again with the alternate parameter set. Quality of SLAM output is again evaluated. Acceptable SLAM output is produced for all orchard blocks from the fruit dataset.

Chapter 5

Canopy Density

The density of the canopy is one of the factors that may contribute to variation in occlusion rate. Intuitively, higher foliage density should result in a higher rate of occlusion. Measuring the foliage density directly is difficult without additional sensors. Therefore, as a proxy for canopy density, the ratio of sky pixels to canopy pixels is estimated.

5.1 Sky Detection Algorithm

Sky pixels are distinct from canopy pixels as they are bright and have a high blue component. A simple colour threshold in the RGB colour space can distinguish between sky and canopy. The following thresholds are found by trial and error:

$$R > 100$$

$$G > 100$$

$$B > 235$$

The binary threshold is followed by a morphological close with a 3×3, 'plus' shaped kernel:

$$\begin{matrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{matrix}$$

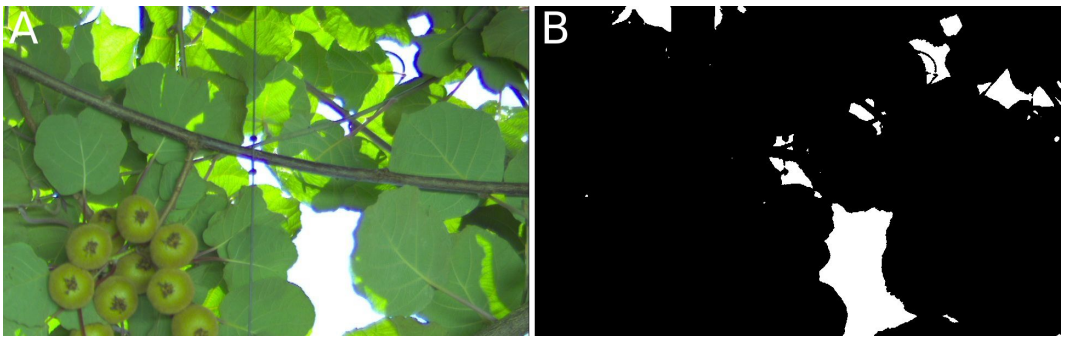


Figure 5.1: An example of sky detection. Panel A shows the raw image. Panel B shows the output of the sky detection algorithm, where white is sky.

The total number of white pixels is then counted and divided by the total number of pixels in the image to give a canopy coverage percentage. The algorithm is applied to all images taken within rows of an orchard block. The mean of all canopy coverage values from an orchard block is taken.

An example of the sky detection is shown in Figure 5.1. The white sections in panel B show the regions of the image identified as sky.

5.2 Evaluation Method

Sky pixels are manually labelled in a random selection of 17 images from the fruit training dataset using the Labelbox labelling tool[1]. The images are taken in a variety of weather conditions. Some images have the sun directly overhead, some are overcast, some have clear blue sky. However, none of the images are taken in low light conditions (dawn, dusk, night-time) as all orchard data was collected during the day.

When labelling sky pixels in images there is ambiguity. Precisely classifying each pixel as either sky or background is a difficult and time consuming task, even for a human. Approximations are made to represent each area of sky with a polygon (Figure 5.2).

A logical XOR operation is applied to the ground truth and algorithm produced sky masks to identify incorrectly classified pixels. The number of

[1]https://www.labelbox.com

Figure 5.2: A section of an image showing ground truth sky labels. The red outlined areas are the sky labels. Note the approximations made and the ambiguity around the edges of the labelled areas.

mismatching pixels is counted. Results are averaged across all of the images in the evaluation set.

The processing time is measured across the evaluation images. The CPU used is a Xeon E5-1650 v3 (Intel, Santa Clara, California, USA). Image loading is not included in the timing, only calculating the canopy coverage percentage from a preloaded image.

5.3 Results

Across the 17 evaluation images 0.57% of all pixels are misclassified (Table 5.1). A typical example of sky detection is shown in Figure 5.3. Panel D shows each pixel as a true positive, true negative, false positive or false negative.

Processing time is 5.3 ms per image. With 28 frames captured per second (7 frames per second on each of 4 cameras), this equates to 148 ms of processing time per second of captured data. However, this does not take into account

Images	17
Misclassified	0.57%
True Positives	6.74%
True Negatives	92.69%
False Positives	0.25%
False Negatives	0.33%
Sensitivity	0.95
Specificity	1.00
F_1**-score**	0.96
Processing Time	5.3 ms (per image)

Table 5.1: Sky detection evaluation statistics on a per pixel basis. Percentages are expressed as a percentage of the total number of pixels.

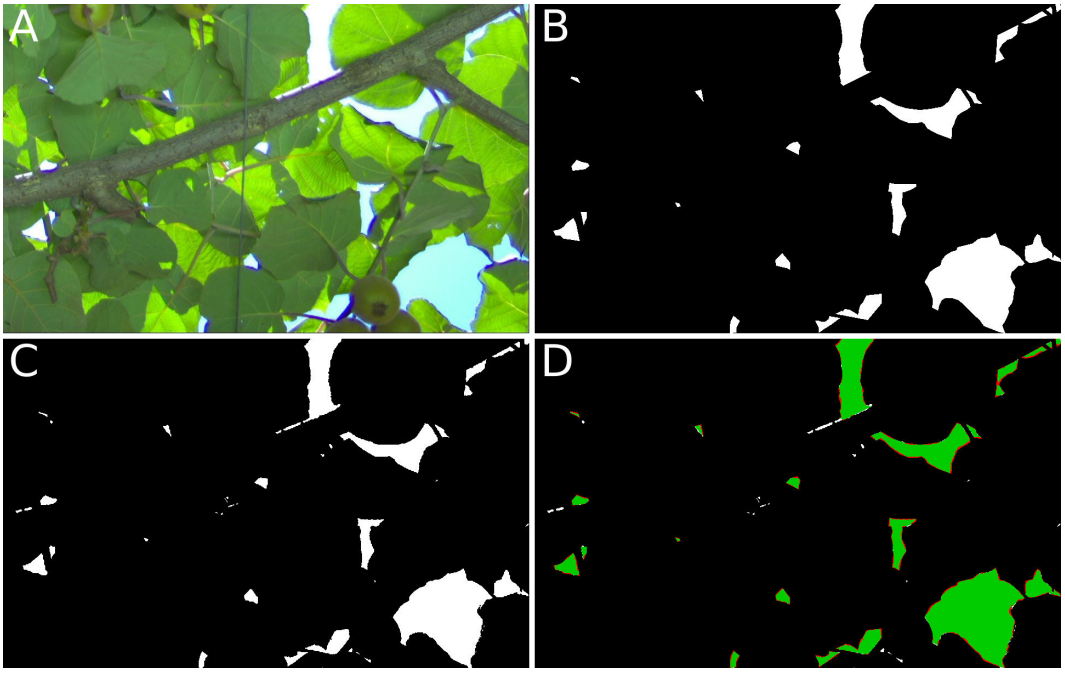


Figure 5.3: Sky detection evaluation. Panel A shows the raw image. Panel B shows the ground truth, where white is sky. Panel C shows the output of the sky detection algorithm. Panel D shows true positives in green, true negatives in black, false positives in red and false negatives in white.

multi-threading which is implemented for use in bulk processing of images.

The majority of false negatives are around the edges of sky areas. This is due to the ambiguity and approximations made in the ground truth labels (Figure 5.2).

The main cause of false positives is direct sunlight reflecting off of branches or stems. The intense reflections are very bright spots in the images that are misclassified as sky. An example is shown in the upper centre of the images in Figure 5.3. However, these reflections represent a very small percentage of the overall image and thus have a small impact on accuracy.

The algorithm would likely fail in low light conditions as it relies on the sky being bright. A more robust approach would be required if this system is to be used at night.

The algorithm takes a simple approach to averaging over a block with each image being equally weighted in the final average. Therefore, variations in overlap between images are not taken into account. Image overlap variations can be caused by varying row widths, inaccuracies in pass trajectory and variation in vehicle speed. For example, in an extreme case, the vehicle may be stopped for a period of time underneath a hole in the canopy. Images are still being captured while the vehicle is stationary and the algorithm will give them equal weighting when calculating the canopy coverage average for the block. Thus the calculated average will be lower than it is in reality. For the data captured for this study, this kind of biasing is not expected to be an issue as significant stoppages are not present in the data. However it should be considered for future work.

The algorithm is also susceptible to errors in row labels. The timing of a `row_start` or `row_end` label is subjective as it is controlled by the operator at time of data capture. If, for example, a `row_start` label is placed prematurely, images taken before entering a row will be used in the canopy coverage calculation. The result is a calculated average that is lower than it is in reality. This could be overcome by implementing an automated in-row detection

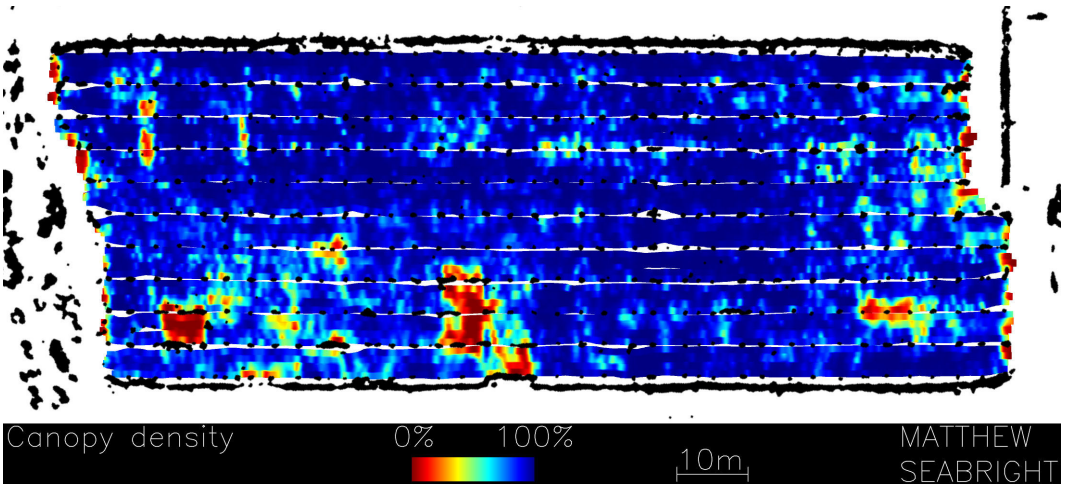


Figure 5.4: The canopy density of an orchard block shown from a top down view. Red areas represent gaps in the canopy, dark blue represents dense canopy. The black features are the trunks, posts, shelter belts, fences, trees and other features as mapped by the SLAM system.

system rather than relying on the operator to correctly place labels.

5.4 Canopy Density Maps

Using the trajectory output obtained from SLAM (Chapter 4), the canopy density can be represented spatially over each block. This gives growers an insight into their orchards and can be used to identify areas of canopy that need attention. An ideal orchard has a consistent and dense canopy for maximum light absorption and growing area. An example can be seen in Figure 5.4 where multiple large gaps in the canopy can be easily identified (red spots).

The accuracy of the canopy density map in Figure 5.4 is verified by taking it into the orchard and comparing the map and orchard. There is a clear correspondence between the two with even the small holes shown on the map being visible in the orchard. A more through validation of the accuracy would be a time consuming task and is outside of the scope of this work.

Chapter 6

Kiwifruit Detection

The kiwifruit detection system is the foundation of the yield estimation system. A high performing detection system is able to reliably detect fruit across a variety of lighting conditions, fruit sizes, shapes and distances from the camera.

6.1 Detection System Options

Other researchers have demonstrated the high performance of convolutional neural networks (CNNs) for fruit detection (see Section 2.1). In particular, Faster R-CNN, YOLO and SSD have all been applied to fruit detection tasks with impressive results [22, 35, 39, 67, 68, 74].

Faster R-CNN is a bounding box object detection system developed by Facebook AI Research (Menlo Park, California, USA) [85]. It is a two stage system. The first stage is a region proposal network that generates candidate object bounding boxes. The second stage extracts features and performs classification and bounding box regression.

Mask R-CNN is an extension of Faster R-CNN that produces segmentation masks for each object instance in addition to bounding boxes [111]. Segmentation masks are predicted by a branch section of the neural network) that is added in parallel to the bounding box recognition branch. The extra computation adds only a small speed penalty.

You Only Look Once (YOLO) is a real time bounding box object detection

system [83]. A single convolutional network predicts bounding boxes and class probabilities from one evaluation of an input image. The latest iteration, YOLOv3, improves on the accuracy of the previous versions [112]. The single stage approach makes YOLO much faster than the two stage systems such as Faster R-CNN, however accuracy is generally lower.

Single Shot Detector (SSD) is a single stage bounding box object detection system [84]. It is similar to YOLO in design but has many more bounding box locations and uses a multi scale approach. The authors claim it is more accurate than Faster R-CNN while being faster than YOLO.

RetinaNet is a bounding box object detection system developed by Facebook AI Research (Menlo Park, California, USA) [113]. RetinaNet is a single stage system (like YOLO and SSD) but is trained using a modified loss function called focal loss. Focal loss is designed to emphasise examples that are most difficult for the network to classify, and de-emphasise easy example. This modification can improve the accuracy of the trained network, with the authors claiming accuracy on par with two stage detections such as Faster R-CNN.

Mask R-CNN is selected for the kiwifruit detection system. The superior accuracy of Faster R-CNN over single stage detection systems like YOLO and SSD is deemed to be more important than the higher speed these alternative systems offer. Mask R-CNN is chosen over Faster R-CNN as the addition of masks could be beneficial for size estimation of fruit (size estimation is not implemented in this work). Requiring semantic labelling of fruit masks is a severe drawback to Mask R-CNN as producing semantic labels can take more than 10 times as long as simple bounding boxes [39].

6.1.1 Mask R-CNN Configuration

The Matterport (Sunnyvale, California, USA) developed version of Mask R-CNN for Tensorflow is used [114, 115]. Resnet101 is used as the backbone of the detector. Images are resized from 1920×1200 pixels to 1024×1024 with the original aspect ratio preserved using black bars on the top and bottom of

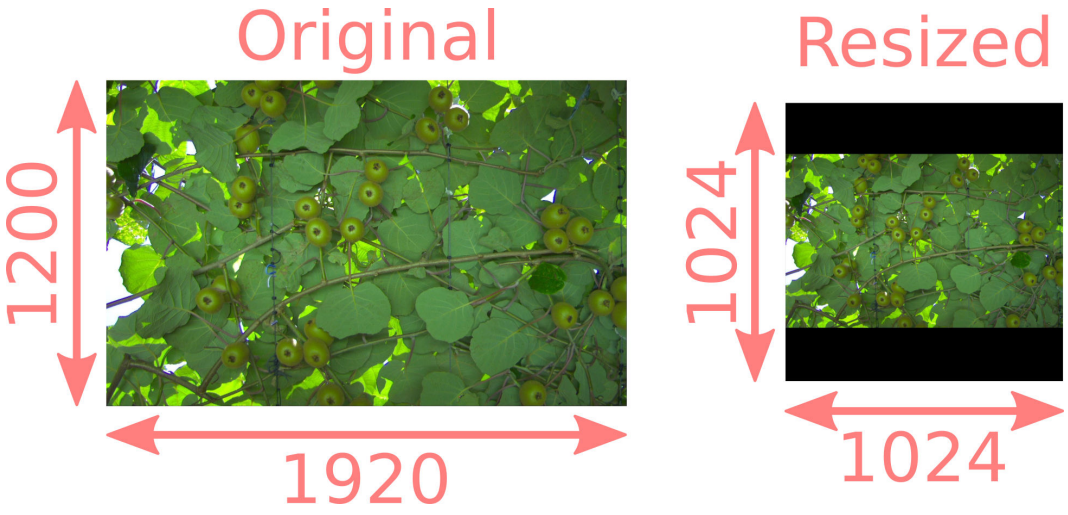


Figure 6.1: Images are resized from a resolution of 1920×1200 to 1024×1024 with black bars top and bottom.

Backbone	Resnet101
Backbone strides	[4, 8, 16, 32, 64]
Region proposal network ratios	[0.5, 1, 2]
Region proposal network scales	(12, 20, 34, 54, 80)
Region proposal network stride	1
Detection minimum confidence	0.9
Detection maximum instances	250
Use mini mask	True

Table 6.1: Mask R-CNN configuration parameters.

the resized image (Figure 6.1). Anchor sizes are adjusted from the defaults to better suit the size of kiwifruit in the images. The important Mask R-CNN configuration parameters are shown in Table 6.1.

6.2 Image Labelling

To train the kiwifruit detection CNN, labelled data is required. The Labelbox image labelling tool[1] is used to label 109 randomly selected images from the fruit training dataset. More images would be labeled if time allowed. Each

[1]https://www.labelbox.com

Figure 6.2: Example of a labelled image. Individual masks are used to label each fruit (pink). Bounding boxes are used to label calyxes (white).

Images labelled	109 (69 training, 20 validation, 20 testing)
Kiwifruit labels	4715
Calyx labels	4065
Mean calyxes per image	37.3
Mean time per image	22.2 minutes
Total labelling time	40.3 hours

Table 6.2: Statistics on labelled data.

kiwifruit is labelled with a mask and each kiwifruit calyx is annotated with a bounding box as shown in Figures 6.2 and 6.3. Fruit that are intersected by a wire, branch, leaf or other object are labelled as one fruit, with the mask split into sections (examples are shown in Figure 6.3). Labelling is a time consuming process with each image taking an average of over 22 minutes (Table 6.2). The labelled images are randomly split into a training set (69 images), a validation set (20 images) and a testing set (20 images).

Figure 6.3: A second example of a labelled image. Two instances of fruit being split by a branch can be seen near the centre of the image, and another two near the top right.

6.3 Training

Training is run on a Titan Xp GPU (Nvidia, Santa Clara, California, USA) and takes approximately three days. The 69 images in the training set are used for training. The network is trained on both the kiwifruit and calyx classes simultaneously. The training parameters used are shown in Table 6.3. Pre-trained weights for the Common Objects in Context (COCO) dataset are used for the network [116].

6.4 Inference

Inference is run on a PC with four GPUs (Table 6.4). To maximise performance and GPU utilisation, the inference pipeline is split into multiple threads (Figure 6.4). One thread is dedicated to loading images from the rosbags they are stored in, uncompressing and undistorting them and adding them to the inference queue. Sixteen inference threads (four per GPU) take images from the queue, run inference and place the results in an output queue. The final

Learning rate	0.001
Learning rate decay	0.0001
Learning momentum	0.9
Steps per epoch	200
Batch size	1
Number of epochs	2000

Table 6.3: Mask R-CNN training parameters.

CPU	AMD Threadripper 1950x (3.4 GHz, 16 cores)
Memory	64 GB DDR4
GPU1	Nvidia Quadro P6000 (24 GB)
GPU2	Nvidia Quadro P6000 (24 GB)
GPU3	Nvidia Titan Xp (12 GB)
GPU4	Nvidia GTX 1080 Ti (11 GB)

Table 6.4: Inference PC specifications.

thread takes the data from the output queue, trims the unneeded data and write the output to a file. Using this pipeline, inference runs at 22 fps, which is slightly lower then real-time (28 fps).

6.5 Evaluation

Evaluation is performed using the tools built into the Matterport version of Mask R-CNN. The 20 images of the testing set are run through the detection system and the results compared to the ground truth labels. Using a bounding box intersection over union (IoU) of 0.5, the mean average precision (mAP, area under the precision-recall curve, averaged over both classes and all 20 images) is 0.90.

In most cases, the detection system performs very well. A typical example is shown in Figure 6.5. In some images, the sun is directly overhead and

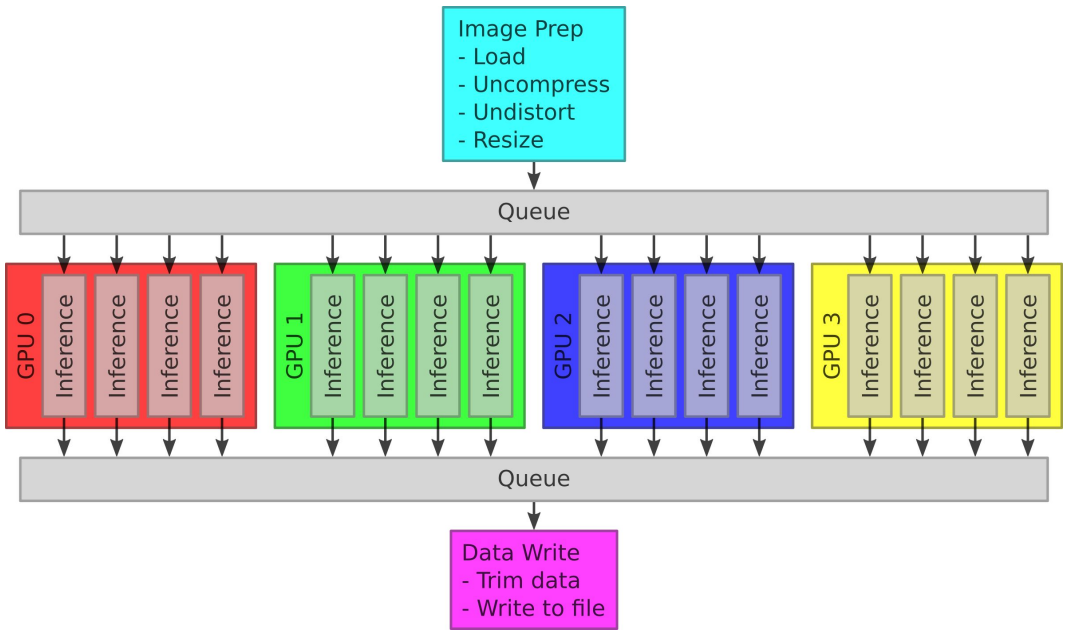


Figure 6.4: The inference pipeline. Images are loaded from rosbags, uncompressed and undistored by the image preparation thread. Each of the GPUs runs four instances of Tensforflow, performing inference. One thread is dedicated to taking the inference output, and writing it to a file.

there are gaps in the canopy, causing the image to be very bright. In these challenging conditions, the detection system still detects fruit as seen in Figure 6.6. The detection system is capable of producing a single mask that is split into two segments where a fruit is intersected by a wire or branch (Figure 6.7).

6.5.1 Failure Cases

There are cases where the detection system produces incorrect results. The most common error is false positive calyx detections, which are often caused by small dark spots in the images. Holes in leaves and the ends of the clips used to train the canes are often incorrectly classified as calyxes (Figure 6.8). Less common is leaves being misclassified as kiwifruit as seen in Figure 6.9.

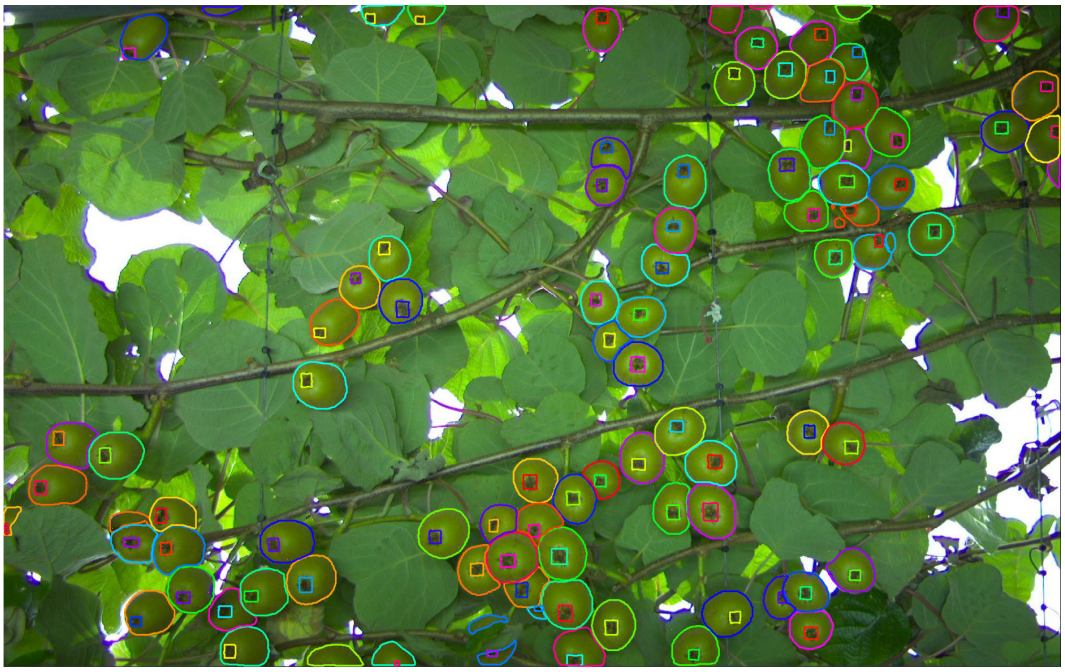


Figure 6.5: An example of fruit and calyx masks produced by Mask R-CNN. The colours represent different instances of an object (each fruit is a separate instance). The calyx masks are very rectangular as bounding boxes are used for training as opposed to the masks used for fruit.

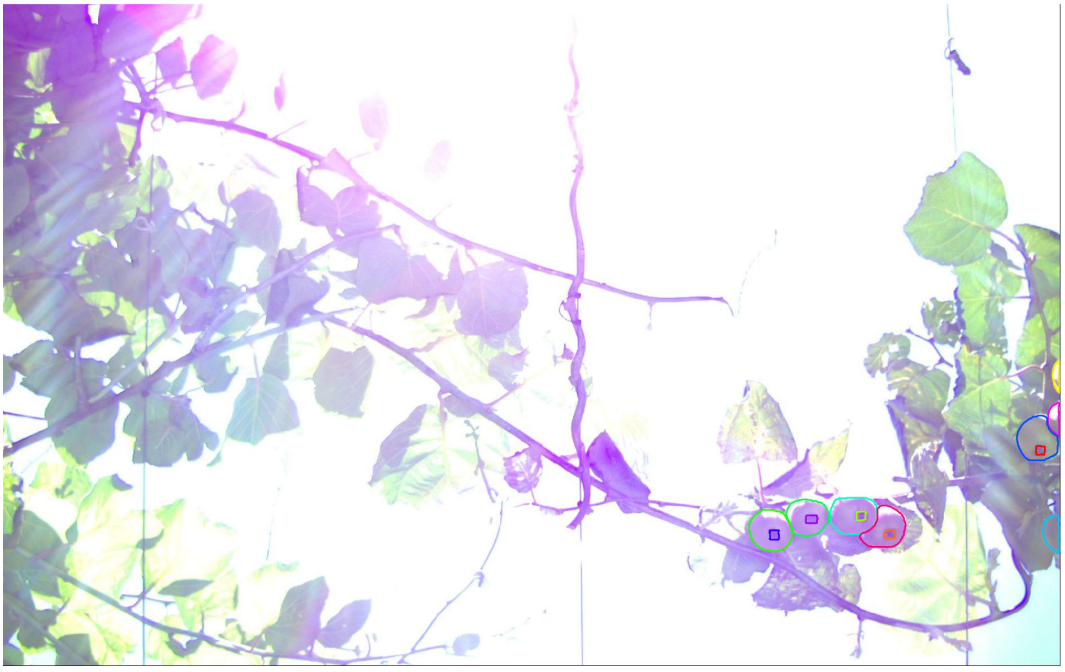


Figure 6.6: An example of direct sun light on the camera. The detection system is still able to detect the fruit.

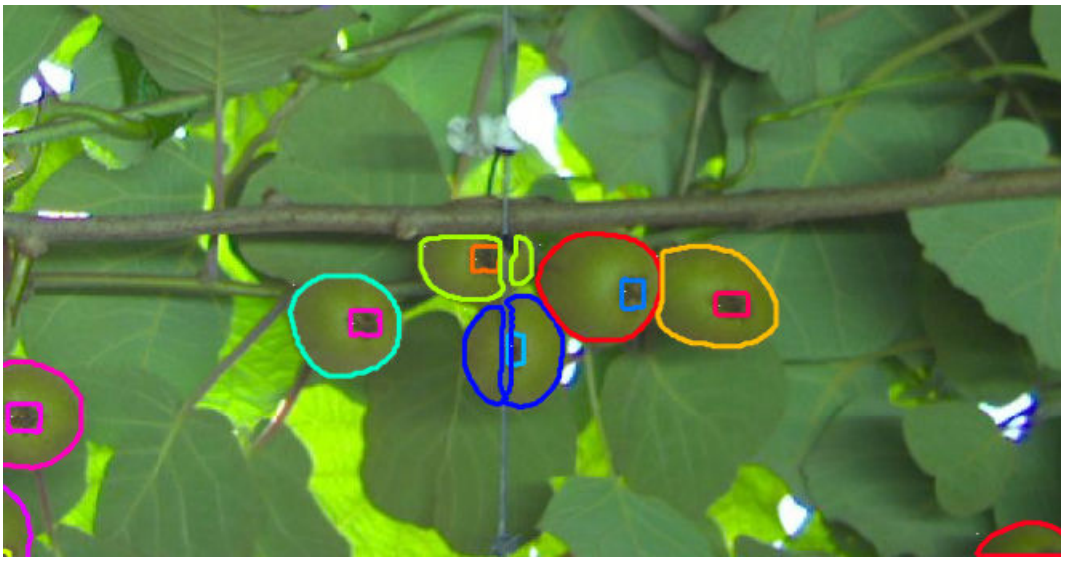


Figure 6.7: An example of split masks caused by a wire intersecting the fruit. The two sections of the mask having the same outline colour shows that they are counted as the same fruit.

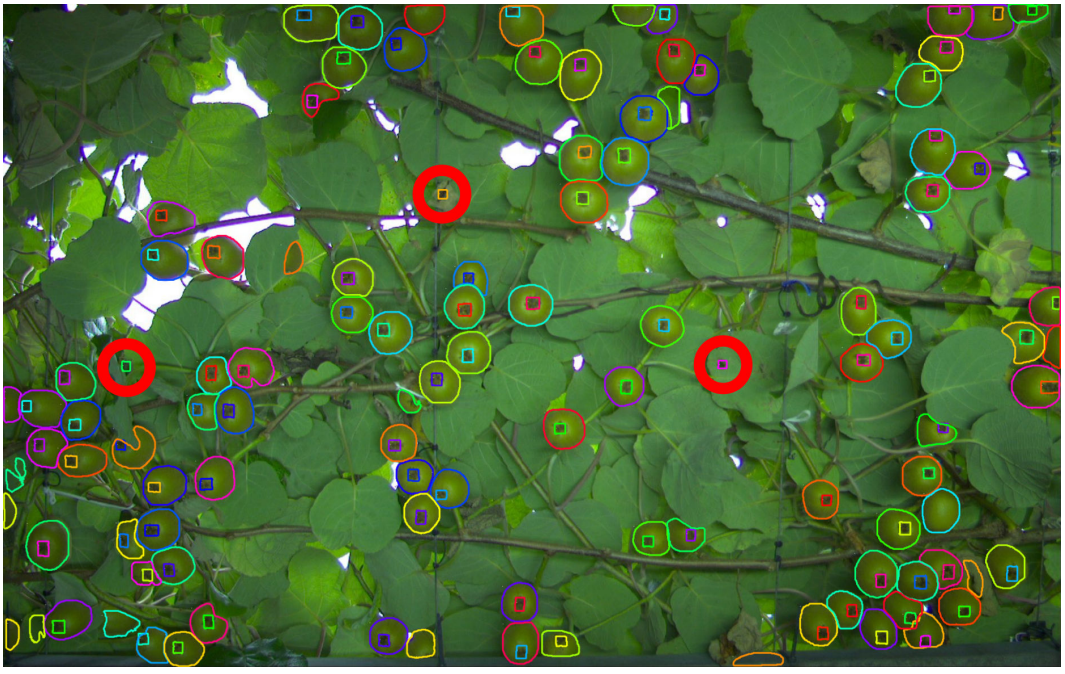


Figure 6.8: An example of an image with multiple false positive calyx detections, circled in red.

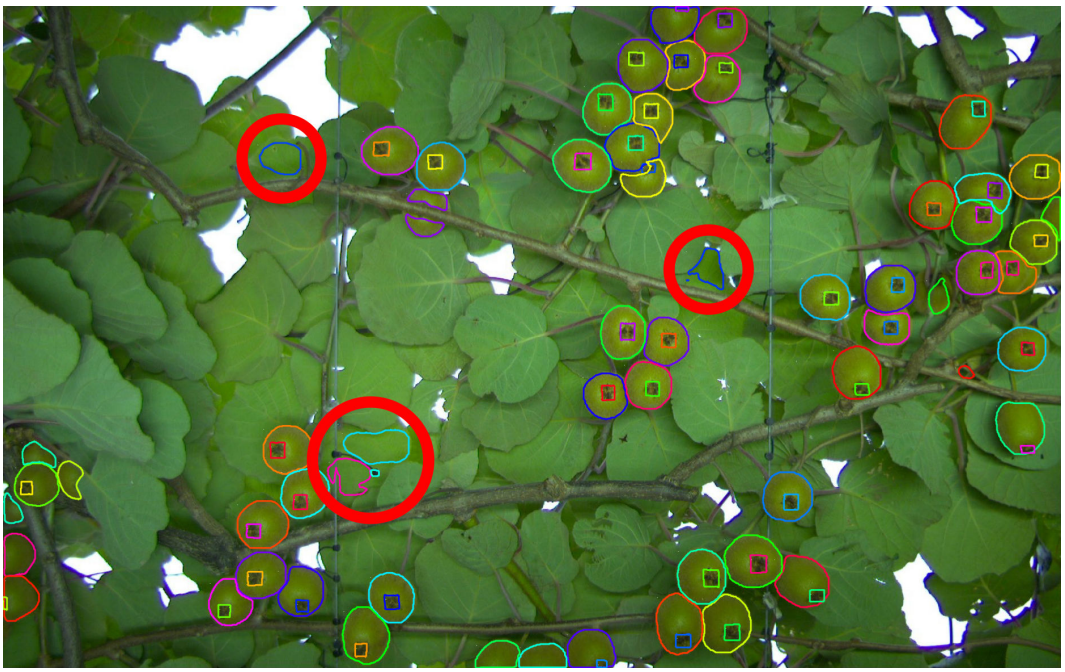


Figure 6.9: An example of an image with multiple leaves mistaken for kiwifruit, circled in red.

Chapter 7

Kiwifruit Localisation

Before detected kiwifruit are localised using stereo triangulation, the stereo correspondence problem must be solved. The stereo correspondence problem is determining which objects in one image of a stereo pair correspond to which objects in the other image of the stereo pair. There are two approaches to solving this problem, dense stereo and sparse stereo. A dense stereo system finds correspondence on a pixel level, whereas a sparse stereo system finds correspondence on an object level. Sparse stereo systems are typically less computationally intensive as they are only finding correspondence for a low number of objects. Anecdotal evidence shows that dense stereo methods are neither accurate or fast enough in kiwifruit applications. Therefore, a sparse stereo approach is pursued [117].

The objects to be localised by the sparse stereo matching system are kiwifruit calyxes as detected by the detection system. The calyxes are used for triangulation as they are smaller in area than the entire fruit and therefore, their position in the image is less effected by partial occlusion. The centre of the calyx bounding box is used as the keypoint for triangulation. A reduced search space, sparse stereo matching algorithm is developed to effectively match fruit.

7.1 Stereo Matching Algorithm

The stereo matching algorithm requires pre-existing knowledge of both the camera geometry (Section 3.3) and the expected scene geometry. This knowledge is used to construct a search window for each of the detected kiwifruit calyxes (referred to as 'keypoints' henceforth). These search windows represent the area in the other image where the matching keypoint should be found, given the camera and scene geometry. The geometry of the search window is shown in Figure 7.1. For each keypoint in one image of a pair, the epipolar line for the other image is calculated using OpenCV (using the `computeCorrespondEpilines` function). The epipolar line represents where the matching keypoint should be found in the other image based on epipolar geometry. The search window are centered on these epipolar lines. The width (`w`) and position of the search window along the epipolar line (`o`) is defined by the camera geometry and the expected object distance from the cameras (mean stereo disparity `d`). Ideally, the search window would have a height (`h`) of one pixel as the matching keypoint should lie exactly on the epipolar line. However, detection systems are not perfect, especially when an object is partially occluded in one or both images. There can also be inaccuracies in the camera calibration. Therefore, the height of the search window is large enough that small errors in the location of keypoints do not prevent matches. Both the width and height of the search window can be increased to account for these inaccuracies, but the larger the search window the higher the chances of an incorrect match and the more computationally intensive the algorithm becomes.

The algorithm will only match two keypoints if both the keypoints are contained by the other's search window and hence, meet the set geometry constraints. The algorithm consists of two stages. Stage one identifies unambiguous matches, while stage two evaluates and selects the best matching combination when there is ambiguity.

The following notation is used to describe the keypoints and search win-

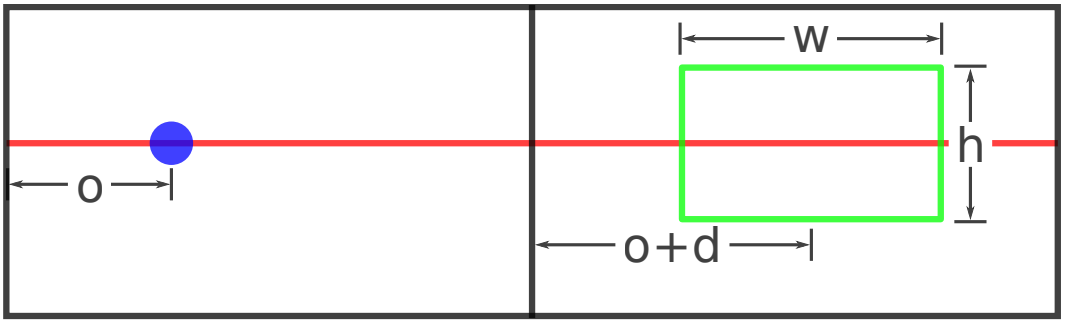


Figure 7.1: The geometry of a keypoint and search window. The two black rectangles represent the left and right images. The red line is an epipolar line running through the rectified and undistorted images. The blue dot is a keypoint in the left image. The green box is the search window in the right image where a keypoint matching the blue dot should be found.

dows; Keypoint `x` in the left image is denoted K_{Lx}, while keypoint `y` in the right image is K_{Ry}. The search window in the right image where a match to keypoint `x` (K_{Lx}) should be found is denoted W_{KLx}. Note, the numbering of keypoints in the left and right images are independent, hence, keypoint K_{Rx} is not necessarily a match to K_{Lx}.

Stage two of the algorithm judges matches based on their conformance to the mean fruit height based on all previously seen fruit (both in the current image pair, and in past image pairs). To do this, the mean of the position of each keypoint within its matching search window is logged (Figure 7.2). Two separate logs are maintained, one for all previously seen left images, and the other for all previously seen right images. These means are referred to as the 'mean position in window' henceforth.

7.1.1 Stage One

The first stage of the algorithm identifies all matches where a pair of keypoints from the left and right image are exclusively contained in the other's search window. For example, in Figure 7.3c-d, K_{L3} is the only keypoint contained in W_{KR2} and K_{R2} is the only keypoint contained in W_{KL3}. These matches are unambiguous as there is only one possible matching combination that meets

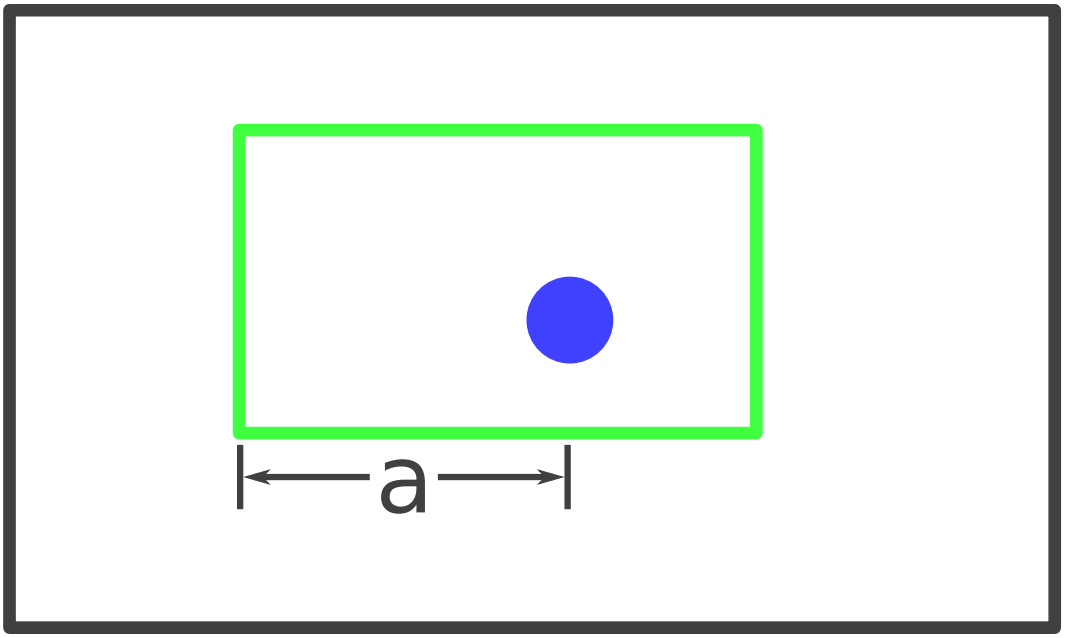


Figure 7.2: A keypoint (blue) in its matching search window (green). The distance, a, from the left edge of the search window to the keypoint is logged for both the right image and left image. The mean of all previously matched keypoints is calculated and is called the 'mean position in window'.

the geometry constraints. For each match found, the distance between the left edge of the search window and the keypoint contained within it is logged as part of the 'mean position in window'. Any search windows that contain no keypoints are discarded as there is no match that meets the geometry constraints. Search windows containing more than one keypoint are passed to stage two of the algorithm. The decision tree of stage one is outlined in Figure 7.4.

7.1.2 Stage Two

Stage two of the algorithm attempts to find matches for the keypoints remaining after stage one has been applied. Firstly, each keypoint is assigned to a group henceforth referred to as a 'group of ambiguity' (Figure 7.5). There can be any number of groups of ambiguity for an image pair. If a search window contains multiple keypoints, those keypoints and the keypoint corresponding to the search window are added to one group of ambiguity. This process is

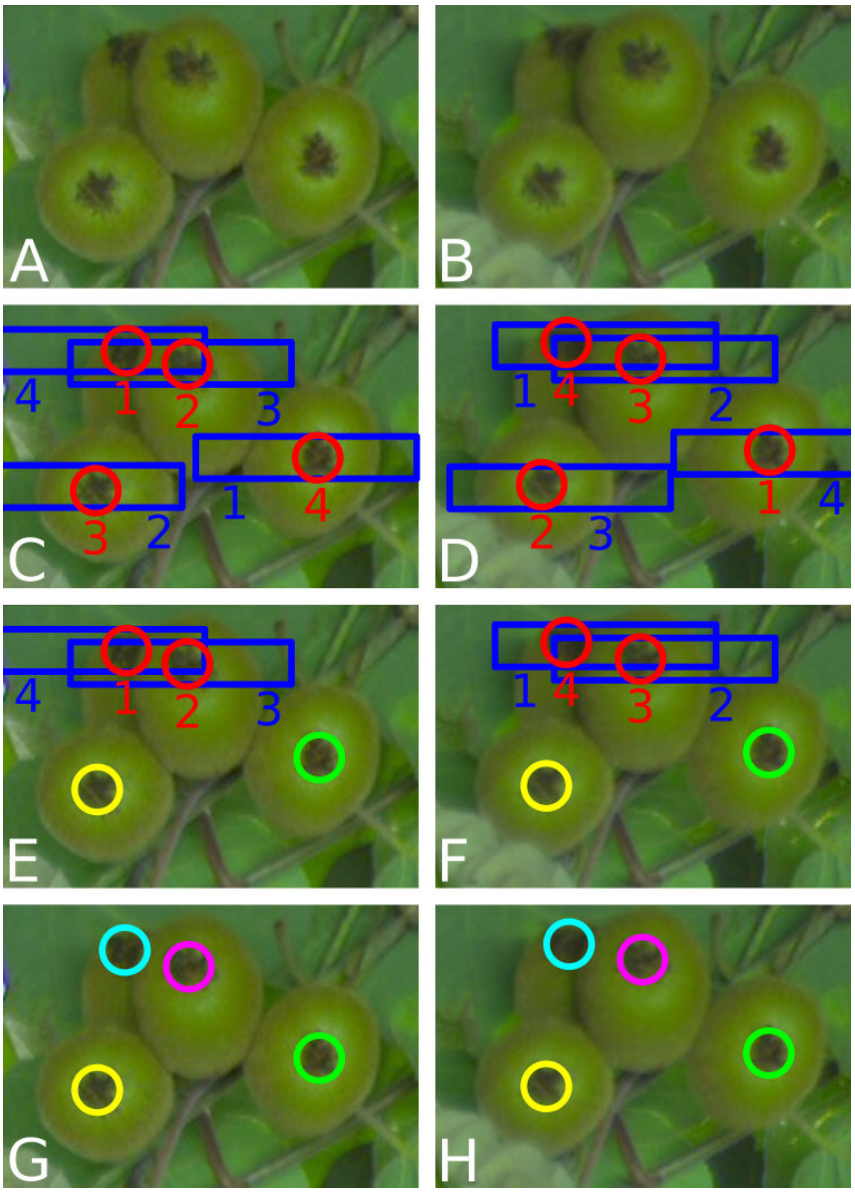


Figure 7.3: The steps of the matching algorithm. Panels A and B show a section of images from the left and right cameras. Panels C and D show the same images with the keypoints (calyxes) marked as red circles, and the corresponding search windows marked as blue rectangles. Notice the number on the calyx labels correspond to the numbers on the search windows in the opposite image. Panels E and F show the result after stage one has been applied, two of the fruit have been correctly matched (each colour represents a match). The remaining four keypoints (two in each image) form a group of ambiguity because their search windows contain multiple keypoints. Each potential matching combination is evaluated and the combination that most closely conforms to the mean height of all fruit previously matched is chosen. Panels G and H show the final result with the correct matches being found for all four fruit.

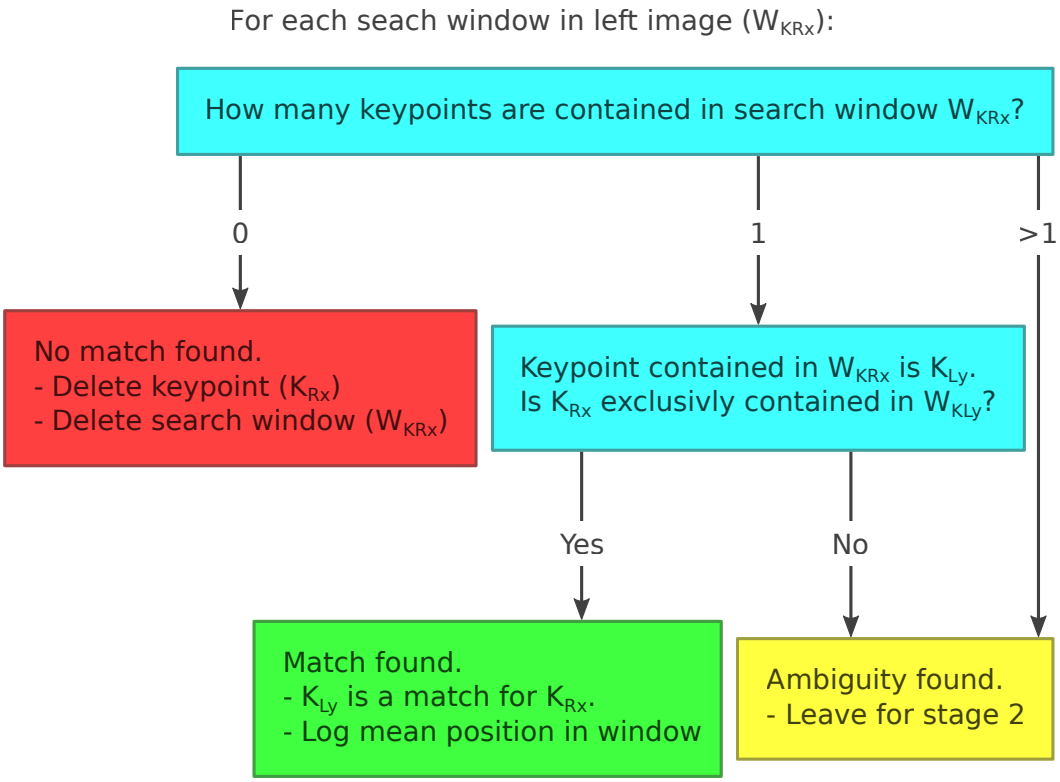


Figure 7.4: A decision tree describing stage one of the matching algorithm.

continued recursively until all remaining keypoints are assigned to a group of ambiguity. For example, in Figure 7.3e-f, keypoints K_{L1}, K_{L2}, K_{R3} and K_{R4} would form the only group of ambiguity.

For each group of ambiguity, the number of keypoints from each image is counted. If there is an imbalance of keypoints between images, 'dummy' keypoints are added to correct the mismatch. A 'dummy' keypoint can be matched with any keypoint from the other image, which represents no match being found for the keypoint.

Every possible matching combination of the keypoints is listed. A possible match is when two keypoints are contained in the other's search window. For example, the two possible matching combinations in Figure 7.3e-f are;

1. K_{L1} matched to K_{R3} and K_{L2} matched to K_{R4}

2. K_{L1} matched to K_{R4} and K_{L2} matched to K_{R3}

For each match in a potential matching combination, the distance between the keypoint and the left edge of the search window that the keypoint is being

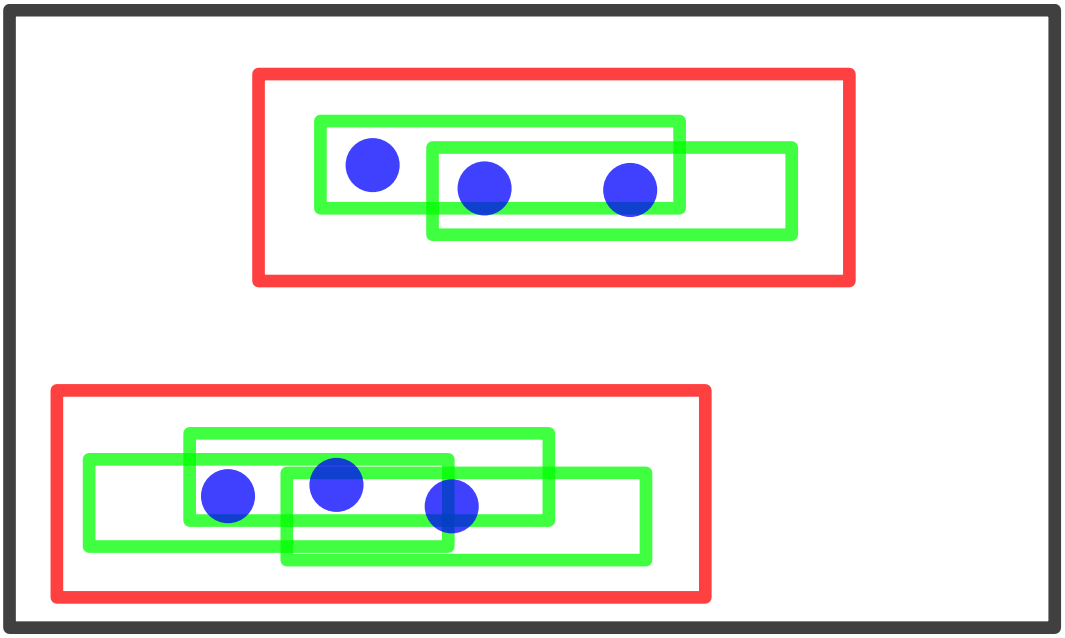


Figure 7.5: Two groups of ambiguity. The blue dots are keypoints in an image. The green rectangles are search windows. The red rectangles show how stage two would group the keypoints. Within each group, there is ambiguity as to which search window should be matched with each keypoint.

matched with is calculated and subtracted from the 'mean position in window'. The absolute value of each of these differences is summed for all of the matches in a potential matching combination to give a cost. For the example above (Figure 7.3e-f) the two costs equations are;

$$\begin{aligned} \text{cost}_1 = |M_L - a(K_{L1}, W_{KR3})| + |M_R - a(K_{R3}, W_{KL1})| \\ +|M_L - a(K_{L2}, W_{KR4})| + |M_R - a(K_{R4}, W_{KL2})| \end{aligned} \tag{7.1}$$

$$\begin{aligned} \text{cost}_2 = |M_L - a(K_{L1}, W_{KR4})| + |M_R - a(K_{R4}, W_{KL1})| \\ +|M_L - a(K_{L2}, W_{KR3})| + |M_R - a(K_{R3}, W_{KL2})| \end{aligned} \tag{7.2}$$

where M_L is the left image 'mean position in window', and $a(K_{L1}, W_{KR3})$ denotes the distance between K_{L1} and the left edge of W_{KR3}. The potential matching combination with the lowest cost is selected as the winner. For each match found, the distance between the left edge of the search window and the keypoint contained within it (`a`) is logged as part of the 'mean position

in window'. In the case of the example (Figure 7.3), combination 2 has the lowest cost and is selected.

After all matches have been selected, each fruit is triangulated using the OpenCV `triangulatePoints` function. Coordinates are then converted from homogeneous to Cartesian using the OpenCV `convertPointsFromHomogeneous` function. The result is an estimate of the location of each fruit relative to the camera pair. Using a transform and the trajectory output from the SLAM system, the estimated position of each matched fruit relative to the orchard block is calculated.

7.2 Setting Parameters

The valid range of fruit distances from the camera plane is set to 900–1700 mm based on the orchard dimensions and accounting for lower hanging fruit (Figure 3.3. OpenCV is used to calculate the mean disparity and search window width given the camera geometry. The mean disparity is 207 pixels and the search window width is 128 pixels (`d` and `w` in Figure 7.1 respectively). Search window height (`h` in Figure 7.1) is set by investigation to 20 pixels as this is large enough to account for most of the location inaccuracy observed.

7.3 Evaluation

The algorithm is run on 121 randomly selected image pairs from the fruit training dataset and the results manually evaluated. Matches are classified as 'correct' if a fruit in the left image is matched to the same fruit in the right image. Matches are classified as 'incorrect' if a fruit in the left image is matched to a different fruit in the right image. Matches are classified as 'false positive' if two false positive detections are matched to each other.

To measure computation time, the matching algorithm is run over 5000 random image pairs and the execution of the matching algorithm is timed. Tests are conducted on a Xeon E5-1650 V3 CPU (Intel, Santa Clara, USA)

	Occurrences	% of total matches
Image pairs	121	-
Total matches	3768	100 %
Correct matches	3739	99.23 %
Incorrect matches	6	0.16 %
False positive matches	23	0.61 %

Table 7.1: The results of the matching system evaluation. Over 99% of matches are correct.

Figure 7.6: Example of a false positive match. The match marked in green is not a fruit, but a dead leaf. The detection system has misclassified the leaf as a calyx in both images and the matching algorithm has matched it.

with 16 GB of memory.

7.4 Results

A total of 3768 matches are found (31.1 matches per image pair on average) across the 121 image pairs evaluated (Table 7.1). The correct match is found in 99.23% of cases and incorrect matches in 0.16%. False positive detections account for the other 0.61% of cases and are due to the detection system misclassifying areas of the image (Figure 7.6).

In five of the six cases of incorrect matches, a cluster of fruit are all detected in one image, but one fruit is missed in the other image (Figure 7.7). When fruit in a cluster that is at a height significantly different from the mean fruit height are missed, an incorrect matching combination can be selected.

Figure 7.7: Example of an incorrectly matched fruit. The match marked in grey is not the same fruit in both images. There are three fruit in a cluster with the middle fruit being partially occluded by the other two. All three of the fruit are detected in the right image, but the middle fruit is not detected in the left image. The missed fruit resulted in an incorrect match being found.

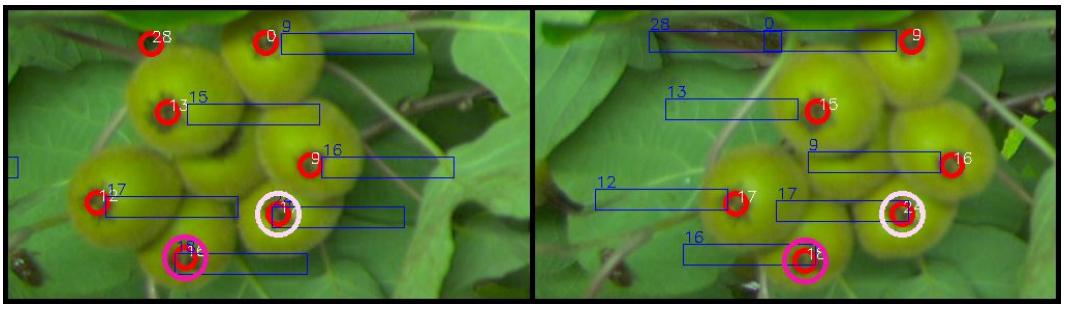


Figure 7.8: Example of fruit not being matched because they are too close to the camera plane. The keypoints fall outside the search window that they would be in if they are within the valid height range. Detected objects that lie outside the distance bounds will either not be matched or will be matched incorrectly.

Fruit that are outside of the set geometric bounds will not be matched. This is shown in Figure 7.8 where most of the fruit in a very low hanging cluster are not matched.

Williams et al. used a Hungarian method approach to match kiwifruit in stereo images for the purposes of harvesting [15]. Their imaging configuration is very similar to that used for this work. Identical cameras are used, with a similar baseline and camera to fruit distance. The authors report a 99.7% successful matching rate which is very similar to the 99.8% achieved by the system described above (ignoring false positive matches).

Computation time is 1.97 ms per image pair. This time could be im-

proved by evaluating matches in parallel, taking advantage of modern multi-core CPUs.

The algorithm described is best suited to quickly localising objects in environments that are largely structured but contain variation. Horticultural applications such as harvesting, spraying, weeding, counting and monitoring are a good fit. Other applications such as aerial surveying, certain manufacturing tasks and automated sports analysis could also be suitable.

Chapter 8

Multi-Frame Fruit Tracking

There is significant overlap between images (Figure 3.3). There is overlap between consecutive images (images are taken at approximately 200 mm increments), overlap between the two camera pairs and overlap between passes. Because of this overlap, each unoccluded fruit is seen in 3–10 image pairs. Therefore, if each individual detection is counted, many more fruit will be counted than are present in the orchard. There are two options to solve this double counting issue. The first is to account for double counting statistically, the second is a multi-frame fruit tracking system that can identify the same fruit across multiple image pairs.

Accounting for double counting statistically would require the overlap between images to be consistent. Both canopy height and row width vary between orchards, and between rows in some cases (Table 8.1). The variation results in the overlap between images being inconsistent, and hence, the rate of double counting will be inconsistent. Therefore, a multi-frame fruit tracking system is required. Herein tracking is identifying the same fruit across multiple images, each taken from a different viewing angle. This tracking prevents counting any given fruit more than once.

As the tracking system is required to be run over millions of images in a reasonable time frame (<2 days), processing time is of high importance. Therefore, methods that are highly computationally intensive are disregarded.

Block	Row Width (min-max)	Mean Fruit Height
S_01	4.4 - 4.6 m	**1.66** m
S_03	3.0 - 5.0 m	1.71 m
M_01	3.2 - 4.7 m	**1.77** m
K_D1	4.3 - **6.0** m	1.72 m
N_01	**2.9** - 3.1 m	1.73 m

Table 8.1: Differences between orchard blocks. The row widths and fruit height vary both between blocks as well as within blocks. This means overlap between images is inconsistent.

Using the stereo localisation and the SLAM systems (Chapters 4 and 7), an estimate of the position of each detected fruit within the orchard block is known. However there is error in each step of the fruit localisation process that needs to be corrected.

8.1 Fruit Localisation Error

There are three main sources of error in the fruit position estimates; stereo triangulation error, intra-pass SLAM error and inter-pass SLAM error. A demonstration of each type of error is shown in Figure 8.1. An effective multi-frame fruit tracking system is capable of tracking fruit despite these errors.

Stereo triangulation error is the smallest of the three errors (Figure 8.1A). It is in the order of 0–50 mm in scale. The causes of stereo triangulation error are; camera calibration error, detection location inaccuracies and partial occlusion of fruit. The error is independent for each detected fruit, that is the error is not consistent across an image pair.

Intra-pass SLAM error is positioning error from the SLAM system between camera frames (Figure 8.1B). It is larger than stereo triangulation error, in the order of 0–100 mm in scale. The main cause of intra-pass SLAM error is the pitch and roll of the ATV as this is not taken into account by the 2D SLAM system. Translational and yaw errors also contribute due to the SLAM system

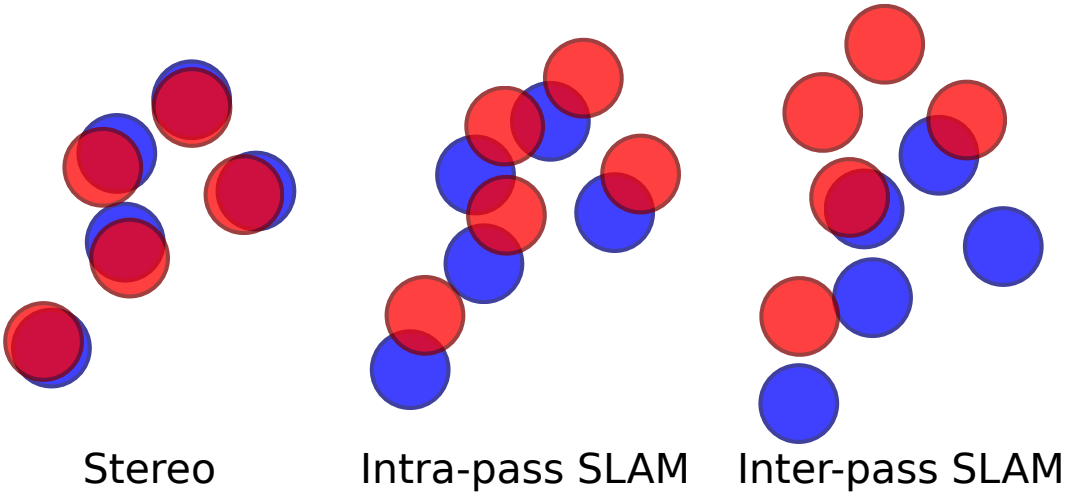


Figure 8.1: A demonstration of the three sources of fruit localisation error, simplified to 2D. In each of the three panels, the red and blue circles represent the same cluster of fruit, but seen in different image pairs.

being imperfect. The intra-pass SLAM error is consistent for all fruit in an image pair but varies between image pairs.

Inter-pass SLAM error is the largest of the three error sources (Figure 8.1C). It has been measured at up to 700 mm in scale (Figure 8.2). The cause is a build up or errors in the trajectory estimated by the SLAM system. The SLAM system does have a loop closure feature to reduce this type of error, but because of the challenging nature of lidar based SLAM in an orchard environment, it does not always totally negate it. The inter-pass SLAM error is consistent for all fruit in an image pair but varies between image pairs.

8.2 Closest Point

The simplest approach to a multi-frame fruit tracking system is a closest point algorithm. A closest point algorithm uses the fruit position estimate and assumes that any two fruit within a certain distance of each other and seen in different image pairs are the same fruit. This is the approach used by Wang et al. and Gongal et al., both of whom faced issues with accuracy due to errors in position estimates [19, 20, 28]. For this method to be effective, the positional errors must be less than the distances between the fruit. The distances between

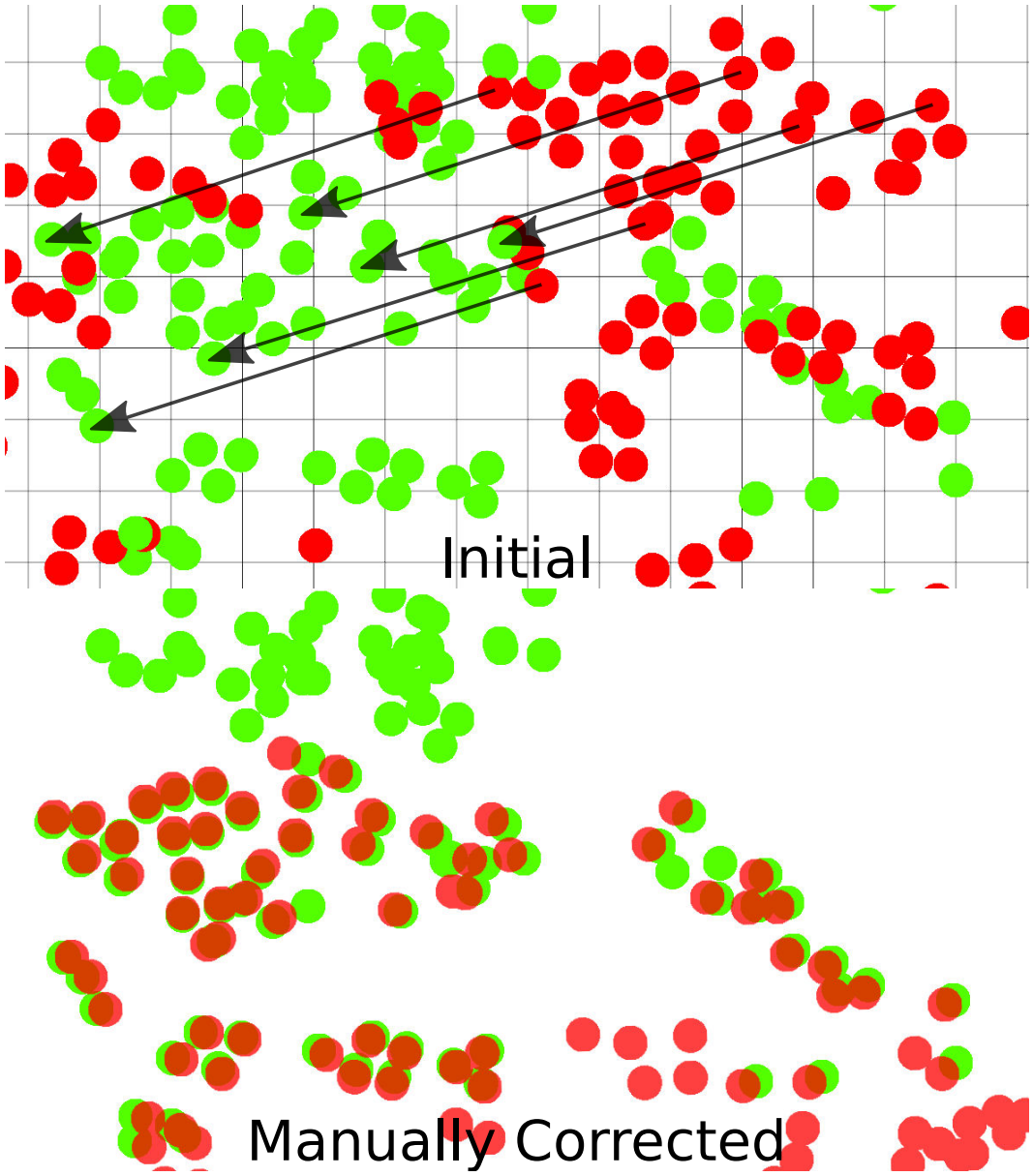


Figure 8.2: A real example of very large inter-pass SLAM error, viewed from above. Green fruit are from one pass down a row, red are from an adjacent pass. In the top panel, fruit are placed by the fruit localisation system (SLAM and stereo). The arrows show the correspondence of a subset of the fruit. Reference grid spacing is 100 mm. The lower panel shows the same view but with the red fruit transformed manually to correct for the SLAM error. The correspondence between the fruit is clear. The inter-pass SLAM error is approximately 655 mm in this case.

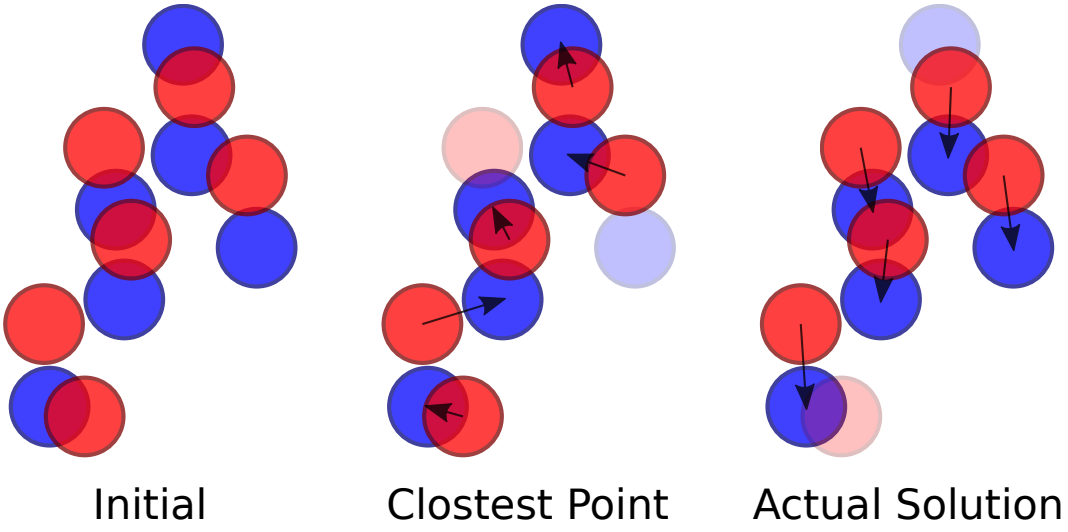


Figure 8.3: A demonstration of why the simplest approach (closest point) does not work with large errors. The red and blue circles represent the same cluster of fruit, but seen in different images pairs. The matches chosen by the closest point algorithm are incorrect. Because the main source of error is SLAM error, which is equal for all fruit in an image pair, all of the match vectors should point in approximately the same direction.

kiwifruit calyxes in a cluster is 50–80 mm, which is smaller than the intra-pass SLAM error in many cases, and significantly smaller than the inter-pass SLAM error. There are numerous examples in the data where a closest point aproach produces correct results. However for the vast majority of examples, the errors are simply too large for such a simplistic approach. Figure 8.3 demonstrates how the closest point approach fails when the positional errors are too large.

The stereo triangulation errors are smaller than the distances between fruit. Hence, the closest point algorithm is suitable for matching fruit if the stereo triangulation error is the only error source. As the SLAM errors are consistent across all fruit in an image, they can be corrected for using point cloud registration algorithms (Section 8.4). Once a transform correcting the SLAM error is calculated for each image pair, the closest point algorithm can match each fruit, completing the fruit tracking system.

8.3 Iterative Closest Point

The iterative closest point (ICP) algorithm is a point cloud registration technique [99]. A floating point cloud is moved using a 6-axis, rigid transform, to register it to a fixed point cloud. The transform is optimised by minimising a cost function, usually the sum of squared distances between each point in the floating cloud and the nearest neighbour in the fixed cloud.

The ICP algorithm is implemented and applied to fruit registration. The fruit in a new image pair form the floating point cloud to be transformed. All the previously seen fruit within a 3 m radius form the fixed point cloud. The `minimize` function of SciPy, with the Nelder-Mead solver is used to optimize the transform [118]. A KDTree is formed using the `neighbors.KDTree` class of scikit-learn to efficiently calculate the distance from each point to its nearest neighbour [119]. The cost function is a sum of squared distances with additional terms to penalise large transforms (Equation 8.1).

$$\text{cost} = a(x^2 + y^2 + z^2) + b(\phi^2 + \theta^2 + \psi^2) + c\sum_{i=0}^{n} D_{nn}(p_i)^2 \tag{8.1}$$

where a, b and c are constants, x, y and z are the translational components of the transform, ϕ, θ and ψ are the rotational components of the transform and $D_{nn}(p_i)$ is the distance from point p_i to its nearest neighbour in the fixed cloud. Once the optimiser has sufficiently optimised the transform, the closest point algorithm is used to form the final matches.

The ICP algorithm performs well in some cases. However, when the SLAM errors are larger than approximately 50 mm and/or there are new fruit seen or fruit missed in an existing cluster, errors are frequently made. One image pair that is incorrectly optimised, often leads to a series of images being incorrectly optimised as the errors compound.

The errors are caused by the optimiser becoming stuck in a local minima and/or the global minimum not being at the correct location. Figure 8.4 shows an example of a situation where the ICP system could make an error. Both of

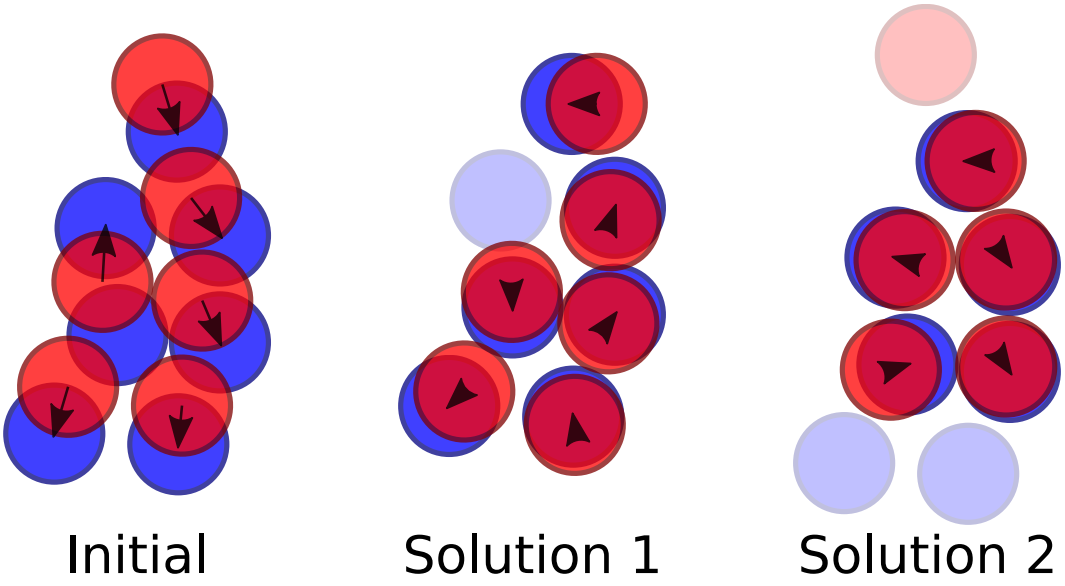


Figure 8.4: A demonstration of why the ICP algorithm fails in some cases. The initial locations of the fruit are shown with the closest point indicated by the arrows. There are two potential solutions the ICP algorithm could find, that both have a similar cost associated with them. It is ambiguous which is the correct solution, without more information.

the two potential transform and matching solutions have similar costs. Ideally, the cost function would have a clear global minima at the correct transform location, however in situations as illustrated, this is not the case. The cost function is often not a fair representation of the problem.

A thorough evaluation of the performance of the ICP algorithm is not conducted as the performance is clearly not adequate to be useful for the given data. Efforts are instead directed towards forming a more robust solution.

8.4 Kernel Correlation

The kernel correlation (KC) approach is similar to ICP. However, instead of only the closest fixed point to each floating point being used in the cost function, every fixed point is used (Figure 8.5) [120]. This change makes KC more robust to noisy data than ICP.

The KC algorithm is implemented with a Gaussian kernel (Figure 8.6). Using all of the points proved to be too computationally intensive, so only

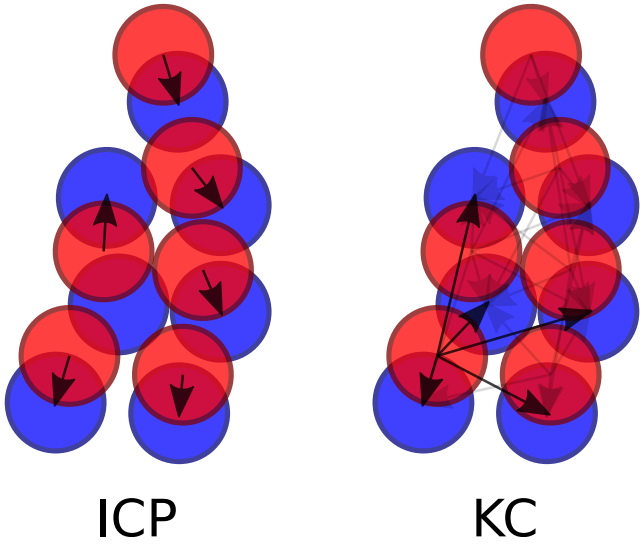


Figure 8.5: The difference between the ICP algorithm and the KC algorithm. The KC algorithm uses all fruit within a set radius rather than just the closest fruit. Note that the correlations for one fruit are highlighted for clarity in the KC diagram.

points within a radius of 200 mm are used. Otherwise, the implementation is identical to the ICP implementation.

The KC algorithm performs better than the ICP algorithm. In particular it is more robust in situations where there are missing fruit in either of the two point clouds. However, the performance is not high enough to provide reliable fruit tracking.

A thorough evaluation of the performance of the KC algorithm is not conducted as the performance is clearly not adequate to be useful for the given data.

Both the ICP and KC algorithms operate on only the point clouds. However, more information is available in the images. The KC algorithm could be augmented with information taken from the images to improve its reliability and performance.

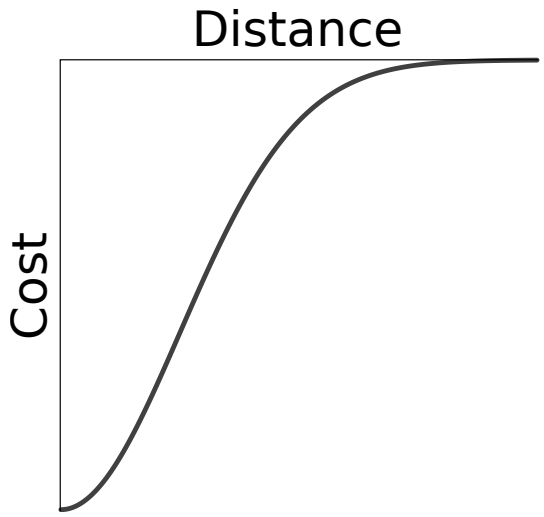


Figure 8.6: The negative Gaussian function used as the cost for each pair of points in the KC algorithm.

8.5 Calyx Comparison

Comparing the visual similarity of kiwifruit in different images can provide a score with which to weight the KC costs. Scarfe claims that kiwifruit blossom (calyx) ends 'appear to be as unique as a finger print' [13]. If that claim is true, accurately differentiating kiwifruit from each other based on their appearance should be possible.

The kernel correlation algorithm requires a mean of 12.0 comparisons between calyxes per detected fruit. The mean number of localised fruit per image is 30.9. So for every image, an average of 370.8 calyx comparisons will be made. As the algorithm is required to be run on millions of images in a reasonable time frame (<2 days), computation time for the calyx comparisons should be less than 100 ms per image.

A dataset of different views of the same calyxes is formed by manually identifying the same fruit in different image pairs (Figure 8.7). A total of 262 fruit are included, with a mean of 11.0 images per fruit. For each image pair that a fruit is identified in, two calyx images are obtained, one from the left image and one from the right. Each calyx image is 40×40 pixels, centred on the centre of the calyx bounding box.

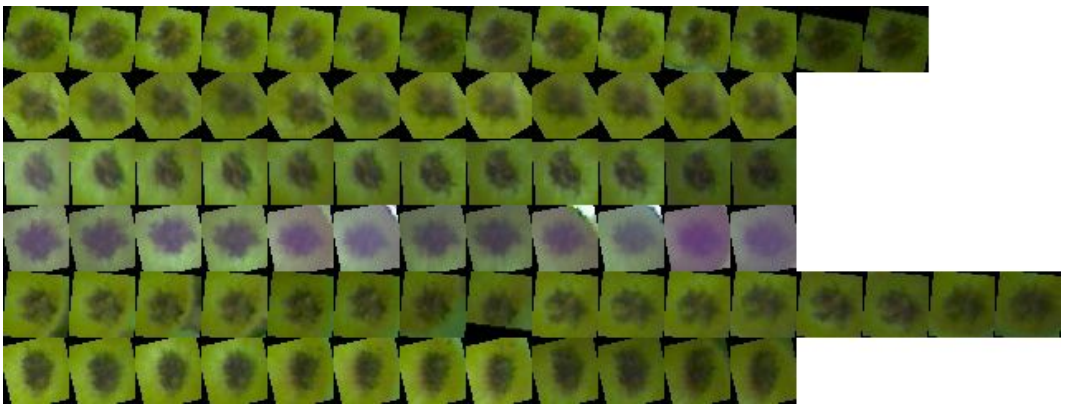


Figure 8.7: A subset of the calyx dataset. Each row is a different fruit. The images have been rotated to correct for the rotation of the ATV at the time of imaging.

The rotation of the ATV is not necessarily the same for each pass down a row, particularly at the start and end of rows. Therefore, a rotation is applied to each calyx image to correct for the rotation of the ATV at the time of imaging, as estimated by the SLAM system. Applying this rotation means the calyx comparison system is not required to be robust to varying rotation and should thus be quicker and more accurate.

8.5.1 Facial Recognition for Kiwifruit

Modern facial recognition algorithms, such as FaceNet, can be trained/tuned to discriminate between human faces with up to 99.6% accuracy [121]. They can achieve this high performance despite the images being taken from a variety of viewpoints. FaceNet encodes each face it's shown as a 128-dimensional vector. Each of the 128 dimensions describes a different property of the face. These properties are learned via the training process. To train, the network is fed triplets of images (images A, B and C). Image A and B are different images of the same person, while image C is of a different person. The three images are passed through the network to generate a 128-dimension vector describing each. The network will then be adjusted to make the vectors from images A and B closer to each other (measured as the Euclidean distance between the two vectors) and those from images A and C further apart. In this way,

Embedding size	128
Weight decay	0.0001
Learning rate decay epochs	100
Learning rate	0.01
Fruit per batch	33
Epoch size	1000
Optimiser	RMSPROP
Batch size	90
Random flip	False
Random crop	False
Number of epochs	500

Table 8.2: Training parameters used for FaceNet.

once training is complete, FaceNet is able to distinguish between two faces that it has never before seen. If an algorithm can be tuned to encode the subtle differences between human faces, it can also be tuned to do the same for kiwifruit calyxes.

8.5.1.1 Training

The 262 fruit calyx dataset is split randomly into a training set of 220 fruit and a validation set of 42 fruit. The training set contains a total of 2470 individual fruit images, of 220 distinct fruit, as there are multiple images of each fruit (from different viewpoints). An open-source, TensorFlow based implementation of FaceNet is trained[1]. Images are scaled up from 40×40 pixels to 160×160 to suit the architecture of the network. The model is trained from random initial values. The parameters used for training are shown in Table 8.2.

The authors of FaceNet observed significant improvements in performance when increasing from 2.6 million training images to 26 million (76.3% to 85.1%

[1] https://github.com/davidsandberg/facenet

accuracy) [121]. Further improvements are seen when using up to 260 million images (86.2% accuracy), although with diminishing returns. Because of the time consuming nature of creating the calyx dataset, only 2470 images are included in the training set. Therefore, high accuracy is not expected.

8.5.1.2 Evaluation

The real world use case for comparing calyx images is to compare a fruit seen in a new image pair, to a fruit that has been seen in a random number of previous image pairs, and may or may not be the same fruit. To mimic this use case, the following procedure (outlined in Figure 8.8) is used to select images to compare from the validation dataset of 42 fruit (420 total calyx images).

From the validation dataset two fruit are selected at random. From the first fruit, one calyx image pair is selected at random. This pair of calyx images is referred to as the 'fixed set'. From the remaining calyx image pairs of the first fruit, a random number of calyx image pairs are selected. This set of calyx image pairs are referred to as the 'same set'. From the second fruit, a random number of calyx image pairs are selected. This third set of calyx image pairs is referred to as the 'different set'. To summarise; there are three sets of calyx image pairs. The 'fixed set' contains a single pair of calyx images. The 'same set' contains a random number of calyx image pairs, of the same fruit as in the fixed set. The 'different set' also contains a random number of calyx image pairs, but of a different fruit.

Each image in the three sets (fixed, same and different sets) is run through the trained FaceNet model to obtain the vector describing each image. The Euclidean distance between each vector from the 'fixed set' and each vector from the 'same set' is calculated (Figure 8.9). The minimum of these distances is taken as the 'same score'. The Euclidean distance between each vector from the 'fixed set' and each vector from the 'different set' is calculated. The minimum of these distances is taken as 'different score'. The two scores are both logged and the full procedure (selecting new images and running them

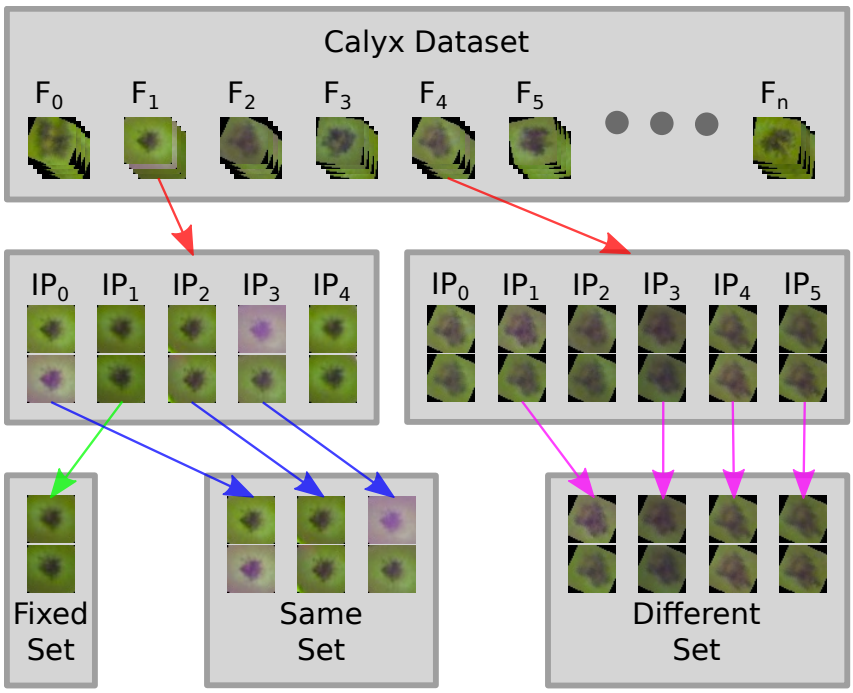


Figure 8.8: The evaluation image selection process. From the calyx dataset, two fruit are selected at random (red arrows). From the image pairs of the first fruit, one pair of images is randomly selected, forming the 'fixed set' (green arrow). From the remaining images pairs, a random selection is taken, forming the 'same set' (blue arrows). From the image pairs of the second fruit, a random selection is taken, forming the 'different set' (purple arrows).

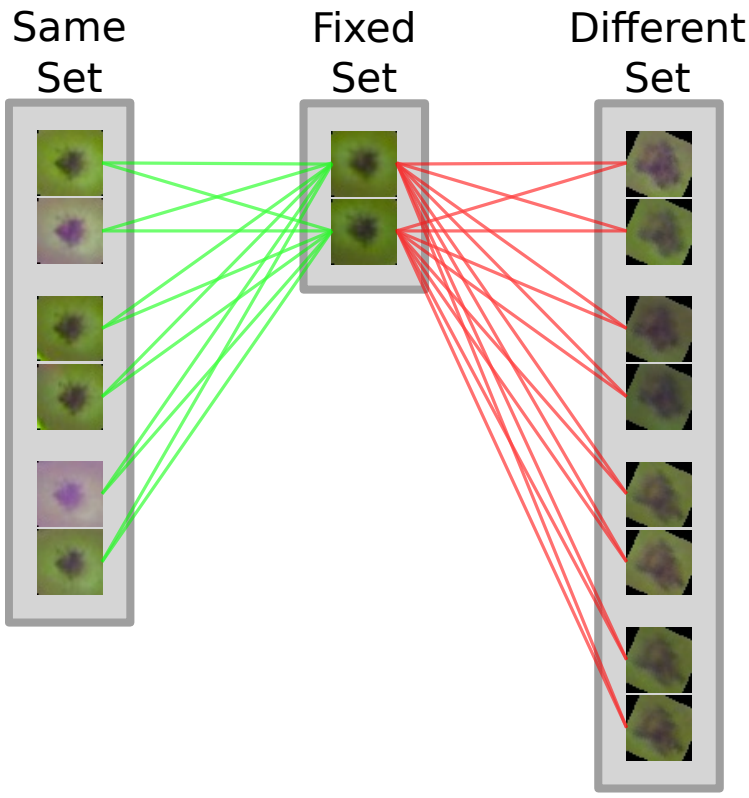


Figure 8.9: Each image in the 'fixed set' is compared to each in the 'same set' (green) by measuring the Euclidean distance between the generated vectors. The minimum distance is logged as the 'same score'. Each image in the 'fixed set' is compared to the each in the 'different set' (red) by measuring the Euclidean distance between the generated vectors. The minimum distance is logged as the 'different score'.

through FaceNet) is repeated 50,000 times.

8.5.1.3 Results

The distribution of scores is shown in Figure 8.10. If a threshold is placed such that an equal number of false positives and false negatives are produced, the error rate is 8.74%. This is very high performance given the low number of images in the training set. The distribution of scores for 'different fruit' is approximately normal, however the distribution of scores for 'same fruit' has many spikes. These spikes are due to the low number of fruit in the validation set (42), which causes scores to cluster.

The FaceNet system takes an average of 2.8 ms to encode a pair of calyx images as vectors when using the computer outlined in Table 6.4. The encod-

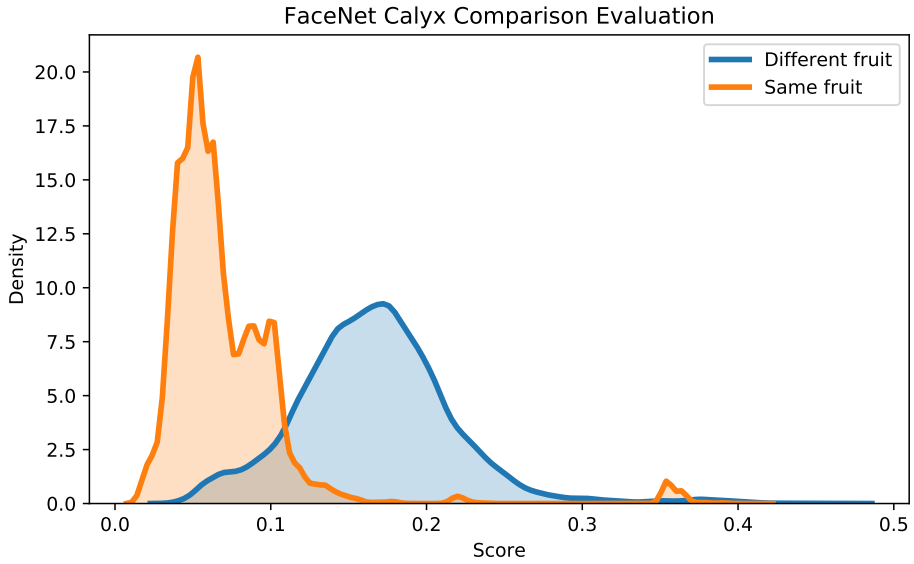


Figure 8.10: The performance of the FaceNet based calyx comparison system on the validation calyx dataset. The scores for the 'same fruit' comparisons are lower, on average, than those for the 'different fruit'. This suggests that the system is able to identify the same calyxes in different images.

ing is only performed once for each calyx image pair. For a full image pair containing the mean number of fruit (30.9), 86.5 ms are required to calculate all vectors. The timing is conducted in a single threaded test using only one of the GPUs available in the system. Comparison of two image pairs (calculating distance between each pair of vectors) takes 33 µs, using the same computer. Comparison timing is also conducted in a single threaded configuration, using only one of the CPU cores available.

Investigation into the cause of the high scores for 'same fruit' (>0.2) shows the cause to be fruit near the edge of the images. If the 40×40 pixel crop of a calyx extends beyond the edge of the image it is taken from, black pixels are inserted to ensure the full 40×40 pixel image size, as seen in Figure 8.11. The high scores for images near edges suggests FaceNet is evaluating fruit images based on the distribution of black pixels in the images. Black pixels are due to both fruit near the edges of images and from the rotation applied to images to account for ATV rotation. In creating the calyx image dataset,

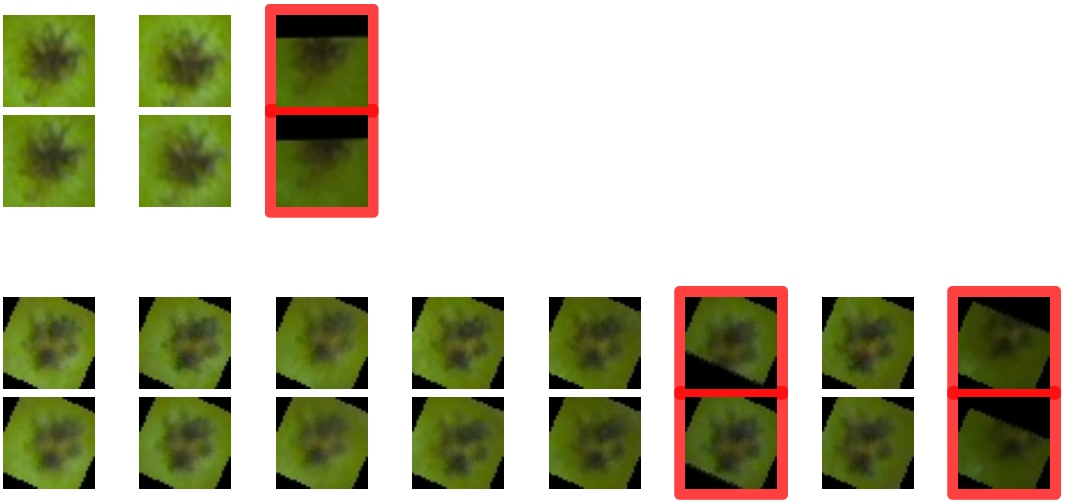


Figure 8.11: An example of images of two fruit in the calyx validation dataset where the edge of images is seen. The calyx images with red borders have been taken from the edge of images and hence have a different arrangement of black pixels to the other images of the same fruit. The difference causes the FaceNet based calyx comparison system to misidentify these fruit.

fruit are randomly selected from all of the orchard blocks in the fruit training set. The 40×40 pixel crop of each calyx is taken and a rotation is applied to account for the rotation of the ATV at the time of image capture. Therefore, in the calyx dataset, the rotation (and hence the distribution of black pixels) is likely to be different between any two randomly selected fruit, but very similar between images of the same fruit (see Figure 8.7). In a real world situation, the rotation applied to fruit that are near each other will be very similar as the ATV is always facing down the direction of the row. The only exceptions are at the start and end of rows and in rare occurrences of obstacles in orchards. Therefore, a system evaluating calyx images based solely on the black pixels would perform well when applied to the calyx dataset but very poorly in a real world situation. It appears that the FaceNet based system is using the black pixels as its method for differentiation. This is a flaw in the design of the experimental set-up. Further evaluation of FaceNet with this flaw fixed is not performed due to time limitations.

8.5.2 Direct Calyx Comparison

For stereo matching purposes, Scarfe used a normalised cross-correlation template matching system [13]. A 16×16 pixel, normalised grey-scale image of a calyx from one image is translated around an area of the other image of the stereo pair to find the locational of best fit. The fit is evaluated using a sum of squared differences between each pixel of the two images. Scarfe claims this algorithm performed very well for stereo matching purposes, but it is very slow, taking 14–26 seconds per image pair. A similar, cross-correlation template matching approach could be applied to inter-frame calyx comparison.

Four versions of a direct calyx comparison system are developed and evaluated. The first compares the raw calyx images, pixel for pixel. The second compares a grey-scale, normalised version of the calyx images. The third compares a grey-scale mean compensated version of the calyx images. The fourth version uses a binary representation of the calyx images for comparison.

8.5.2.1 Raw Image Comparison

A pixel-wise, sum of squared differences system is implemented to compare two calyx images. Comparison is conducted on each of the red, green and blue channels of each pixel. To prevent the black pixels, caused by the image rotations, from biasing results, black pixels are disregarded. The sum of squared differences, divided by the number of non black pixels, is used as the final difference score. A high difference score suggests the images are from different fruit. The process is shown in Figure 8.12.

Evaluation of the raw colour image comparison algorithm is conducted using the same method as for the FaceNet system (Section 8.5.1.2), but with the full calyx dataset used rather than just the validation dataset. Generation of each calyx image, which in this case is simply applying a rotation, takes 34.6 µs, with the comparison step taking 179 µs (Table 8.3). The error rate is 24.7%.

A second version of the raw image calyx comparison system using grey-

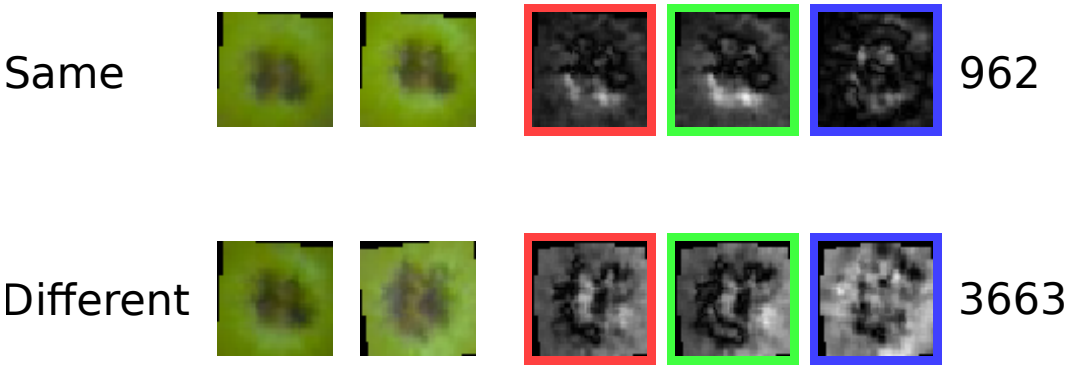


Figure 8.12: Two examples of the 40 pixel colour calyx comparison process. The top row shows two images of the same fruit being compared. The two calyx images to be compared are on the left, followed by the differences in the two images in the red, green and blue colour channels. The brightness of each pixel in the difference images represents the difference in the two calyx images with bright being a large difference. The final difference score is also shown. The bottom row shows the same process for two images of different fruit.

Method	**Generate**	**Compare**	**Error Rate**
40px, colour	**34.6 µs**	179 µs	**24.7%**
40px, grey-scale	37.9 µs	**131 µs**	30.3%

Table 8.3: The performance of the two variants of the raw image calyx comparison system.

scale images as opposed to colour images is also implemented. The process is outlined in Figure 8.13. The generation step is slightly slower at 37.9 µs due to the need for a colour conversion as well as a rotation, but the comparison step is faster, at 131 µs. The error rate is higher than the colour system at 30.3%.

Analysis of the errors from both the colour and grey-scale version of the raw image comparison system show the main source of error is different overall brightness levels between images of the same calyx. An example of brightness difference can be seen in Figure 8.14.

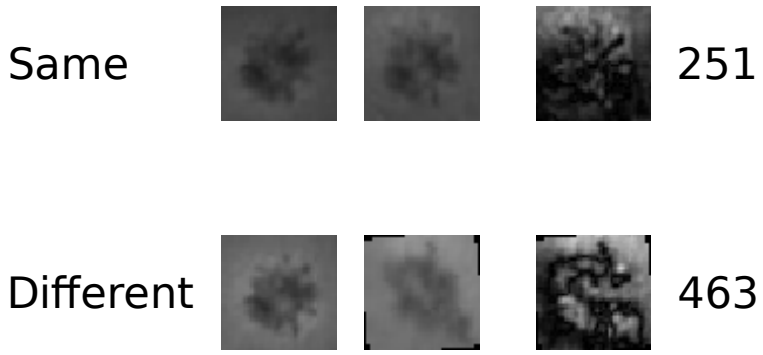


Figure 8.13: Two examples of the 40 pixel grey-scale raw calyx comparison process. The top row shows two calyx images of the same fruit being compared, followed by the differences in the two images. White pixels represent large differences and black pixels, small differences. Note the differences have been exaggerated to increase visibility. The final score is also shown. The bottom row shows the same process for two images of different fruit.

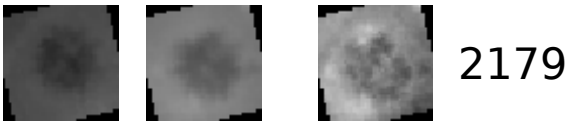


Figure 8.14: An example of two images of the same calyx with different brightness levels. The first two images are the two calyx images, the third image is the difference image. The brightness difference causes a high difference score.

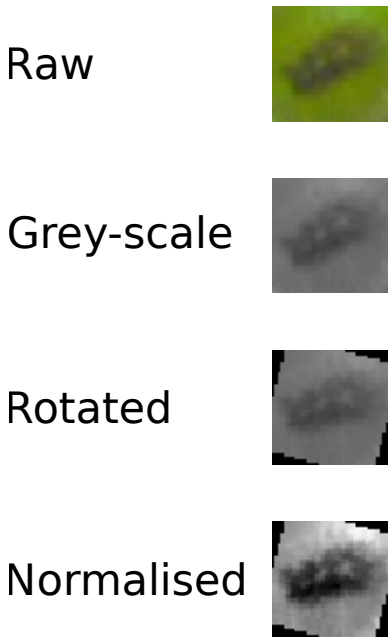


Figure 8.15: The steps used to create the normalised calyx images. The image is first converted to grey-scale. Then a rotation is applied to correct for the rotation of the ATV. The image is then normalised, ignoring the already black pixels.

8.5.2.2 Range Normalisation

To overcome the brightness difference issues of the raw image comparison system, a system is implemented using a normalisation step. Grey-scale images are normalised so the darkest pixel has a value of 0 and the brightest, 255. The pixels that are already black due to the rotation applied to the image and fruit near the edges of images are ignored in the normalisation step. An example of the normalisation is shown in Figure 8.15.

Using normalised grey-scale images, the error rate is 13.1% (Table 8.4). The generation time is 133 µs while the comparison time is 131 µs.

Analysis of the errors highlights two issues. The first issue is sky being visible beside the fruit in some images, but not others. The sky is much brighter than the fruit and causes the rest of the image to be very dark when normalised. The effect of sky can be seen in Figure 8.16. The second cause of errors is calyxes not being properly centred in the images. If two images of the same calyx are misaligned, the difference score will be large, causing errors, as

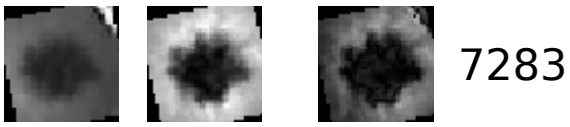


Figure 8.16: An example of sky pixels in a calyx image. The sky pixels are seen in the top right of the first image, which cause the fruit to be very dark after normalisation. The large difference in overall brightness of the two calyx images means there is a high difference score.

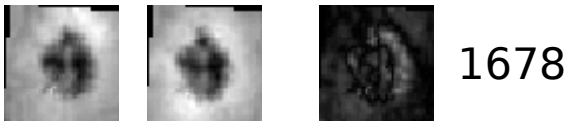


Figure 8.17: An example of the calyx not being properly centred in both images. The first image has the calyx to the right of the centre of the image whereas the second image has the calyx towards the left. This causes a high difference score.

seen in Figure 8.17.

To overcome these issues, two modifications are applied to the system. The first modification is to disregard very bright pixels in the normalisation step in the same way that black pixels are disregarded. The second modification is to repeat the comparison multiple times with one of the calyx images translated to different positions (Figure 8.18). A difference score is calculated at each position and the lowest of the scores is used as the final difference score. Five positions are tested for each calyx comparison; one with no translation and one each with translations of one pixel up, down, left and right. A version using nine positions is also tested, which adds positions two pixels up, down, left and right.

Adding white pixel rejection deceases the error rate to 12.7% (Table 8.4). Both generation and comparison time are unchanged. Using both white pixel rejection and the five position translation system decreases the error rate to 11.5%. As the translation system is repeating the comparison step multiple times, with computational overhead in between each comparison, comparison time increases significantly to 966 µs. The nine position translation system

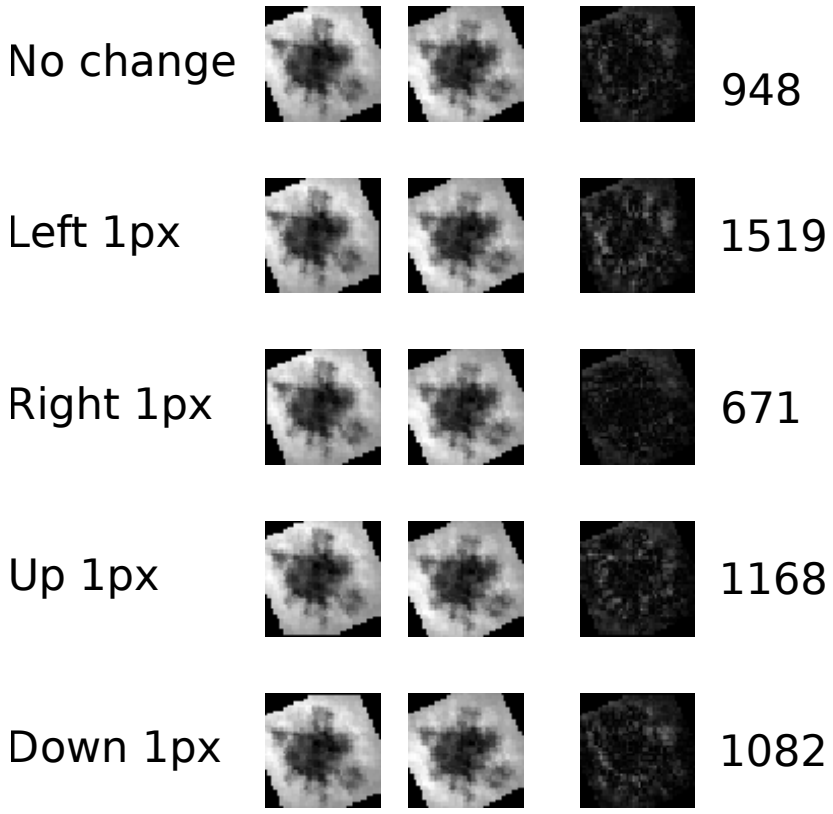


Figure 8.18: An overview of the translation system. Each row shows one of the tested configurations, with the two calyx images, followed by the difference image and the difference score. The first row has both calyx images in their usual form. The other four rows show the first calyx image translated one pixel left, right, up and down respectively, which results in a a set of difference scores. The lowest difference score is taken as the final score, which in this case is obtained when the first calyx image is translated one pixel to the right.

Method	Generate	Compare	Error Rate
40px	133 µs	131 µs	13.1%
40px, no white	133 µs	131 µs	12.7%
40px, no white, 5-position	133 µs	966 µs	12.1%
40px, no white, 9-position	133 µs	1789 µs	11.5%
16px, no white	**92 µs**	**86 µs**	6.9%
16px, no white, 5-position	**92 µs**	640 µs	**5.1%**

Table 8.4: The performance of the variants of the range normalisation calyx comparison system.

reduce the error rate to 11.5% and increases comparison time to 1789 µs.

Further, analysis of the errors shows higher pixel differences around the edges of the images as opposed to in the centres. A version of the range compensation system is implemented using a crop taken from the centre of the calyx image. The optimal crop size is found to be 16×16 pixels by investigation. Using 16×16 pixel images reduces the error rate to 6.9%, while both generation time and comparison time are also decreased (Table 8.4). Adding the five position translation system used with the range normalisation system, further reduces error rate to 5.1%, however comparison time increases to 640 µs.

8.5.2.3 Mean Compensation

Range normalisation expands the range of pixel brightness in an image to match the maximum allowable range. This range expansion has more of an effect on images that have a small dynamic range than those with a large dynamic range. In contrast, mean compensation does not effect the dynamic range of images, it adjusts the mean brightness of each image so they are all the same.

A mean compensation system is implemented. Once an image has been converted to grey-scale and the ATV rotation applied, the mean pixel value is

Method	**Generate**	**Compare**	**Error Rate**
40px, no white	120 µs	93 µs	15.2%
16px, no white	**94 µs**	**88 µs**	9.5%
16px, no white, 5-position	**94 µs**	692 µs	**6.7%**

Table 8.5: The performance of the variants of the mean compensation calyx comparison system.

calculated. The mean value calculation ignores pixels that are already black or white. The mean is then subtracted from each pixel.

Evaluation shows the mean compensation calyx comparison system achieves a 15.2% error rate when using 40×40 pixel images. Generation time is 120 µs, and comparison time is 93 µs (Table 8.5).

A version of the mean compensation system is implemented using a 16×16 crop of the calyx image. Using the cropped calyx images, the error rate is 9.5%, while both generation time and comparison time are decreased (Table 8.5). Adding the five position translation system further reduces error rate to 6.7%, however comparison time increases to 692 µs.

The LED lighting used on the data capture system causes a visible bright spot on each fruit. The position of the bright spot depends on the orientation of the fruit and the position of the fruit within the image. The bright spot will tend to be below the calyx on fruit near the top of the image and vice versa. Fruit near the centre of the image tend to have the bright spot surrounding the calyx as can be seen in Figure 8.19. The differences in position of the bright spot can cause high difference scores when comparing images of the same fruit as can be seen in Figure 8.20. The bright spot issue effects the raw image comparison system and both the range normalised and mean compensation systems.

Figure 8.19: A full image showing the how the bright spot on each fruit changes across the image. The LED lighting on the data capture system causes a bright spot on each fruit. The position of the bright spot on the fruit depends on both the orientation of the fruit and its position within the image. Fruit near the left of the image will have the bright spot on the right of the calyx and vice versa.

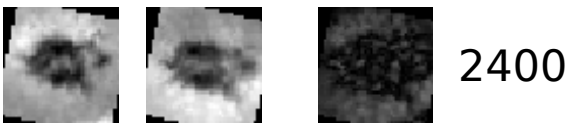


Figure 8.20: An example of bright spot mismatch between two images of the same calyx. The bright spot is above the calyx in the first image and below it in the second. The result is a high difference score despite the two images being of the same calyx.

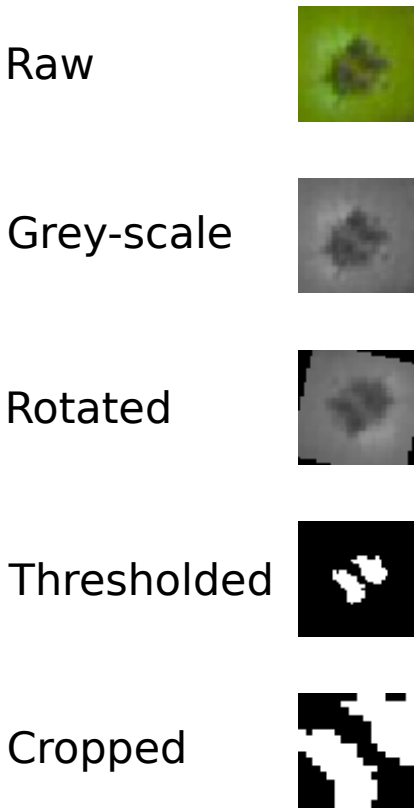


Figure 8.21: The binary calyx image formation process. The raw image is first converted to grey-scale, before being rotated to correct for ATV rotation. A threshold is then applied, with the darkest 9% of pixels not already black identified. The image is then cropped to 16×16 pixels (testing is also conducted without this cropping step).

8.5.2.4 Binary Image Comparison

To reduce the effect of bright spots on calyx comparisons, a binary image comparison system is implemented. Rather than comparing the difference in brightness for each pixel of the image, only the location of the darkest pixels is compared. To create the binary images, the 40×40 pixel calyx images are converted to grey-scale and rotated to correct for ATV rotation. A binary threshold is then applied to the image to form the binary image. The threshold is chosen for each image so that the darkest 9% of pixels (excluding already black pixels) are below the threshold. The 9% threshold is found to produce the lowest error rates. The comparison step is then a pixel-wise exclusive or (XOR) operation followed by a sum to count the number of pixels that are different between two binary images. The process is outlined in Figure 8.21.

Method	Generate	Compare	Error Rate
40px	**237 µs**	**15 µs**	10.9%
16px	**237 µs**	**15 µs**	8.0%
16px, 5-position	**237 µs**	122 µs	**6.4%**

Table 8.6: The performance the variants of the binary image calyx comparison system.

The error rate of the 40×40 pixel binary image system is 10.9% (Table 8.6). Generation time is significantly higher than the other methods at 237 µs due to the added complexity of the thresholding step. However, comparison time is very low at 15 µs. Note that the comparison time is the same for both the 40×40 pixel and the 16×16 pixel versions. This is because the XOR operation is fast enough that it makes up a small percentage of the total comparison time.

A version of the binary image system is tested using a cropped region of the binary image. It is found that 16×16 pixels provides the highest accuracy with an error rate of 8.0% (Table 8.6). Increasing the number of positions tested to five, using the five position translation system described above, further decreased the error rate to 6.4%. However, the additional computation increases comparison time to 122 µs.

Other methods and modifications are tested but not fully evaluated. These include, centring the binary images at generation time to avoid the need for the translation system. This centring is achieved by balancing the number of white pixels in each of the four quadrants of the image. However this centring approach increases both the error rate and generation time. Also tested is modifying the binary image approach by applying a blur to the image before the threshold step, which slightly increases the error rate and generation time. Weighting the difference in pixels nearer the centre of the images higher than the outer pixels using a Gaussian weighting function, is also tested. This weighting increases comparison time and slightly increases the error rate.

8.5.3 Summary

The performance of all the variants of the calyx comparison systems is shown in Table 8.7. When using the kernel correlation system for multi-frame fruit tracking, the mean number of calyx comparisons executed is 12.0. The mean number of fruit per image is 30.9. The total time to conduct calyx comparisons for an average image is calculated using the following equation:

$$t = 30.9(g + 12c) \tag{8.2}$$

where t is the total image time, g is the generation time and c is the comparison time.

As the algorithm is to be applied to millions of images, low processing time is a priority. Therefore, only systems with a total image processing time less than 100 ms are considered for use in the final fruit yield estimation system. Of the algorithms that meet this time requirement, the algorithm with the lowest error rate is the 16×16 pixel binary image, five position method (henceforth referred to as 'the calyx comparison system'). Therefore, this method is selected for use in the final fruit yield estimation system. The distribution of difference scores produced by the the calyx comparison system is shown in Figure 8.22.

8.6 Intra-Pass Fruit Tracking System

The intra-pass fruit tracking system consists of three sections:

Kernel correlation with calyx comparison to correct for the intra-pass SLAM error.

Match selection to identify correspondence between fruit in new images and previously seen fruit.

Final fruit matching to identify the remaining fruit to be matched.

Method	Total Image Time	Error Rate
Raw		
40px, colour	**67.4 ms**	24.7%
40px, grey-scale	**49.7 ms**	30.3%
Range Normalisation		
40px	**52.7 ms**	13.1%
40px, no white	**52.7 ms**	12.7%
40px, no white, 5-position	362.3 ms	12.1%
40px, no white, 9-position	667.5 ms	11.5%
16px, no white	**34.7 ms**	6.9%
16px, no white, 5-position	240.2 ms	**5.1%**
Mean Compensation		
40px, no white	**38.2 ms**	15.2%
16px, no white	**35.5 ms**	9.5%
16px, no white, 5-position	259.5 ms	6.7%
Binary Image		
40px	**12.9 ms**	10.9%
16px	**12.9 ms**	8.0%
16px, 5-position	**52.6 ms**	6.4%
Other		
FaceNet	**98.8 ms**	8.7%

Table 8.7: The performance of each of the calyx comparison systems evaluated. The total image time is the time to perform all the generation and comparison steps required to process an average image. All total image times below 100 ms are shown in bold as they meet the computation time requirement. The FaceNet system is included for reference, however the error rate reported should not be trusted due to the issues discussed in Section 8.5.1.3.

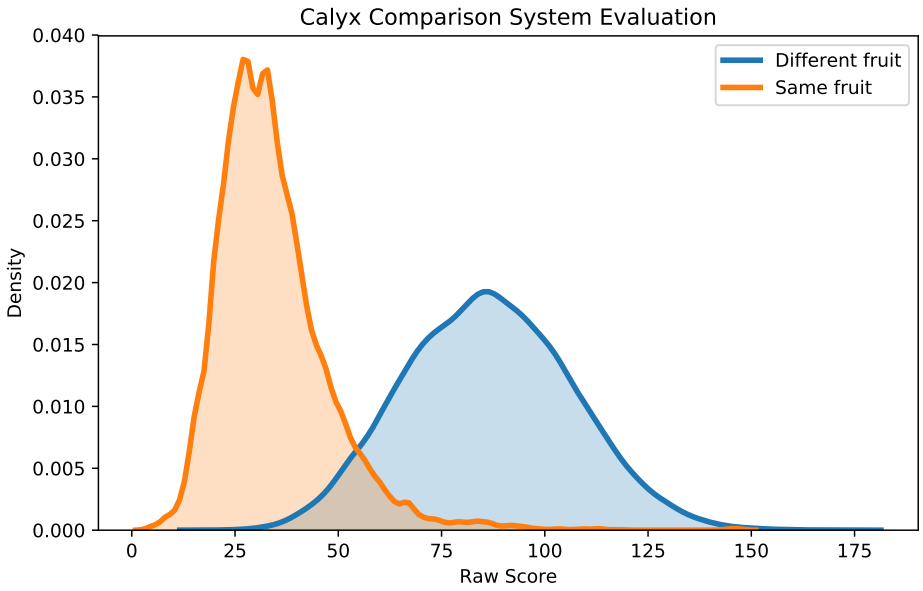


Figure 8.22: The distribution of difference scores produced by the calyx comparison system.

All three of the sections are implemented using a single threaded architecture. As the system is applied on a per pass basis, multiple passes can be processed in parallel on a multi-core CPU to significantly lower overall computation time.

8.6.1 Kernel Correlation with Calyx Comparison

The kernel correlation with calyx comparison system (KC+CC) is a modified version of the kernel correlation system described in Section 8.4. The cost function is weighted by the score from the calyx comparison system (Section 8.5) rather than being solely a function of distance between two points. Figure 8.23 demonstrates the differences between the ICP system, the KC system and the KC+CC systems.

The detected fruit in a new image make up a point cloud referred to henceforth as the 'floating cloud'. All fruit detected in previous images taken within 1.5 m of the new image make up a point cloud referred to henceforth as the 'fixed cloud'.

Fruit that are detected near the edge of an image are often not accurately localised by the stereo system as the detected calyx positions are influenced

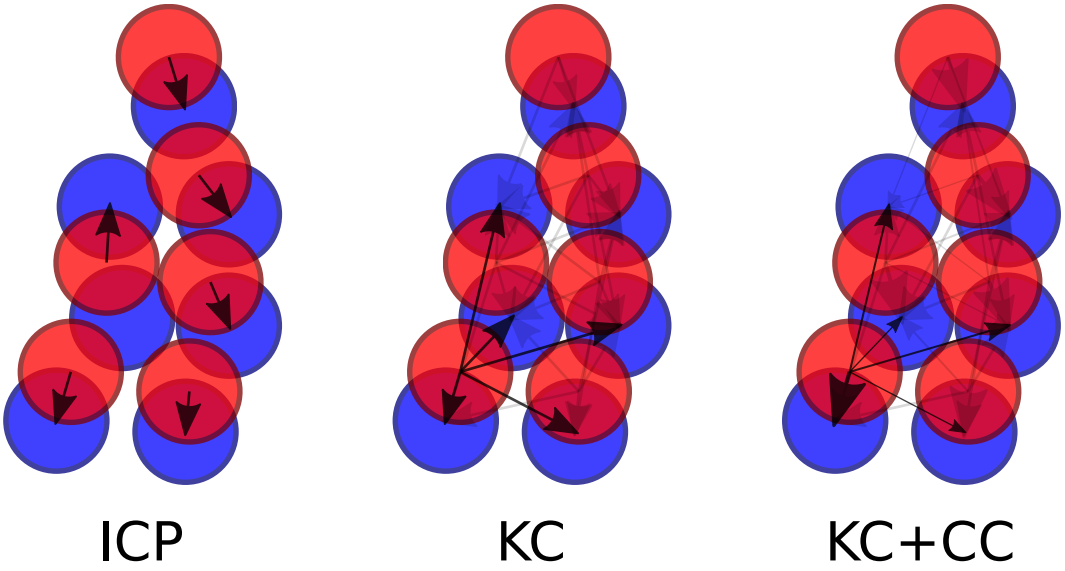


Figure 8.23: A comparison of the ICP, KC and KC+CC algorithms. The red circles represent fruit seen in one image, the blue circles represent fruit seen in another image. For the KC+CC diagram, the size of the arrow represents the weight from the calyx comparison system. For both the KC and KC+CC diagrams, arrows from just one fruit have been highlighted for clarity.

by the image edge (Figure 8.24). Hence, fruit detected within 20 pixels of the edges of images are disregarded.

For each of the fruit in the floating cloud, fruit in the fixed cloud within a search radius are identified. This search radius is the greater of either 0.2 m or 1.5 times the last transform distance (the last transform distance is explained below). For each of the nearby fruit identified in the fixed cloud, a calyx comparison is conducted (Section 8.5). Each calyx comparison score is normalised using the hyperbolic tangent function shown in Equation 8.3 and visualised in Figure 8.25. The threshold (thresh) is set to the calyx comparison optimal threshold, which is 54.0. The denominator value of 15 is chosen so the function approximately mimics the probability of two fruit with a particular score being the same fruit.

$$\text{weight} = 0.5 + \frac{1}{2}\tanh\left(\frac{\text{thresh} - \text{score}}{15}\right) \tag{8.3}$$

A three axis, translational rigid transform is adjusted to minimise a cost function. A three axis translational transform is used rather than a six axis

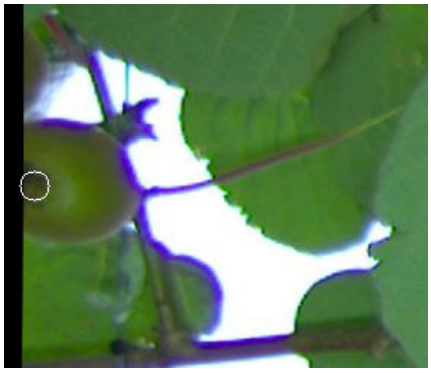


Figure 8.24: An example of a fruit detected on the edge of an image. The white circle shows the detected location of the calyx. When fruit are near the image edge, the centre of the calyx tends to be inaccurately located, which results in localisation error when stereo triangulated. The localisation error causes errors in the fruit tracking system leading double counting of fruit. Hence, fruit detected within 20 pixels of image edges are disregarded.

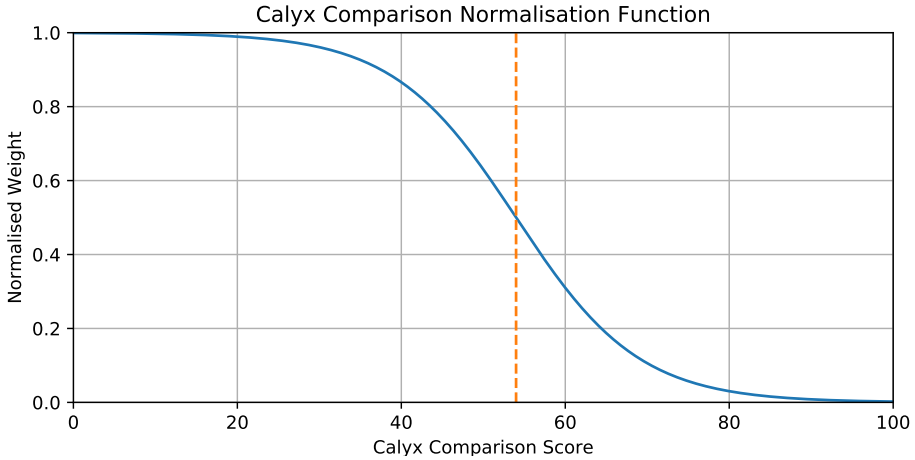


Figure 8.25: The calyx comparison score normalisation function from Equation 8.3. The dotted line represents the threshold (thresh), which is 54.0.

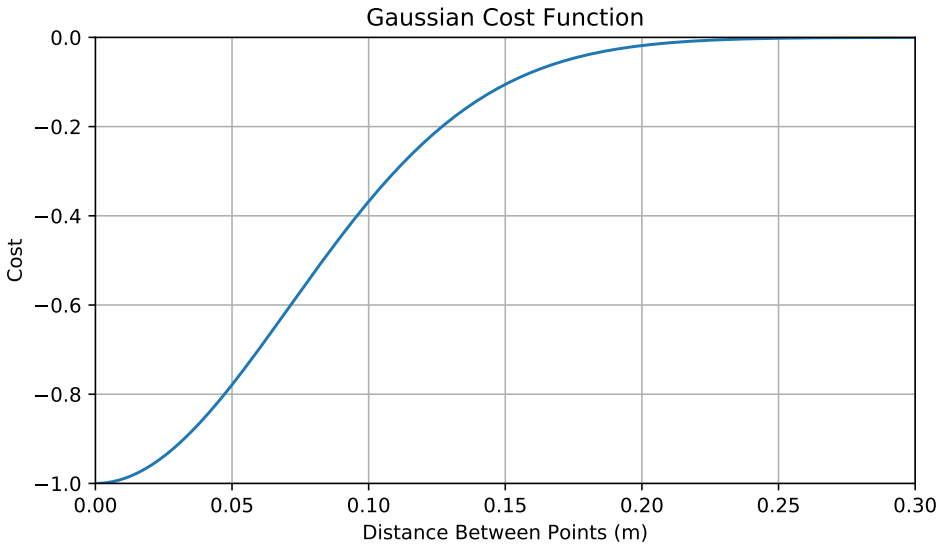


Figure 8.26: The Gaussian part of the cost function used for the transform optimisation, seen in Equation 8.4. Note the Y-axis is negative.

transform to reduce computation time. The cost function consists of two parts, the first being the magnitude of the transform to penalise large transforms. The second part is the sum of a cost between each of the points in the floating and the fixed clouds. The cost between each point is a Gaussian function of the distance between the points and the calyx comparison score between the two fruit. The full cost function is shown in Equation 8.4 with the Gaussian function visualised in Figure 8.26. In practice, the cost function is only evaluated for fruit combinations that have a normalised calyx comparison score above 0.3 to reduce computational intensity.

$$\text{cost} = \text{tf}_{mag} + \sum_{i=1}^{n_{float}} \sum_{o=1}^{n_{fix}} -cc_{io} e^{-{d_{io}}^2/0.01} \tag{8.4}$$

where tf_{mag} is the magnitude of the total transform, cc_{io} is the normalised calyx comparison score between fruit i in the floating cloud and fruit o in the fixed cloud, and d_{io} is the distance between those two fruit. The constant used in the exponential function of 0.01 is set by investigation.

The `minimize` function of SciPy is used with the Broyden-Fletcher-Goldfarb-Shanno (BFGS) solver [118, 122]. The BFGS solver is found to produce accurate results while converging quicker than other solvers available for use with

SciPys `minimize` function. The *gtol*, the gradient normal threshold for optimisation termination, is set to 0.00001, and the maximum number of iterations is set to 50. The optimisation is initialised with an initial guess of 0, (no transform).

The resulting transform produced by the optimiser is applied to all fruit in the floating cloud. The magnitude of the transform is known as the last transform distance and is used in the next iteration of the process.

8.6.2 Match Selection

Once the intra-pass SLAM error has been corrected by the KC+CC system, fruit correspondence is calculated using the match selection system. The match selection system is similar in operation to the stereo matching system (Chapter 7). Two stages are used with the first identifying unambiguous matches and the second selecting the best matching combinations in cases of ambiguity.

The match selection system evaluates potential matches based on a match score. The match score is calculated for each pair of points between the fixed and floating cloud. This match score is the same Gaussian function of distance weighted by the normalised calyx comparison score as used in the KC+CC system and shown in Equation 8.5.

$$\text{matchScore} = cc_{io} e^{-{d_{io}}^2/0.01} \tag{8.5}$$

where cc_{io} is the normalised calyx comparison score between fruit i in the floating cloud and fruit o in the fixed cloud, and d_{io} is the distance between those two fruit (after the KC+CC transform has been applied). For any pair of fruit to be matched, the match score must be greater than a threshold, which is set at 0.01. Figure 8.27 shows the combinations of distance and calyx comparison score that meet this threshold.

Stage one of the system identifies unambiguous matches using an iterative process. The match score for each pair of fruit is checked. If the match score is above 0.01, all of the other match scores for the two fruit are checked. If each

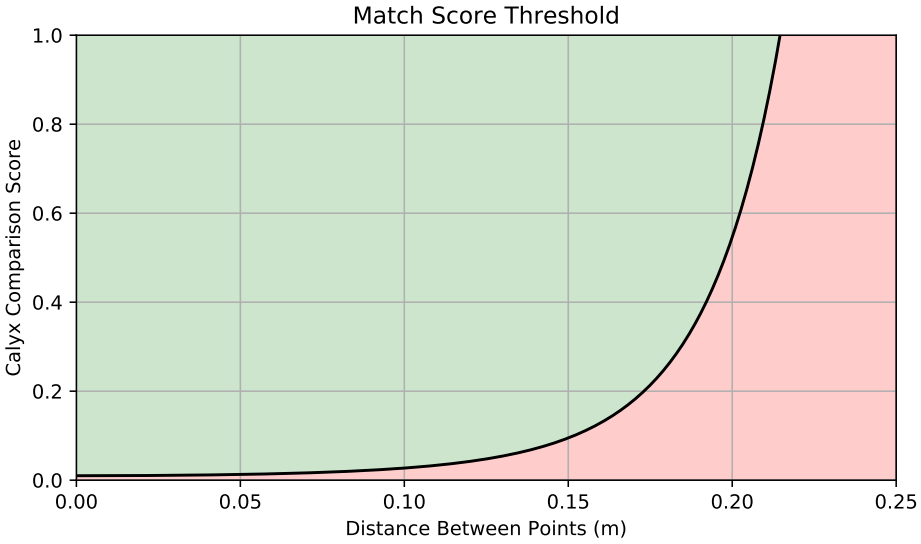


Figure 8.27: The calyx comparison scores and distance between points that are eligible for matching. Any potential matching combination in the green zone will be matched, assuming it meets the other matching requirements.

of those other match scores are less than half of the match score in question, a match is declared. Both the fruit are removed from the remainder of the match selection process. Any fruit with no match scores above 0.01 are removed from the match selection process with no match found. The process is repeated until no changes are made. The process is outlined in Figure 8.28 and an example is shown in Table 8.8.

Stage two of the match selection system takes the remaining unmatched fruit and selects the best matching combinations. Each fruit is assigned to a group of ambiguity. A group of ambiguity is a set of matching combinations where the correct set of matching combinations is not clear. Any two fruit that have a match score >0.01 will be assigned to the same group of ambiguity. If a group of ambiguity has a different number of fixed fruit than floating fruit, 'dummy' fruit are added to correct the imbalance. These 'dummy' fruit can be matched to any fruit in the opposite cloud and represent no match being found for the matching fruit. The match score from any fruit to a 'dummy' fruit is 0, however, it is always considered a valid match despite the match score being <0.01. For each group of ambiguity, the total match score is calculated for

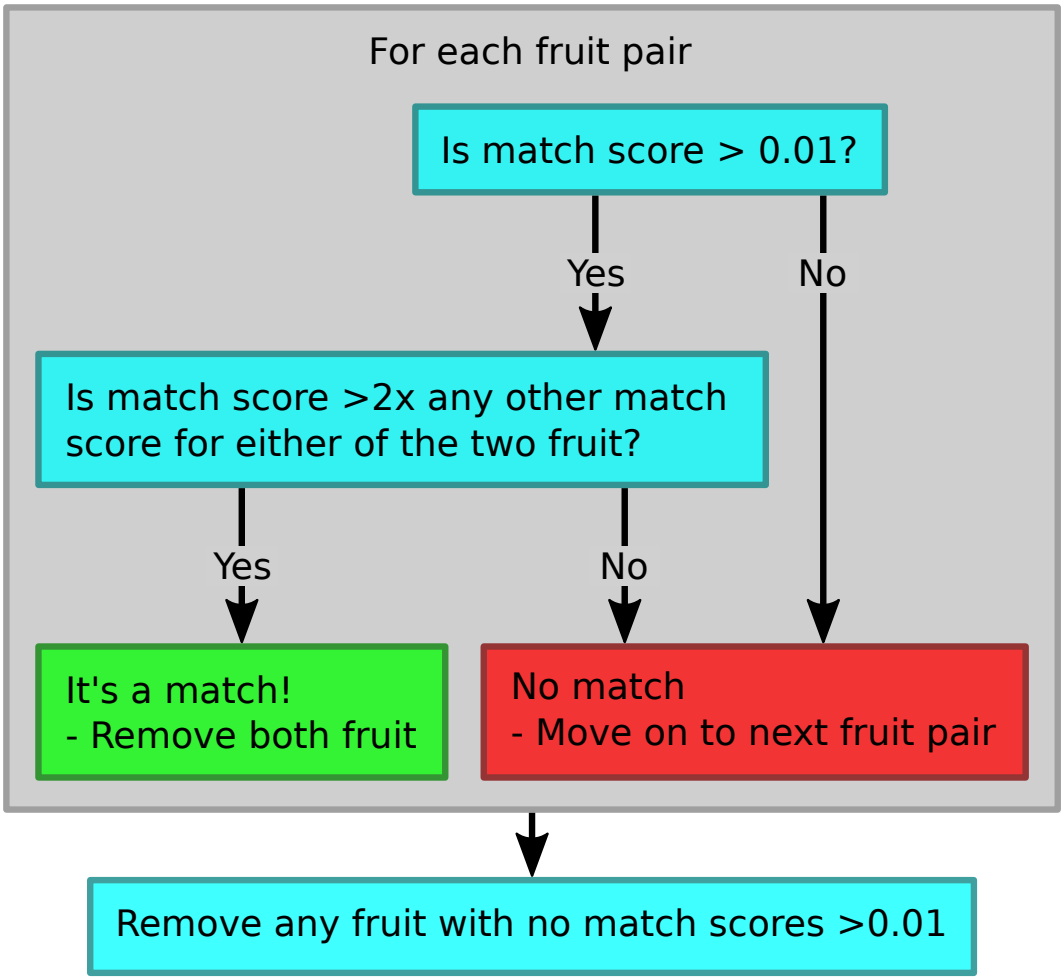


Figure 8.28: A flow diagram for stage one of the match selection process. The match score for each pair of fruit between the floating and fixed cloud is first checked. If it is >0.01 and is more than double the match score of either of the two fruit with any other fruit, it is selected as a match. Once all match scores have been checked and matches selected, any fruit with no match scores >0.01 are removed. The process is then repeated until an iteration is completed with no changes being made.

	1	2	3	4
1	0.04	0	0	0.86
2	0	0	0.20	0
3	0.64	0.04	0	0
4	0.47	0.81	0	0.12
5	0	0	0.73	0.23

→

	1	2
2	0	0
3	0.64	0.04
4	0.47	0.81

→

	1	2
3	0.64	0.04
4	0.47	0.81

Table 8.8: Three iterations of stage one of the match selection system. Each row represents a fruit in the floating cloud, and each column a fruit in the fixed cloud. Each value in the table is the match score for the pair of fruit. The first iteration identifies the two green highlighted match scores. Each of the two matches scores are above the threshold of 0.01 and are more than double any other match scores for those fruit. For example, floating fruit 1 and fixed fruit 4 have a match score of 0.86. The next highest match score for either of those fruit is between floating fruit 5 and fixed fruit 4, of 0.23, which is less than half of 0.86. The two matches are logged and the rows and columns corresponding to those fruit are removed. The second iteration identifies no new matches, but finds floating fruit 2 (red), which has no match scores above the threshold for any of the remaining fixed fruit. Hence floating fruit 2 is removed. Iteration three identifies no new matches as for each match score above 0.01 there is another match score that is greater than half as large for one of the two fruit. The process is terminated as no changes are made during iteration three.

	4	5	8
1	0.73	0	0.28
2	0.38	0	0.20
7	0	0.18	0
9	0	0.29	0

Table 8.9: Stage two of the match selection system. Each row represents a fruit in the floating cloud, and each column a fruit in the fixed could. Each value in the table is the match score for the pair of fruit. Note that fruit have already been removed by stage one of the match selection system, hence the discontinuous numbering. Groups of ambiguity are formed. The fruit marked in yellow belong to one group of ambiguity, and those marked in blue belong to another. For each group of ambiguity, the total match score is calculated for each of the matching combinations. The selected matches are shown in green.

each of the matching combinations. The total match score is the sum of the match scores that make up the matching combination. The combination with the highest total match score is selected. An example is shown in Table 8.9.

For the yellow group of ambiguity shown in Table 8.9, the two matching combinations are:

1. $floating_1$ matched to $fixed_4$ and $floating_2$ matched to $fixed_8$
2. $floating_1$ matched to $fixed_8$ and $floating_2$ matched to $fixed_4$

The total match scores are $0.73 + 0.20 = 0.93$ and $0.38 + 0.28 = 0.66$ respectively. Therefore, matching combination one is selected. For the blue group of ambiguity shown in Table 8.9, there are two floating fruit and only one fixed fruit. Therefore, a dummy fixed fruit is added. The two matching combinations are:

1. $floating_7$ matched to $fixed_5$ and $floating_9$ matched to $fixed_{dummy}$
2. $floating_7$ matched to $fixed_{dummy}$ and $floating_9$ matched to $fixed_5$

The total match scores are $0.18 + 0 = 0.18$ and $0 + 0.29 = 0.29$ respectively. Therefore, matching combination two is selected.

Once all the matches are selected, the orchard block referenced positions of the matched fruit are updated. The position of a matched fruit is the average of the orchard block referenced locations that fruit has been detected in. For example, if a fruit had been seen in two previous image pairs, and is seen in the current image pair, it's new position will be the mean of the three positions. The location after the KC+CC translation has been applied is used as the position from the current image. If no match is found for a fruit in the floating cloud, it is added to the list of previously seen fruit with it's location being that after the KC+CC translation has been applied. The KC+CC and match selection process are then repeated for the next image pair.

8.6.3 Final Fruit Matching

Once all the images in a pass have been processed by the KC+CC and the match selection systems, all fruit should be matched. However, this is not always the case. Consider a situation where a fruit is seen in a series of three image pairs. In the first image pair, the fruit is fully visible, but in the second and third image pairs, the fruit is partially occluded. This occlusion causes a low calyx comparison score between the first and second instances of the fruit, preventing a match. The third instance of the fruit has a match score of 0.3 to the first instance and 0.4 to the second instance. Therefore, it gets matched to the second instance. The result will be two fruit counted where there is really only one fruit. The final fruit matching system is implemented to fix errors like this one.

All pairs of fruit in a pass that are within 0.08 m of each other and are detected in different images are identified. A calyx comparison is conducted between each of these pairs of neighbouring fruit. A final match score is calculated for each neighbouring pair of fruit using Equation 8.6

$$\text{finalMatchScore} = \frac{cc}{d} \tag{8.6}$$

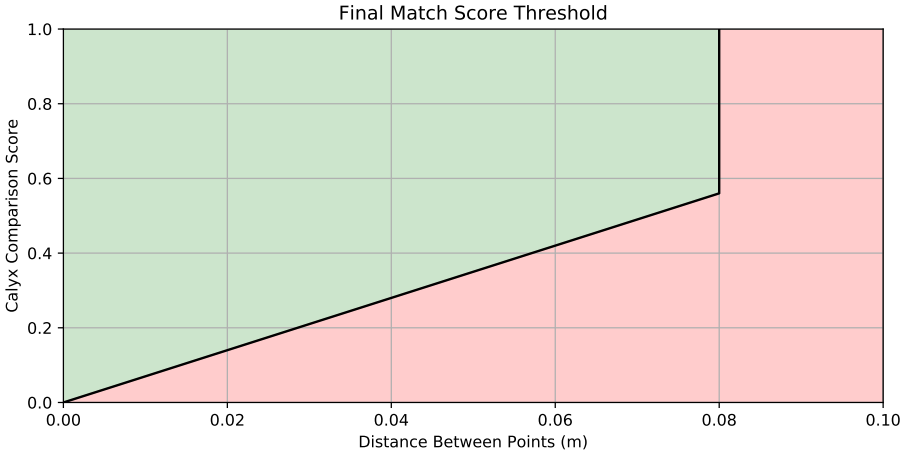


Figure 8.29: The calyx comparison scores and distance between points that are eligible for matching with the final fruit matching system. Any potential matching combinations in the green zone will be matched, assuming they meet the other matching requirements.

where cc is the normalised calyx comparison score between the two fruit and d is the distance between the two fruit. If the final match score is greater than a threshold, set at 7, and is the highest final match score for either of the two fruit, it is declared a match. The combination of distances and calyx comparison scores that can be matched are shown in Figure 8.29.

The matched fruit have their positions averaged as in the match selection system. The final matching process is repeated until no further matches are found.

8.6.4 Evaluation

To evaluate the performance of the intra-pass fruit tracking system, a fruit is chosen at random from an orchard block. This chosen fruit is a 'test' fruit. The closest five fruit, or all fruit within a 0.2 m radius of the first test fruit, whichever is fewer, are also selected as test fruit. All images taken within a 1.0 m radius of the chosen fruit are identified (measured in only the horizontal plane). In each of the image pairs, the test fruit are marked with a coloured circle. The colour of the circle denotes which of the test fruit it is. The images are viewed and each fruit is tracked through the series of images. If the fruit is

Metric	Fruit	Percentage of Total
Total Fruit	792	100%
Correctly Tracked	738	93.2%
Double Counted	42	5.3%
Mixed up	12	1.5%

Table 8.10: The intra-pass fruit tracking system evaluation results.

correctly identified in each image it is detected in, it is recorded as 'correctly tracked'. If a fruit is identified as two different fruit, it is recorded as 'double counted'. If two fruit are both identified as the same fruit in different image, it is recorded as 'mixed up'. An example of correctly tracked fruit is shown in Figure 8.30. The process is repeated six times for each of the 25 orchard blocks in the fruit training data set.

Computation time is measured for a single pass. The time for each iteration of the final fruit matching process is also measured along with the number of fruit matched in each iteration. Computation is carried out on the PC listed in Table 6.4. All algorithms are single threaded, so the high core count on the CPU used does not effect results.

8.6.5 Results

The results of the intra-pass fruit tracking system evaluation are shown in Table 8.10. Of the 792 fruit observed, 93.2% are correctly tracked through all images they are detected in. Partial occlusion is the largest single cause of errors, causing 31.5% of all errors. An example of partial occlusion causing matching errors is shown in Figure 8.31. An example of the system correctly matching fruit in challenging imaging conditions is shown in Figure 8.32.

Processing time for a pass consisting of 1539 image pairs is 144.5 s. Of that time, 15 s is spent loading data from disk. The final fruit matching system took 48.5 s of the total time to run (Table 8.11). Before the final fruit matching system is run, there are 24943 fruit. Afterwards there are 11772, a

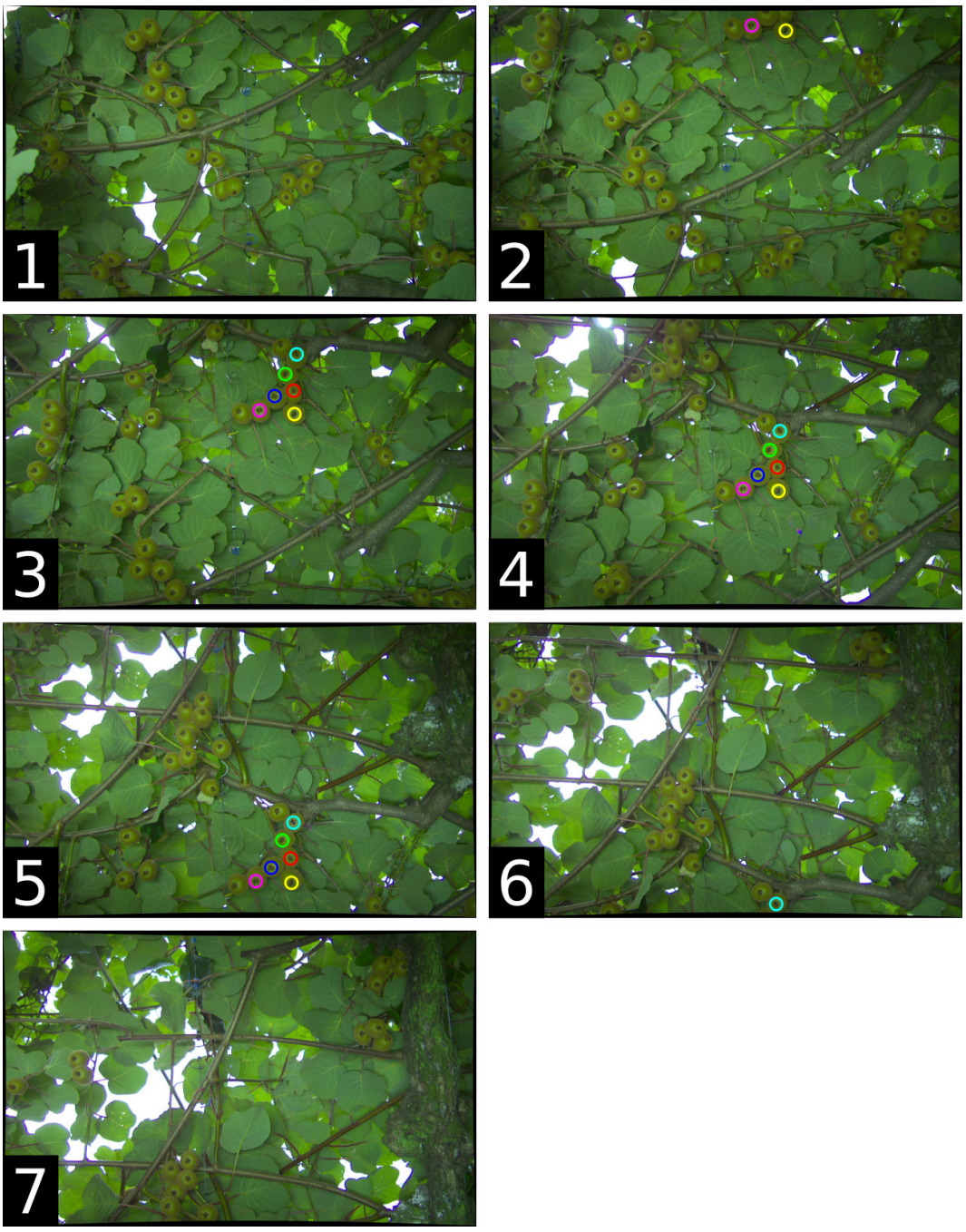


Figure 8.30: An example of the intra-pass fruit tracking system evaluation method. The seven images shown are a sequence of images taken near the location of the test fruit. Note, only one image of each stereo pair is shown, but for evaluation purposes, both the images of each stereo pair are used. All six of the highlighted test fruit are correctly identified in all images they are visible in.

Figure 8.31: An example of the intra-pass fruit tracking system failing due to occlusion. The fruit marked with yellow in image 2 is visible in both the other images. The partial occlusion in images 1 and 2 cause a low calyx comparison score, preventing the correct matches from being made. Note, only the left image of each camera pair is shown for simplicity.

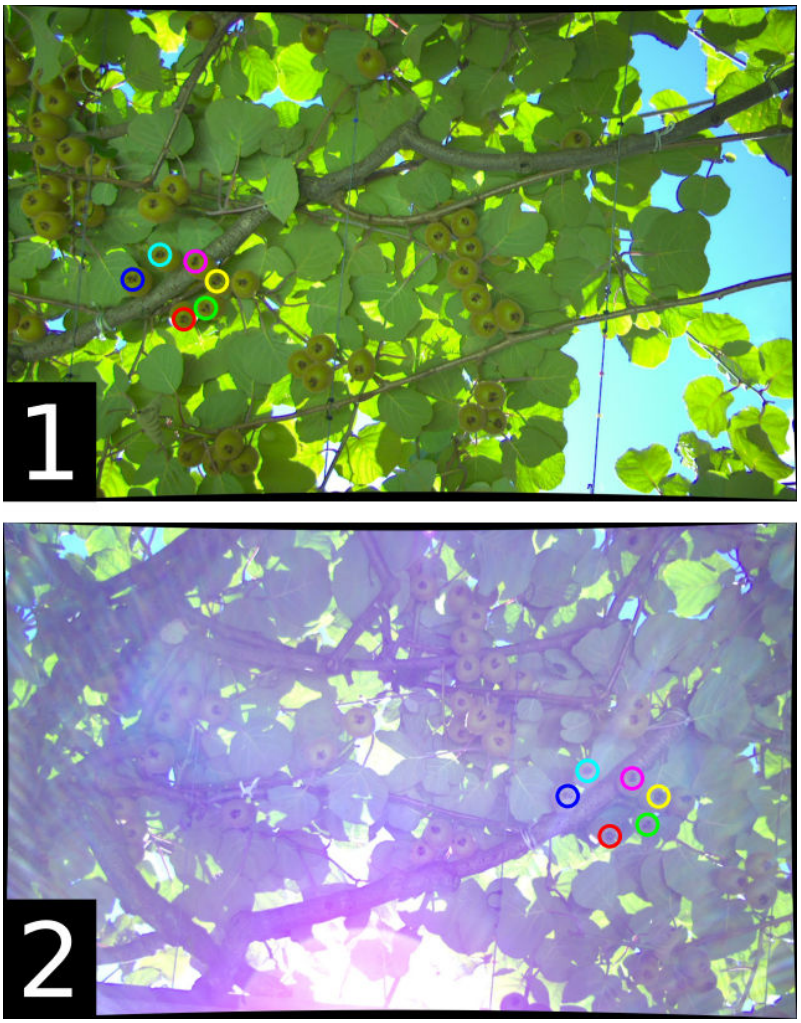


Figure 8.32: An example of the intra-pass fruit tracking performing well despite severe glare in the images. The two images shown are from different camera pairs. Note, only the left image of each camera pair is shown for simplicity.

Iteration	Fruit Matched	Time Taken
1	7353	29.8 s
2	3205	10.8 s
3	951	3.8 s
4	208	1.5 s
5	48	1.0 s
6	7	0.8 s
7	0	0.8 s
Total	11772	48.5 s

Table 8.11: The number of fruit and time taken for each iteration of the final fruit matching system.

52.8% decrease. This shows that the final fruit matching system is contributing significantly to the overall accuracy of the fruit tracking system.

8.7 Inter-Pass Fruit Rejection

Due to the large SLAM errors encountered, achieving inter-pass multi-frame fruit tracking is left as future work. However, for this yield estimation system, a method of preventing double counting due to inter-pass image overlap is required. Therefore, a simple fruit rejection system is implemented.

For each fruit in a row, the closest image capture location is identified (measured only in the horizontal plane). If that image is part of the same pass as the fruit, it is kept, otherwise, it is discarded. Figure 8.33 illustrates the process.

The inter-pass fruit rejection system removes the overlap between passes and hence the over-counting bias that overlap would introduce. The system does not take into account the SLAM error, so it cannot be expected to perform well on a per fruit basis. However, on an orchard block scale, it should perform adequately. An example of performance on real data is shown in Figure 8.34.

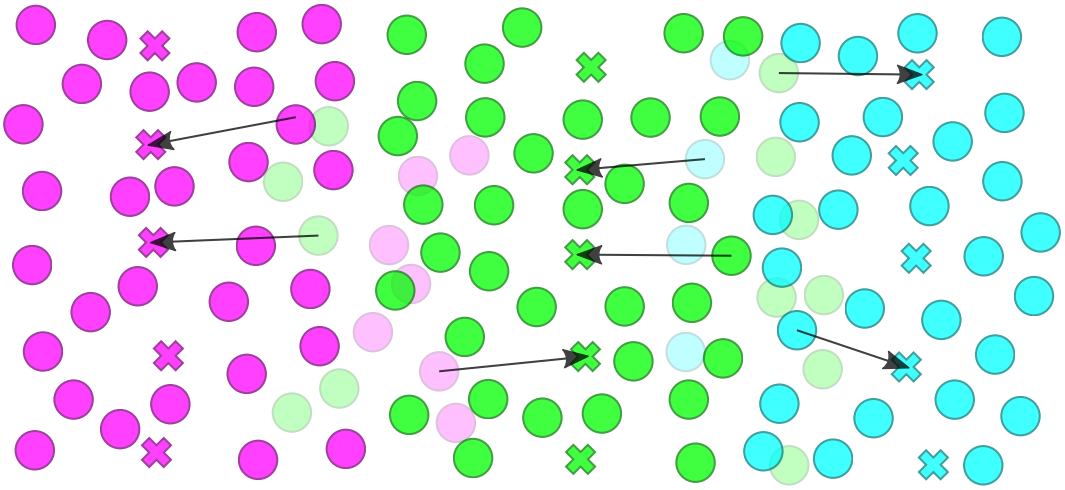


Figure 8.33: An example of the inter-pass fruit rejection system, viewed from above. The three colours represent the three passes down a row. Circles represent fruit, crosses represent the location of the cameras at the time of imaging. Any fruit that is closer to an image location of a different pass is rejected (represented by the greyed out circles). The closest image location for a selection of the fruit is indicated by the arrows.

8.8 Fruit Visualisations

With all fruit matching and fruit rejection finished, the fruit in an orchard can be visualised. Figure 8.35 shows a section of an orchard block from above with the rows highlighted. A section of an orchard block with fruit coloured by height is shown in Figure 8.36. A fruit density map overlaid on an aerial photograph of the orchard is shown in Figure 8.37. These visualisations can be used to quickly and easily identify which areas of the orchard require attention from the grower.

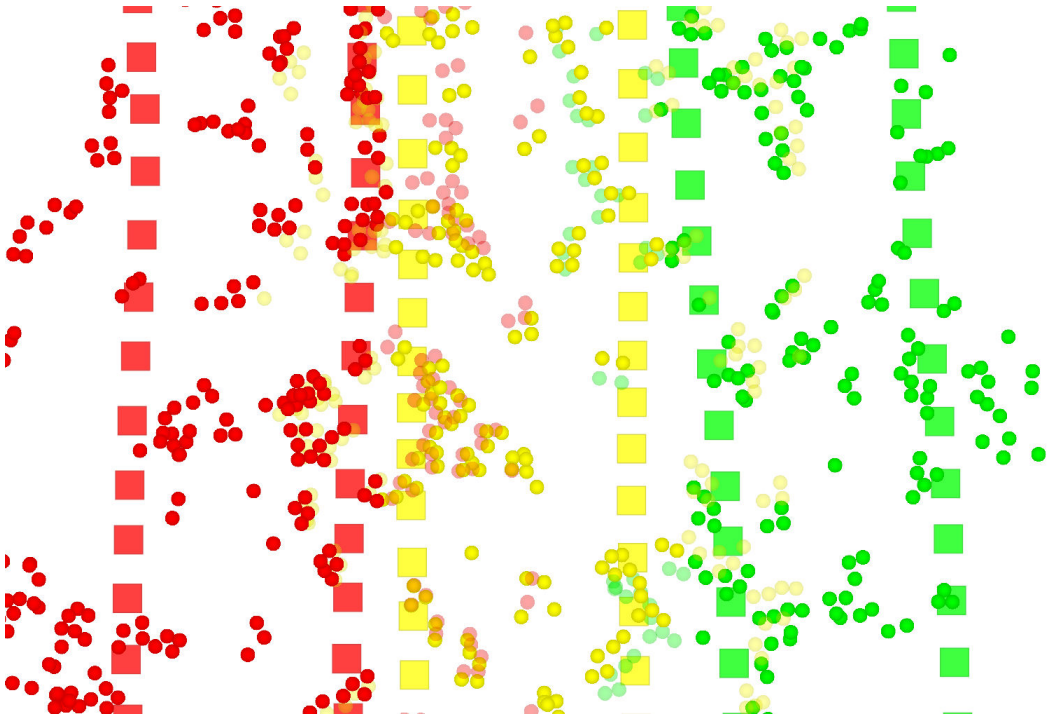


Figure 8.34: An example of the inter-pass fruit rejection system with real data, viewed from above. The three colours represent different passes down the same row. The squares are the camera locations when images are taken. Note there are two rows of each colour square as there are two stereo camera pairs on the data capture system. The circles represent fruit with the greyed out fruit being those that are rejected by the system. Correspondence can be seen between the kept and rejected fruit.

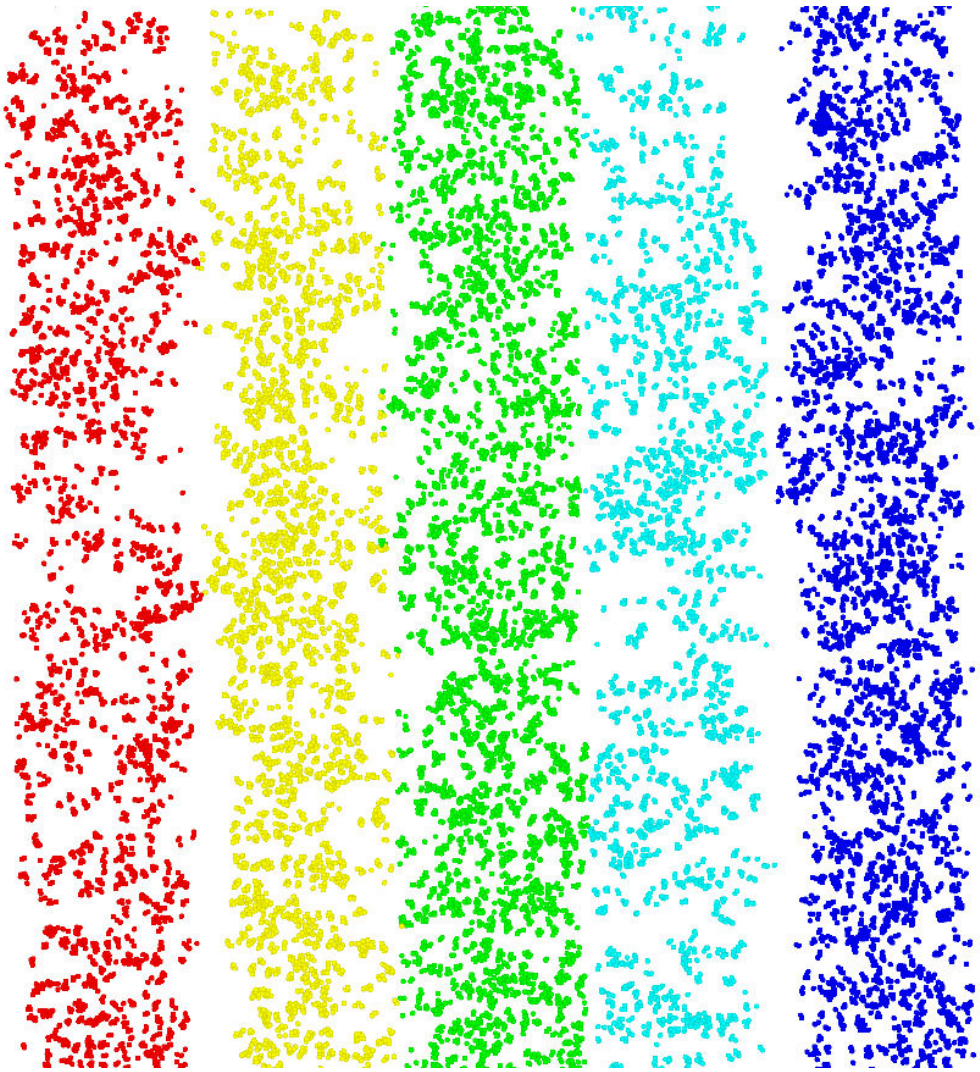


Figure 8.35: The final fruit point cloud viewed from above. The colours denote the rows of the orchard block. Note, only a section of the orchard block is shown.

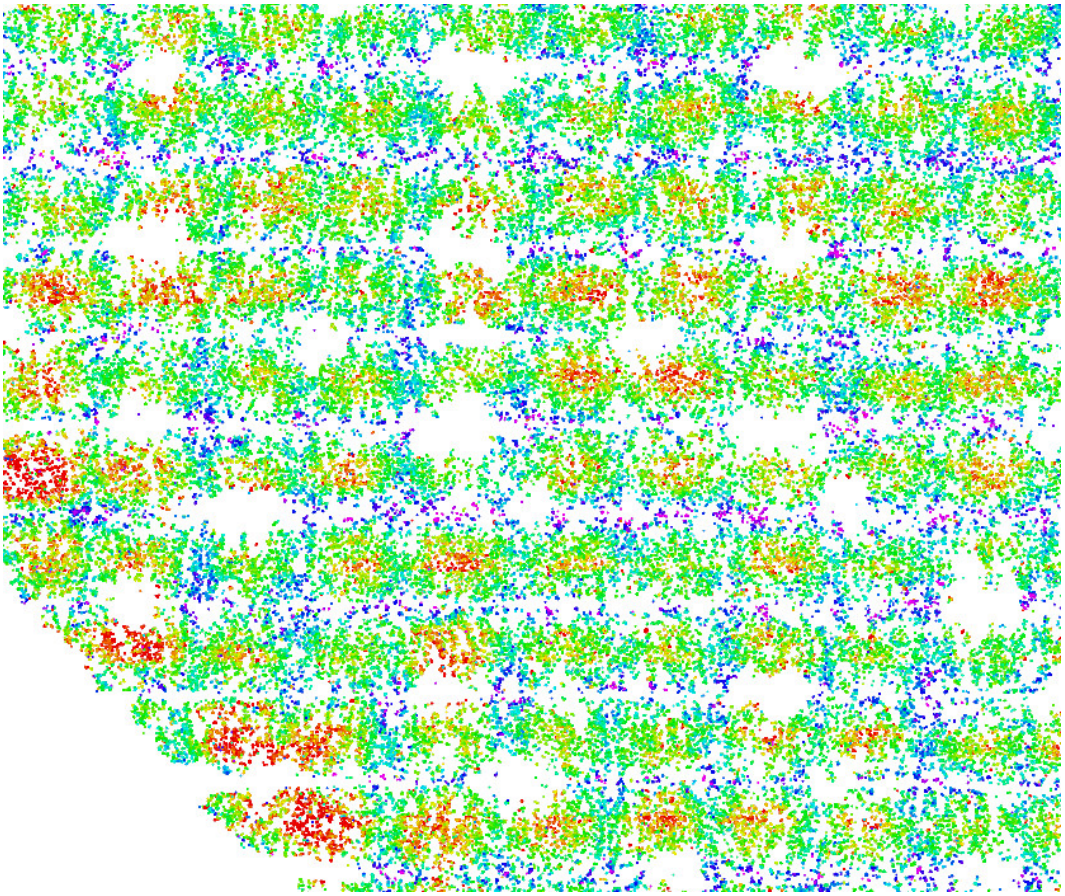


Figure 8.36: The final fruit point cloud viewed from above. The fruit are coloured based on their height. Low fruit are red, average height fruit are green and high fruit are blue/purple. Rows are oriented across the image, which can be seen by the bands of higher (blue/purple) fruit. The vertical bands of blue fruit are where the beams cross over the top of the rows. The areas of no fruit that can be seen along the edges of some rows are male plants, which do not produce fruit (they provide pollen to the female plants).

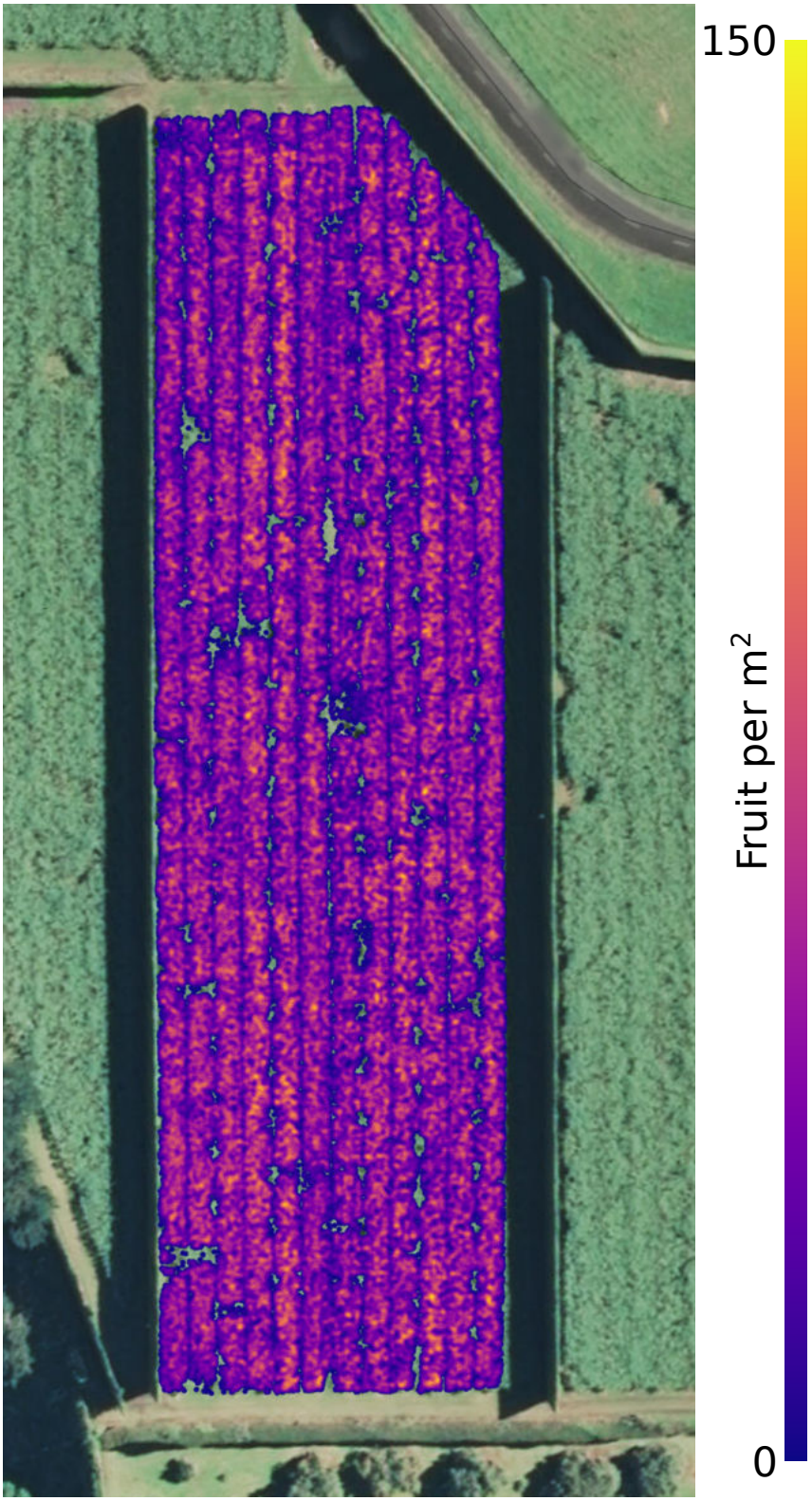


Figure 8.37: A fruit density map of an orchard block, displayed over an aerial photograph of the orchard. Some areas of the orchard are more productive than others. The small evenly spaced areas of no fruit are where the male plants are. Aerial photograph sourced from the Land Information New Zealand (LINZ) Data Service and licensed for reuse under the Creative Commons BY 4.0 licence.

Chapter 9

Model Parameters

Forming a model to predict the occlusion rate in an orchard requires parameters that correlate with occlusion rate. Figure 1.7 shows how the spatial distribution of fruit can effect the occlusion rate. The density and uniformity of the spatial distribution are two factors that can cause varying occlusion rate. Higher fruit density will result in higher occlusion, as will a highly non uniform spatial distribution.

Orchard blocks come in a range of shapes, sizes and arrangements. These differences in the layout of the orchard could also effect the occlusion rate. Therefore, the properties of the orchard layout are parametrised.

9.1 Fruit Density

The more fruit per area, the more likely any fruit is to be occluded. Fruit density is calculated as the number of fruit detected divided by the area of the orchard block. The fruit density across four orchard blocks is shown in Table 9.1. It can be seen that the fruit density varies significantly with the most dense block having twice the density of the least dense.

Orchard Block	Average Fruit Density
A_01	**61.2 fruit/m^2**
P_C2	**29.5 fruit/m^2**
K_A1	51.1 fruit/m^2
N_01	41.9 fruit/m^2

Table 9.1: The estimated fruit density of four orchard blocks. Fruit density varies greatly between blocks.

Orchard Block	Fruit Height Standard Deviation
P_C1	0.123 m
K_E1	**0.108 m**
G_11	0.120 m
S_01	**0.136 m**

Table 9.2: The standard deviation of fruit heights across four orchard blocks.

9.2 Fruit Height Variation

Fruit that are high in the canopy are more likely to be occluded by leaves, branches, orchard structure and other fruit. Fruit that are low are likely to occlude the higher fruit. Those that are very low will not be counted as the stereo localisation system will not accept fruit below a height of 1.34 m from the ground. Therefore, it follows that the more fruit in the higher and lower areas of the canopy, the higher the occlusion rate will be.

Average canopy height varies between orchards. Figure 9.1 shows the distribution of fruit heights in four orchard blocks. It can be seen that the range of fruit heights for all four orchard blocks is very similar (as dictated by the limits of the stereo localisation system). Therefore, the absolute range of fruit heights is not a useful measure. The distribution of fruit heights is different between the four orchard blocks, with S_01 having a lower peak with a wider distribution when compared to K_E1. This difference in distribution is captured by the standard deviation of the fruit heights as seen in Table 9.2.

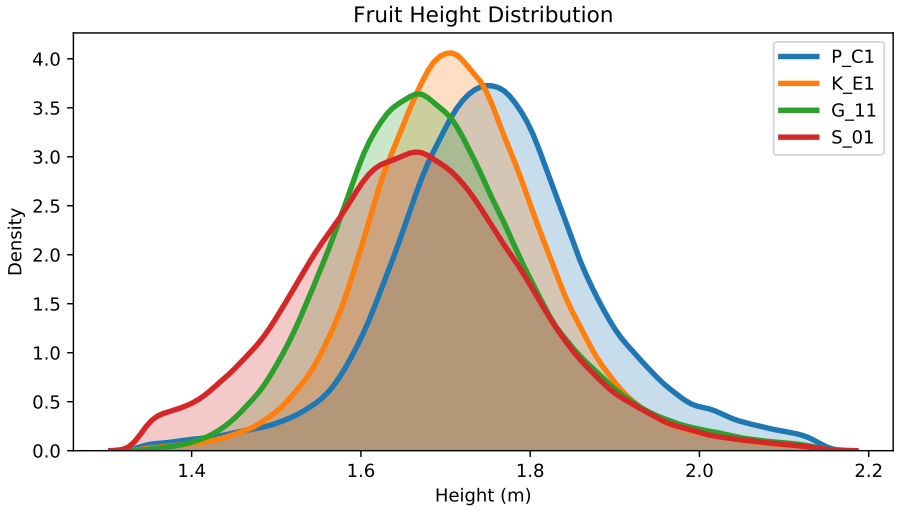


Figure 9.1: The fruit height distribution for four orchard blocks. The average height is different for each of the orchard blocks, as is the standard deviation.

9.3 Fruit Clustering

Kiwifruit tend to grow in clusters, which contributes to a higher occlusion rate. Larger clusters will likely have a higher occlusion rate than smaller clusters. For this work, a cluster is defined as a group of fruit where each fruit is less than 0.1 m from another fruit in the cluster. Put another way, if two fruit are within 0.1 m (Euclidean distance) of each other, they belong to the same cluster. The threshold of 0.1 m is found by taking physical measurements of clusters in orchards.

Figure 9.2 shows a visualisation of part of an orchard block with clusters indicated. Figure 9.3 shows the distribution of cluster sizes for an orchard block.

The mean cluster size captures the difference between clustering characteristics in orchards. Table 9.3 shows the mean fruit count per cluster for four orchard blocks.

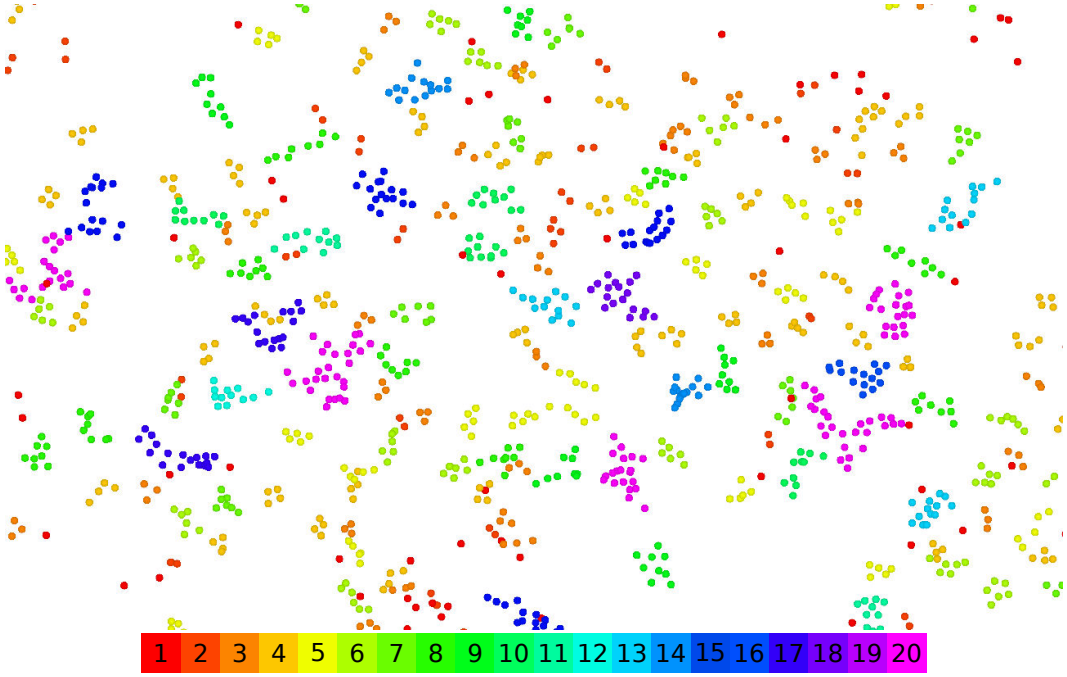


Figure 9.2: Fruit viewed from above. The colour of each fruit indicates the number of fruit in the cluster the fruit is part of. Note fruit that appear to be close to each other in the image can be at different heights and hence not as close to each other as they appear in a 2D image.

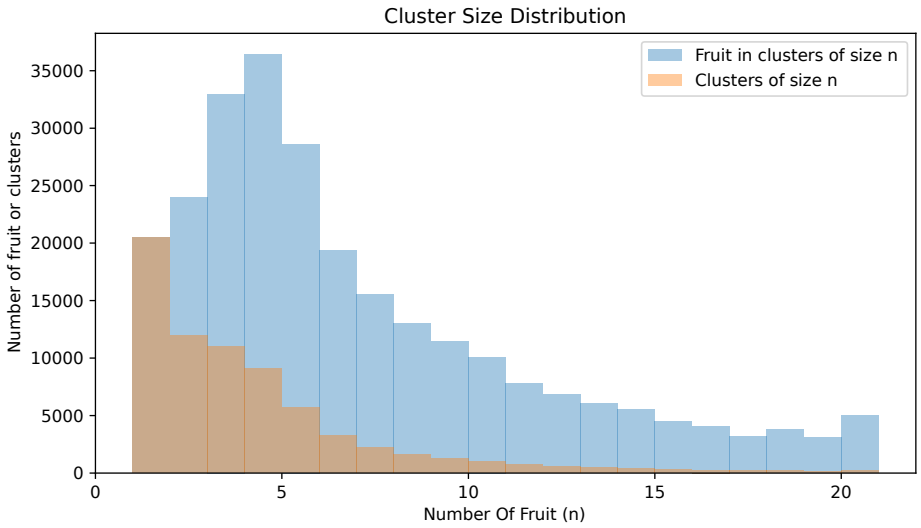


Figure 9.3: A histogram of the cluster sizes in an orchard block.
Note all clusters larger than 20 fruit are plotted as having 20 fruit for clarity.

Orchard Block	Mean Fruit Per Cluster
A_02	**5.5**
P_A1	4.7
P_C6-8	**3.4**
K_E1	3.8

Table 9.3: The mean number of fruit per cluster across four orchard blocks.

9.4 Fruit Distribution Across Row

Fruit are not distributed evenly across the row. More fruit tend to be in the centre than along the edges of the row. Hence, the occlusion rate is likely to be higher in the centre than near the edges.

To quantify the distribution of fruit across the row, the following method is used, which is shown in Figure 9.4. For the centre pass down each row, all of the image locations are found (Section 4). The image locations are the location of the two camera pairs on the data capture system, at the time of each image capture. As the two camera pairs are triggered simultaneously, the image locations are arranged in pairs, one from each of the two camera pairs every time a set of images is captured. The two image locations are used to define a line. This line is perpendicular to the direction of the row, assuming the data capture system is being driven in the direction of the row (which is almost always the case). A zero point is defined on the line at the point halfway between the two image locations. For each fruit in a row, the closest line is found. The distance along the line that the fruit is located, is the position of the fruit across the row.

The position across the row of all fruit in an orchard block is shown in Figure 9.5. To parametrise the distribution, it is compared to a normal distribution with the same mean and standard deviation using a Kolmogorov-Smirnov (KS) test. The KS statistic is used as a measure of the conformance to a normal distribution. KS statistics for four orchard blocks is shown in Table

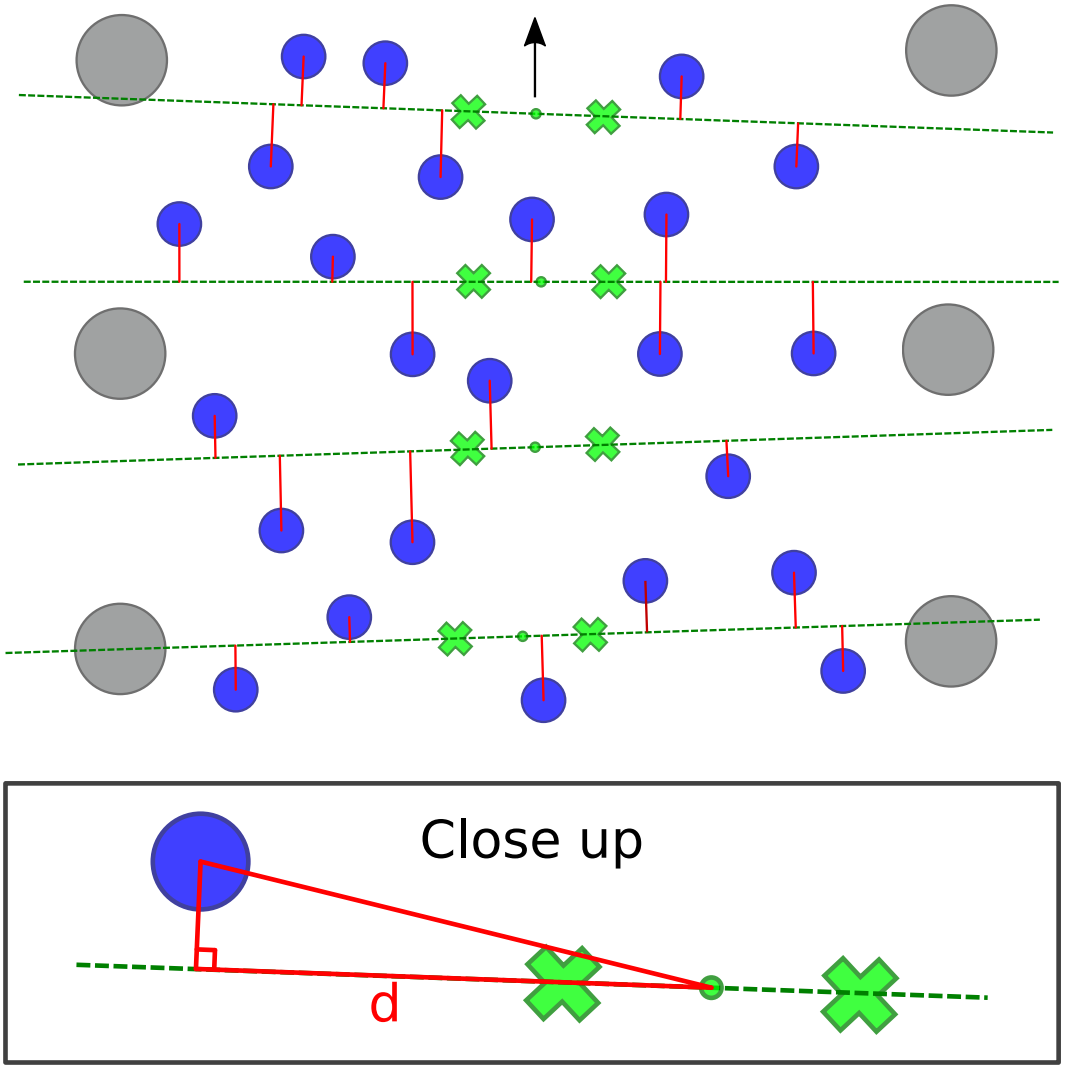


Figure 9.4: A diagram demonstrating how the position of fruit across the row is measured. The black arrow shows the direction of travel of the data capture system. The grey circles represent the posts and trunks that border each row. The blue circles are the fruit. The green crosses represent the image locations for the two stereo camera pairs on the data capture system during the centre pass down a row. A line is drawn that passes through each pair of these image locations (green dotted lines). The closest line to each fruit is found. The position of the fruit along the closest line (d) is measured relative to the centre point of the line (marked by the small green circles).

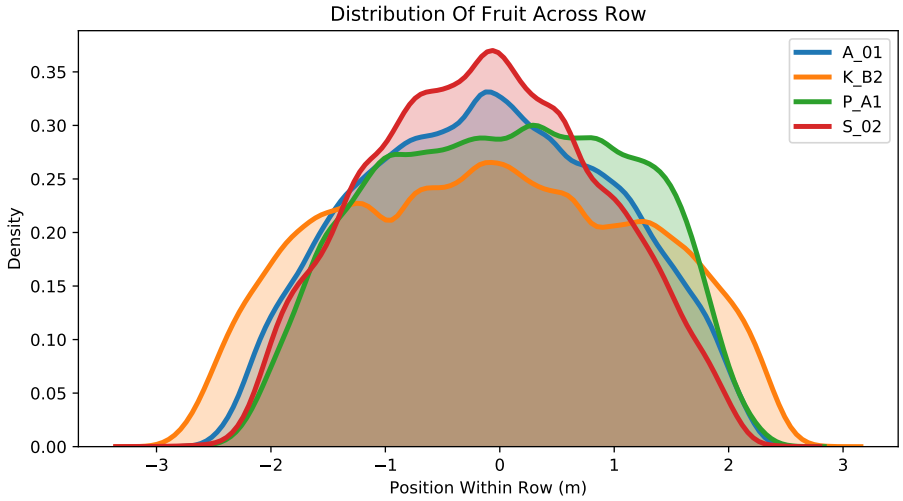


Figure 9.5: The distribution of fruit across the row for four orchard blocks. S_02 has a much higher concentration of fruit at the centre of the row compared to P_A1 and K_B2.

Orchard Block	KS Statistic Compared To Normal Distribution
A_01	0.033
K_B2	0.045
P_A1	**0.048**
S_02	**0.025**

Table 9.4: The KS statistic comparing a normal distribution to the actual distribution of fruit across the row in four orchard blocks.

9.4.

9.5 Orchard Area

The area of the orchard is the most fundamental parameter to describe the layout. Measurement of the area is not required as most (possibly all) orchards have accurate canopy area measurements available. Figure 9.6 shows an example of an orchard map, which has the area of each of the blocks in the orchard. Orchard maps are supplied for all of the orchards visited for data capture by the orchard managers. The area reported on these maps is used for the yield prediction system.

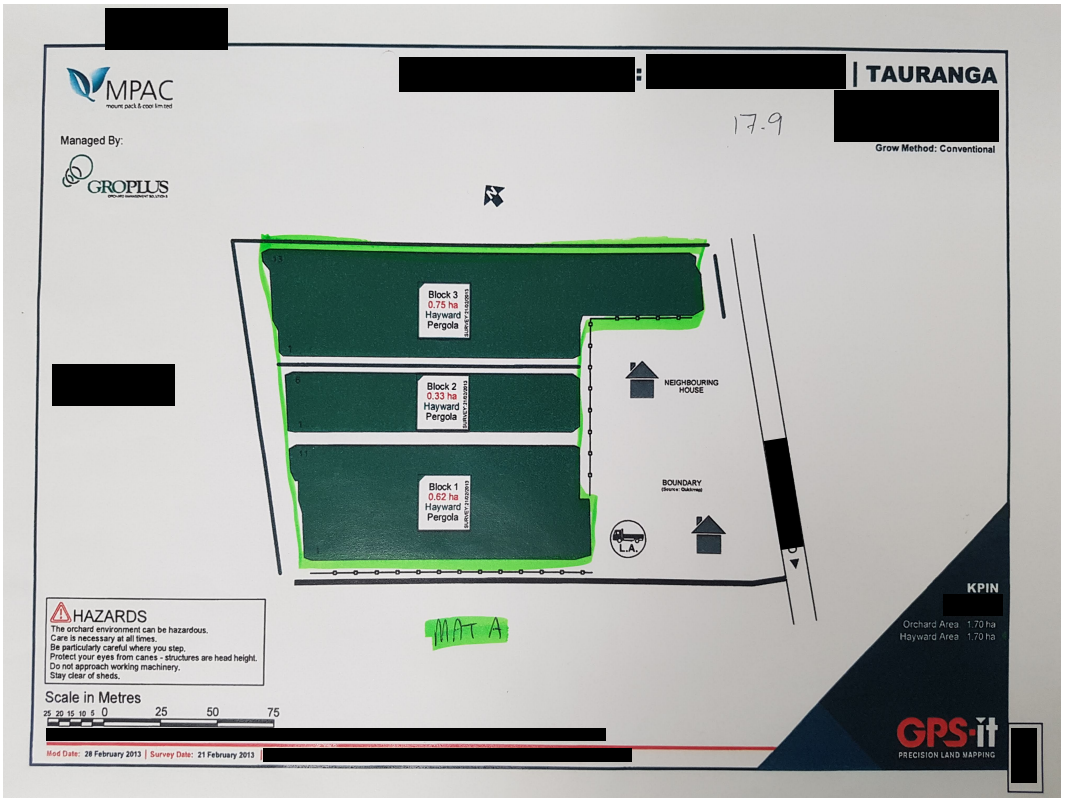


Figure 9.6: An example of an orchard map. The orchard has three blocks shown in green. The area of each block is reported in red. Similar maps are made available by the orchard managers for all orchards visited for data collection. Note, identifying information has been censored for privacy reasons.

Figure 9.7: Diagram showing the side row overhang. The vertical lines are the posts, supporting the canopy. The horizontal line is the canopy, which hangs down over the edge of the orchard block on the left side. The orchard continues to the right. The data capture system cannot be driven under the overhang as it is too low.

9.6 Orchard Block Perimeter

The outer edges of orchard blocks can support fruit that may not be visible to the data capture system. Therefore, orchard edges require parametrisation to be accounted for by the yield prediction system. However, the ends of rows and the outer sides of rows are managed differently by growers and hence are treated separately.

9.6.1 Side Row Overhang

Most orchards have overhang on the sides of the outer rows, as seen in Figures 9.7 and 9.8. This overhang provides additional fruiting area for the plants. The overhang is typically too low to drive the data capture system under. The dimensions of the overhang is inconsistent between orchards. In some orchards, the overhang is very narrow, extending only a small distance from the edge of the row ($\sim$0.5 m), while in others it can extend up to a full row width.

Figure 9.8: View from within the first row of an orchard. The side overhang can be seen in the left side of the image. Fruit can be seen growing on the overhang. These fruit will not be detected as the data capture system cannot be driven under the side overhang.

9.6.2 End Row Overhang

In contrast to the overhang on the sides of the outer rows, there is generally very little overhang on the ends of rows (Figure 9.9). This is because people and orchard equipment (tractors, sprayers, robotic harvesting machines etc.) enter and exit the rows from the ends and would likely damage overhanging fruit.

9.6.3 Measurement

To measure the length of the side row and end row of each orchard block, the SLAM map is used. The SLAM map is loaded in RViz (data visualisation tool part of ROS) and the built-in measure tool is used to measure the lengths of the side row and end row. Some orchard blocks have a section of perimeter that is not perpendicular to the direction of the rows. In this case the length is measured and counted as half side row and half end row, as shown in Figure

Figure 9.9: Unlike the outer row sides, row ends have little to no overhang. Although there are still rogue branches. The branch pictured managed to snag the first aid kit off of the data capture system as it was driving through.

9.10.

The perimeter, side row and end row are quantified not only by the raw numbers, but also as a ratio to of the orchard area. For example, the ratio of orchard area to row side length. Expressing them as a ratio of the orchard area has the effect of normalising the values, which may be more useful for the yield estimation model.

9.7 Row Width

The average row width is measured using the SLAM map and RViz. A measurement is taken across the width of the block, perpendicular to the direction of the rows (Figure 9.11). The distance is divided by the number of rows in the block.

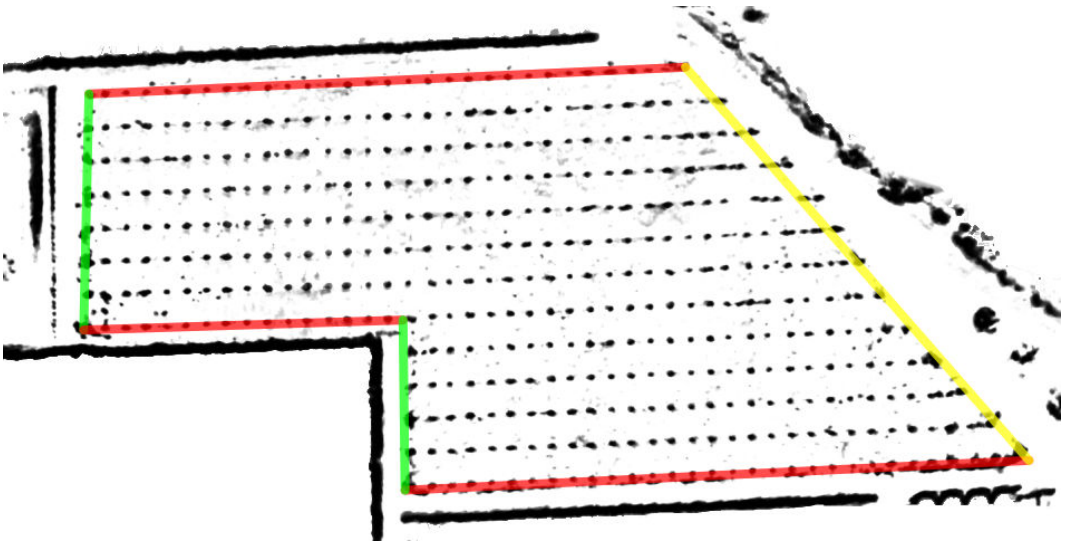


Figure 9.10: SLAM produced map of an orchard block showing how the sections of the perimeter of the block are classified. Red lines are side row, green is end row and yellow is counted as half side row, half end row.

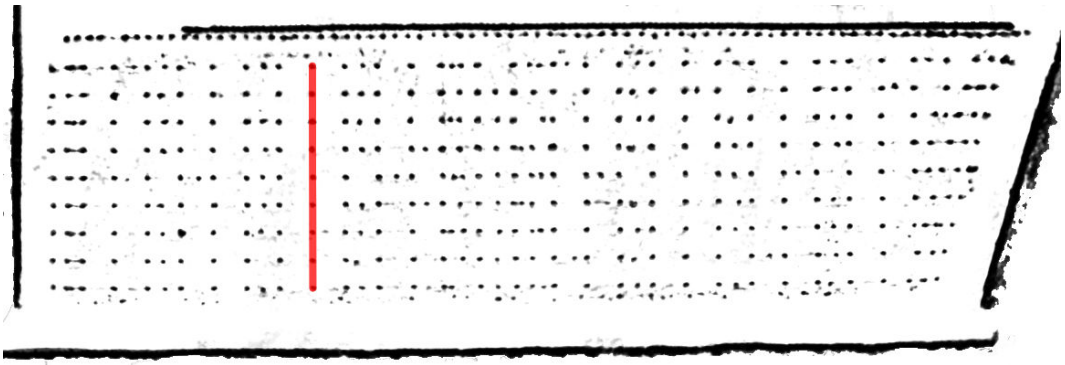


Figure 9.11: SLAM produced map of an orchard block showing how the average row width is measured. The measurement is taken across the orchard block, perpendicular to the direction of the rows, as shown in red. The distance is then divided by the number of rows, in this case there are eight rows.

9.8 Final Parameters

The ground truth data for the yield predictions is supplied on a per maturity area basis. However, data capture and analysis is conducted on a per orchard block basis. Each maturity area consists of one or more orchard blocks. Therefore, to make predictions on a maturity area basis, data from multiple orchard blocks must be combined.

For parameters that accumulate, such as orchard area or fruit count, the values are summed for all blocks of a maturity area. For parameters that are a descriptive statistic, such as average canopy density or height standard deviation, a weighted mean is used where the weight is the detected fruit count in that block.

The complete list of parameters used for the occlusion model creation is shown in Table 9.5. These parameters are henceforth referred to as the 'predictor variables' and are used by the occlusion prediction model to predict the occlusion rate of the orchards.

Parameter	**Source**
Orchard area	Orchard maps
Average row width	SLAM map
Row end length	SLAM map
Row side length	SLAM map
Perimeter	SLAM map
Area to perimeter ratio	Orchard map and SLAM map
Area to row end ratio	Orchard map and SLAM map
Area to row side ratio	Orchard map and SLAM map
Average canopy density	Canopy density estimation system
Height standard deviation	Fruit point cloud
Mean number of fruit per cluster	Fruit point cloud
Distribution across row	Fruit point cloud
Fruit density	Fruit point cloud and orchard map
Algorithm fruit count	Fruit point cloud

Table 9.5: List of all parameters for the occlusion prediction system and the source of the data.

Chapter 10

Occlusion Prediction Model

The purpose of the occlusion prediction model is to take the measured orchard parameters (predictor variables) and accurately predict the correction factor for each maturity area. The correction factor is the ground truth fruit count divided by the number of fruit detected in the maturity area (Equation 10.1). Put another way, it is a representation of the occlusion ratio, or the proportion of fruit that are present in an orchard but not detected by the imaging system due to occlusion.

$$\text{correctionFactor} = \frac{\text{groundTruthFruitCount}}{\text{fruitDetected}} \tag{10.1}$$

If the model can accurately predict the correction factor, the number of fruit in an orchard can therefore be inferred.

10.1 Ground Truth Fruit Count

The yield estimation system is estimating the number of fruit in a maturity area. However the ground truth data is not obtained as a raw fruit count. Instead, the number of class one fruit is given as the number of trays of each count size. Converting number of trays to number of fruit is trivial as the count size of the tray is the number of fruit in that tray. For example, 100 trays of count size 30 fruit contains $100 * 30 = 3000$ fruit. However the quantity of rejected fruit is presented as a mass, eg. 26737.0 kg of reject fruit. To estimate

the number of rejected fruit from the weight, the following equation is used:

$$\text{numRejectFruit} = \text{rejectMass}\frac{\text{averageFruitSize}}{3.6} \tag{10.2}$$

This equation assumes that the rejected fruit are, on average, the same mass as the class one fruit. This assumption is not strictly true but without more detailed information, it is the best estimation available. The average fruit size is measured in count size (the number of fruit per tray), so it is divided by the mass of a tray of fruit, which is 3.6 kg. The ground truth value for fruit in a maturity area is the number of class one fruit plus the number of rejected fruit.

10.2 The Data

Comparing the algorithm fruit counts to ground truth fruit counts over the training dataset shows a very strong correlation (Figure 10.1). A simple linear trend line has an R^2 value of 0.993. The trend line has the following equation:

$$\text{totalFruit} = 1.069 * \text{algorithmFruitCount} + 51803 \tag{10.3}$$

This equation does not have a zero Y-axis crossing, which suggests that an orchard in which no fruit are detected would still contain fruit. Clearly an orchard with no detected fruit isn't likely to have over 50,000 fruit present. However the non zero Y-axis crossing could be explained by the area of the orchard blocks that is not imaged, in particular the side row overhang.

On average the algorithm fruit counts are 11.8% lower than ground truth, with a standard deviation of 6.9% (Table 10.1). The high standard deviation is due to one maturity area in particular (discussed in Section 10.2.1). Without the outlier maturity area, the standard deviation is 2.9%.

Calculating the correlation coefficient (Pearson product-moment coefficient) for each of the predictor variables against the ground truth data shows which are correlated (Table 10.2).

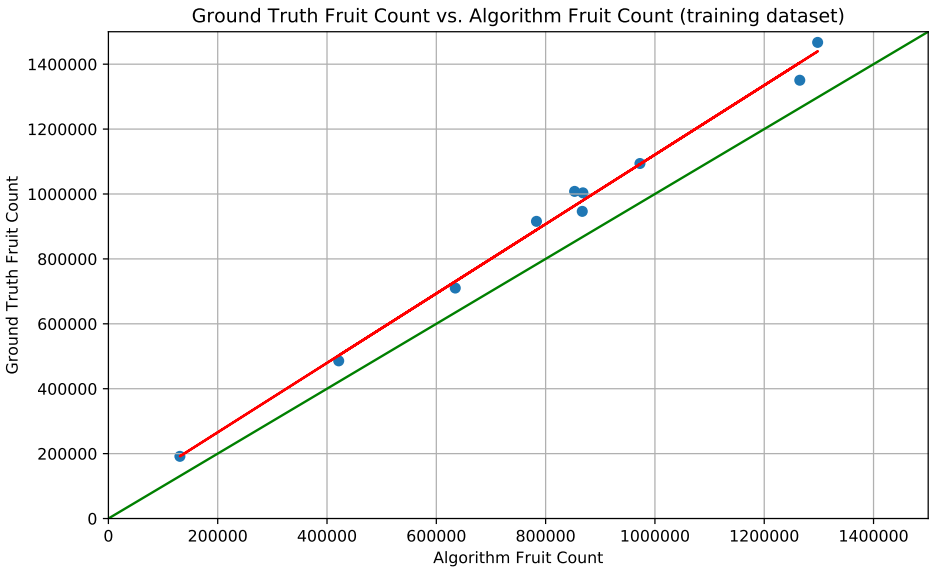


Figure 10.1: The ground truth fruit count vs. algorithm fruit count for the maturity areas in the training dataset. The green line is the ideal line (one to one mapping between algorithm and ground truth fruit counts). The red line is a linear trend-line. The R^2 value is 0.993.

Maturity Area	Fruit Count Error	Correction Factor
A_A	-11.1%	1.12
G_B	**-6.3%**	**1.06**
K_A	-13.5%	1.15
K_B	-11.6%	1.13
K_D	-13.3%	1.15
K_E	-15.4%	1.18
N_A	-8.4%	1.09
S_A	-10.6%	1.11
P_A	**-31.7%**	**1.46**
P_C	-14.5%	1.16
Mean	-11.8%	1.17
Standard Deviation	-6.9%	0.10

Table 10.1: The algorithm fruit count error and correction factor for each of the maturity areas in the training dataset. The maximum and minimum values are shown in bold for each column.

	Avg. Size	**Fruit Count**	**Correction Factor**
Area	0.245	**0.874**	**-0.694**
Avg. Row Width	**0.711**	0.190	0.016
Row End Length	0.333	**0.747**	**-0.681**
Row Side Length	-0.169	0.470	-0.187
Perimeter	-0.013	**0.615**	-0.378
Area : Perimeter	0.270	0.570	**-0.749**
Area : Row End	-0.189	-0.137	0.417
Area : Row Side	0.333	0.455	**-0.722**
Avg. Canopy Density	0.254	-0.274	0.434
Height Stddev	**-0.885**	-0.422	0.084
Avg. Cluster Size	0.140	-0.015	0.150
Across Row Dist.	0.533	-0.024	0.475
Fruit Density	0.450	0.406	-0.334
Alg. Fruit Count	0.465	**0.997**	**-0.746**

Table 10.2: The correlation coefficient (Pearson product-moment coefficient) for all of the predictor variables against the ground truth data for the fruit training dataset. All correlation coefficients larger than 0.6 in magnitude have been highlighted.

Figure 10.2: The average fruit size and average row width have a positive correlation. Modern orchards tend to have wider rows. Other modern orchard practices may lead to larger fruit, causing a positive correlation between row width and fruit size.

The average row width has a positive correlation with average fruit size (correlation coefficient of 0.711). A possible explanation for this correlation is that newer orchards tend to have wider row as orchard layout practices have changed over time. Other practices that are common in newer orchards may encourage larger fruit sizes, causing the correlation between row width and fruit size (Figure 10.2).

The variation in fruit height is correlated with average fruit size. The larger the variation in height, the lower the fruit size (Figure 10.3). This may be explained by better maintained orchards having a flatter canopy (hence lower variation in fruit height) and also having larger fruit.

The two predictor variables that one may expect to be correlated with fruit size are the canopy density and the fruit density. Higher canopy density provides additional leaf area to absorb sunlight and produce larger fruit. A high fruit density would suggest lower fruit sizes as there are more fruit for each plant to support. However, neither of these expected correlations are

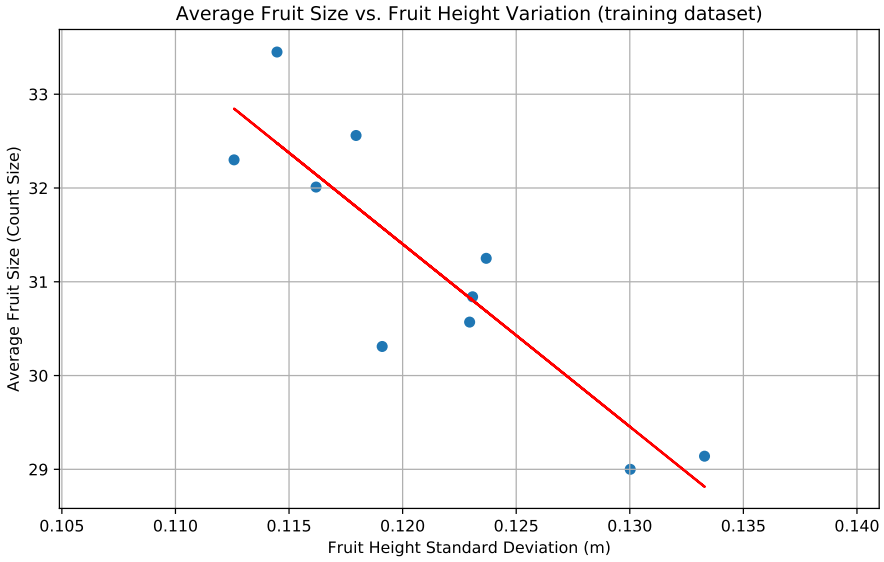


Figure 10.3: The average fruit size and fruit height variation have a negative correlation. A possible explanation for this correlation is that well managed orchards would tend to have both a lower fruit height variation and higher fruit size than less well managed orchards.

demonstrated by the data.

Orchard area is correlated with ground truth fruit count. The larger the orchard is the more fruit that it contains. However, the correlation is not as strong as may be expected, which shows there is significant variation in the productivity of orchards. Orchard productivity is commonly measured by the number of trays of class one fruit per hectare of orchard (referred to as trays per hectare). The trays per hectare measurement is used as it takes into account both the reject rate and fruit size. Looking at the maturity areas in the training dataset, a more than doubling of productivity can be seen between the best and worst producing maturity areas (Table 10.3).

Both row end length and orchard perimeter are correlated with ground truth fruit count. Larger orchards will generally have a higher row end length, perimeter and fruit count. It would follow that row side length would have a similar correlation to fruit count, however, it is a much weaker correlation (correlation coefficient of 0.470 compared to 0.747 and 0.615 for row end length and perimeter respectively).

Maturity Area	Trays per Hectare
A_A	**20398.1**
G_B	**9446.5**
K_A	14877.0
K_B	14212.7
K_D	12767.6
K_E	13028.8
N_A	13452.9
S_A	12182.8
P_A	13216.1
P_C	12092.3
Mean	13567.5
Standard Deviation	2807.4

Table 10.3: The number of trays of class one fruit produced per hectare of orchard area for all maturity areas in the fruit training dataset. The productivity of orchards can vary by more than a factor of two between the best and worst performers. Note, this is ground truth data as opposed to measurements from the yield estimation system.

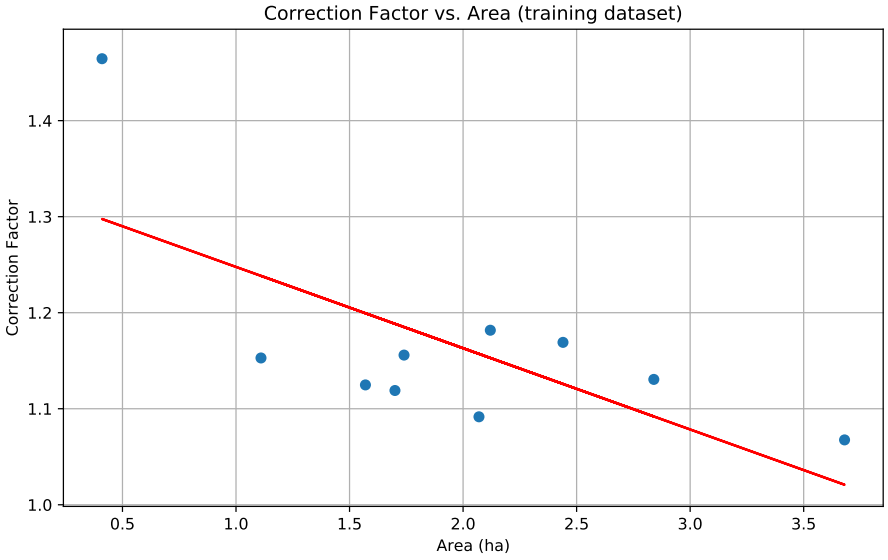


Figure 10.4: Correction factor vs. area for the maturity areas in the training dataset. The larger the maturity area the lower the correction factor. The largest and smallest maturity areas shown may be outliers that are skewing the results.

Orchard area is negatively correlated with correction factor. Saying that in larger orchards, a higher proportion of the fruit is seen than in smaller orchards. Plotting the values shows that this correlation may be significantly influenced by two outlier data points and may not be true in general (Figure 10.4). The row end length, the area to perimeter ratio and the area to row side ratio all have a negative correlation with correction factor, similar to that of the area.

The small sample size of only 10 maturity areas (in the fruit training dataset) makes definitive conclusions about the relationship between variables impractical. A single outlier can significantly change the correlations between variables.

10.2.1 Possible Error In Ground Truth Data

Of the 10 maturity areas in the training dataset, P_A has a significantly higher correction factor than the others (Table 10.1). The correction factor of the

Figure 10.5: An aerial shot of P_A. It consists of two long narrow blocks, both 160 m long by 12.5 m wide. Orchard blocks tend to have a much squarer aspect ratio and much larger area. Image sourced from the Land Information New Zealand (LINZ) Data Service and licensed for reuse under the Creative Commons BY 4.0 licence.

other 9 maturity areas ranges from 1.06 to 1.18 whereas the correction factor for P_A is 1.46. The fruit detection, fruit localisation, SLAM and fruit tracking systems appear to be functioning as they do in all the other maturity areas. This maturity area is the smallest of all those visited for data collection with an area of 0.41 ha. The next smallest maturity area visited is 2.7 times larger at 1.11 ha. P_A also has a very high ratio of side row length to area at 1502 m of side row per ha of area (Figure 10.5). The next highest is 528 m/ha. However, it is part of an orchard that contains multiple maturity areas, which gives a higher likelihood of harvested fruit being attributed to the wrong maturity area either on the orchard or at the packhouse.

The validity of the fruit count for P_A cannot be independently verified. For the purposes of developing the occlusion prediction model, it is assumed that this and all other ground truth data is correct. This doubt demonstrates one of the downsides of not having control of the ground truth data capture process.

10.3 Models

To find the most suitable type of model for yield estimation, multiple models are applied and their performance compared.

10.3.1 Baseline Static Correction Factor Model

The baseline static correction factor model is the reference with which to compare all other models. Only the algorithm fruit count and a static correction factor are used to predict the actual number of fruit in an orchard. The hypothesis proposed at the beginning of this work (Section 1.4) is essentially that a more complicated model using additional predictor variables can out-perform this baseline model.

The baseline correction factor model has the following equation, which is found via a line of best fit between the algorithm fruit counts and the ground truth data, with a forced zero Y-intercept.

$$\text{estimatedTotalFruitCount} = 1.124 * \text{algorithmFruitCount} \tag{10.4}$$

Only the fruit training dataset is used to form the model.

10.3.2 Linear Regression

An ordinary least squares linear regression model is applied using the the `LinearRegression` class from the scikit-learn library [118]. The normalisation feature built into the `LinearRegression` class is used to normalise the predictor variables before fitting the model. The model produces an equation in the following form:

$$\text{estimatedCorrectionFactor} = c_0 x_0 + c_1 x_1 + ... + c_{14} x_{14} + c_{15} \tag{10.5}$$

where x_n is predictor variable n, c_n is coefficient n. Each of the coefficients are calculated via least squares linear regression. All of the predictor variables are used.

A second version of the linear regression model is implemented using a reduced set of predictor variables. The predictor variables used are the five with correlation coefficients greater than 0.6 in magnitude to the correction factor (the highlighted cells in the correction factor column of Table 10.2).

10.3.3 Decision Tree Regression

In a decision tree, a series of decision nodes are connected by branches and eventually terminate in leaf or prediction nodes. The training data is evaluated and split to form the branches, with the goal of minimising an error function. In this case, the error function used is the mean squared error.

The `DecisionTreeRegressor` class from the scikit-learn library [118] is used to form a model. The creation of the decision tree relies on randomness and hence the results are different depending on the random seed used. To find a suitable seed, decision trees are generated with integer seeds from 1 to 1000. The validation dataset is used to evaluate the performance of each decision tree. The model with the lowest mean absolute percentage error (MAPE) is selected as the winning model. The best performer uses a seed of 48 and produced a model that achieved a 3.835% MAPE across the three maturity areas in the validation dataset.

10.3.4 K-Nearest Neighbour

To make a prediction, a K-nearest neighbour (KNN) model finds the examples from the training data that most closely match the new sample. The prediction is then a function of the predicted variable of those closest examples.

The `KNeighborsRegressor` class from the scikit-learn library [118] is used to from the model. The model consists of all of the predictor variables, after they have been normalised. The normalisation step scales all values so they are between 0 and 1. For example, the largest maturity area would have an area of 1 and the smallest, an area of 0.

The model has two meta parameters that need to be tuned, the number of neighbours to look at and the function with which to weigh the contribution from each neighbour. The fruit validation dataset is used for this tuning. Figure 10.6 shows the mean absolute error rate for each combination of meta parameters. The combination of meta parameters with the lowest error rate is 10 neighbours with uniform weighting. However, using these meta parameters

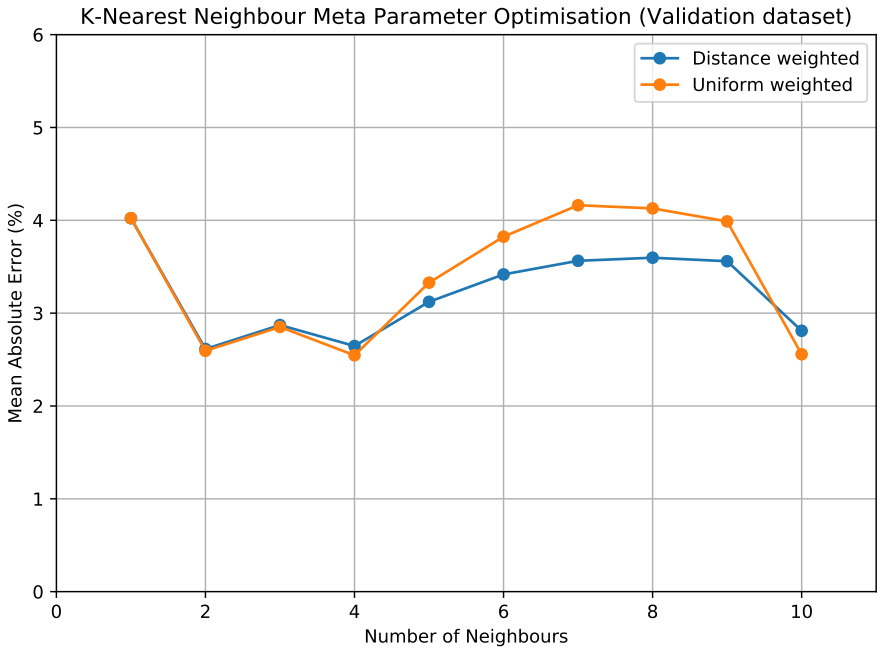


Figure 10.6: The error rate with different combinations of meta parameters for the correction factor K-nearest neighbour model. Using uniform weighting with 10 neighbours gives the lowest error rate.

is the same as using a static correction factor found by taking the mean of all 10 correction factors in the training dataset. The fact that this meta parameter set gives the lowest error rate shows that, at least for the validation dataset, a KNN model is no better than a static correction factor. As using uniform weighting with 10 neighbours does not provide any additional information over using a static correction factor, the meta parameters with the next lowest error rate are used. Which is a uniform weighting with 4 neighbours.

A second version of the KNN model is also implemented, but using a reduced set of predictor variables. The predictor variables used are the five with correlation coefficients greater than 0.6 in magnitude to the correction factor (the highlighted cells in the correction factor column of Table 10.2). Figure 10.7 shows the meta parameter combinations and corresponding error rates for this model. As with the full KNN model, using 10 neighbours with uniform weighting gives the lowest error rate on the validation dataset. So the next best combination is used, which is 5 neighbours with uniform weighting.

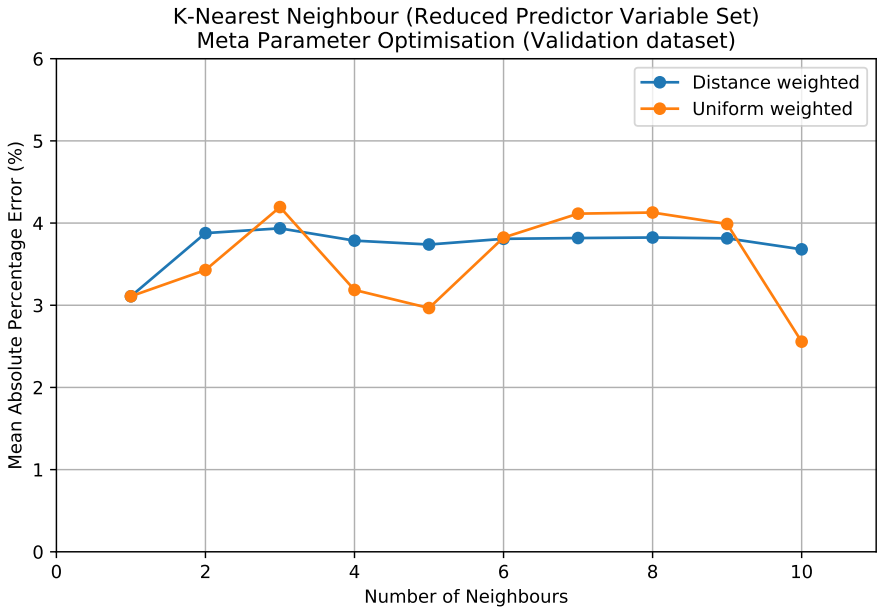


Figure 10.7: The error rate with difference combinations of meta parameters for the correction factor K-nearest neighbour model with reduced predictor variable set. Using uniform weighting with 10 neighbours gives the lowest error rate.

10.4 Performance

The performance of each of the models is compared on the validation dataset and the test dataset (Table 10.4). Comparing the performance of the models on the validation dataset is not a fair comparison as both of the KNN models and the decision tree regressor are tuned using that dataset. Whereas, the other models are not as they do not have meta parameters to optimise. However, due to the low number of samples present in the test dataset (three maturity areas), the validation dataset performance is included to provide a better representation of the overall performance of the models.

The baseline (static correction factor) model is the best performing model on the test dataset with a mean absolute percentage error (MAPE) of 0.63%. However on the validation dataset, performance is much poorer at 4.72% MAPE. This large difference in performance on the two datasets shows the impact of the small sample size. The correction factor decision tree regressor model shows less of a difference in performance between the test and valida-

Model	Validation	Test	Mean
Baseline	4.72%	**0.63%**	2.67%
Decision Tree Regressor	3.84%	2.39%	3.11%
KNN	2.55%	1.63%	**2.09%**
KNN Reduced	2.97%	1.48%	2.22%
Linear Regression	**1.11%**	8.67%	4.89%
Linear Regression Reduced	4.83%	5.57%	5.20%

Table 10.4: The mean average percentage error (MAPE) of each of the models on the validation dataset, the test dataset and the mean of the two. The lowest error rate for each dataset is highlighted.

tion dataset but with worse overall accuracy than the baseline model. Both of the KNN models demonstrate high accuracy on the test dataset with <1.7% MAPE. Their performance on the validation dataset is significantly better than the baseline. The two linear regression models performed the worst overall with MAPE of 8.67% and 5.57% on the test dataset. However, the full linear regression model is the best performer on the validation dataset despite no optimisation based on this dataset.

A significance test is performed to analyse the differences in performance of the models compared to the baseline. The absolute errors for the test dataset are compared using a one-tailed, paired t-test with alpha of 0.05. The results are shown in Table 10.5. The test shows that the differences in performance are not significant.

Due to the small sample size and high variability between per orchard error rates, no single model can be identified as superior to the others. Both the KNN models perform better over the combined validation and test datasets than the baseline model, but they were both optimised using the validation dataset. When looking at only the performance on the test dataset, the baseline model is very impressive. However the error rate is 7.5x higher on the validation dataset, suggesting that the high performance on the validation

Model	T-Test P-Value
Decision Tree Regressor	0.196
KNN	0.198
KNN Reduced	0.116
Linear Regression	0.051
Linear Regression Reduced	0.118

Table 10.5: Significance test for the occlusion models compared to the baseline model.

dataset is not representative of its overall performance. Overall, it cannot be claimed that either of the KNN models is better than the baseline, or vice versa.

Chapter 11

Conclusions

The inaccuracies of the current manual yield estimation system causes significant costs for the New Zealand kiwifruit industry. This is an industry that contributes a large amount to the New Zealand economy, being the country's largest horticultural export. Large growth is projected for the industry with revenue expected to rise from NZ$2.39 billion in the 2017/2018 season to NZ$4.5 billion by 2025 [4]. Labour shortages are already causing issues for the industry, which will only worsen with the projected growth. Increasing yield estimation accuracy and enabling labour saving solutions are two areas that must be addressed to allow progression of the industry.

A new system has been developed for orchard scale kiwifruit yield estimation. The system is based on two pairs of stereo cameras, a lidar and an IMU mounted on an ATV. Data was captured in 16 maturity areas across 34 Ha of kiwifruit orchard. Ground truth harvest data was obtained for each of the maturity areas from a commercial kiwifruit packhouse.

A simultaneous localisation and mapping (SLAM) system is used to localise the data capture system within the context of an orchard block. There are two issues with the SLAM system. The first is the inability of the lidar unit to see through shade cloth that is placed between rows in some orchards. The second is the large magnitude of positional error observed in the system, particularly between passes. Overall, the SLAM system provides accuracy that is adequate

for the rest of the components of the system to identify and track fruit.

Canopy density measurements are taken by detecting the visible sky in images. The system identifies pixels that are both high in brightness and have a high blue component (RGB colour space). In an evaluation across 17 images, the system correctly classified 99.43% of pixels, with a processing time of 5.3 ms per image.

To detect kiwifruit in images, the convolutional neural network Mask R-CNN is used. Mask R-CNN is capable of not only producing bounding boxes around instances of a detected object, but providing masks that identify the pixels that belong to each detected object. Training is conducted using masks for kiwifruit and bounding boxes for calyxes. Across a 20 image testing dataset, the mean average precision at intersection over union of 0.5 is 0.90. The most common cause of errors is false positive calyx detections caused by small holes in leaves and the black plastic clips used to tie canes to wires. Another cause of errors is areas of leaf are being detected as fruit.

Before detected fruit can be localised in 3D, the correspondence of detected fruit in one image of a stereo pair to the other needs to be solved. The correspondence problem is solved with a two step matching process based on geometry constrained search windows. The first step matches fruit where there is no ambiguity in the matching combination. The second step selects a matching combination when there is ambiguity in the correct combination. The ambiguity is resolved by selecting the combination that will result in fruit that most closely conforms to the average fruit height observed in the orchard. An evaluation of 121 image pairs found the correct matches were identified in 99.23% of cases. Only 0.16% of matches were incorrect, with the remaining 0.61% being false positive detections being matched (caused by errors in the detection system, not the matching system). Matched fruit are then triangulated to give their position relative to the cameras.

The imaging configuration used gives significant overlap between images. Therefore, fruit are tracked through multiple images to avoid double counting.

The fruit tracking system corrects for error in the SLAM system by comparing the appearance of fruit in different images. This similarity in appearance is used to weight a kernel correlation algorithm. The kernel correlation algorithm optimises a 3-axis translational transform which corrects the SLAM error. Correspondence of fruit from one image pair to the next is then calculated to identify repeated fruit in a series of images. An evaluation of 792 fruit found that 93.2% were correctly identified in all images in which they were visible within a single pass. The largest cause of errors is partial occlusion causing a difference in appearance of fruit from image to image. Because of the magnitude of the errors in the SLAM position estimates, fruit tracking between adjacent passes was not able to be effectively implemented. Instead, a simplified system that removes fruit from each pass if they are closer to an adjacent pass than the pass they were detected in, is used.

To estimated the occlusion rate based on the predictor variables extracted from the data, a series of occlusion prediction models are created. The baseline model is a simple static correction factor based on the average correction factor needed for the orchards in the training set. Two variations of a K nearest neighbour algorithm, a decision tree regressor and two variations of a linear regression model are tested. On the test dataset, the best performer was the baseline with a mean average percentage error (MAPE) of 0.63%. However, on the validation dataset, performance was much poorer with a MAPE of 4.72%. The KNN models were less accurate on the test dataset, but were more consistent across the test and validation datasets.

The hypothesis set out at the beginning of this work is the following: ‘An occlusion prediction model will produce dynamic correction factors calculated based on measurable characteristics of an orchard. These individualised correction factors will provide more accurate predictions of yield than a static correction factor.’ This hypothesis has been neither proven nor disproven during the course of this work. The small sample size and small differences between the performance of the baseline model and the models under test prevent

definitive conclusions from being made. However, multiple of the measured characteristics of orchards are correlated with the correction factor (Section 10.2), suggesting that a model that outperforms the baseline could be formed. Data over multiple seasons and across a diverse set of orchards is required to know if these correlations are true in general or just for the population sampled.

11.1 Strengths Of The Work

The scale of the work stands out among previous works. A total of 16 maturity areas consisting of over 15 million fruit in 34 Ha of orchard were covered (Section 3.5). Most previous work has focused on estimating the number of fruit on single plants or row of plants. Processing data and collecting ground truth data for plant scale estimation is far less time consuming than doing the same at orchard scale. However, to provide maximum value for the kiwifruit industry, estimating yield on these large scales is required. Utilising pre-existing ground truth data from packhouses as opposed to hand measuring harvests allows this increase in scale. This work is a large step towards achieving highly accurate, automated, orchard scale estimations of yield for kiwifruit. It is the largest fruit yield estimation project ever undertaken, to the best of the authors knowledge.

The fruit detection system implemented in this work is highly accurate (Chapter 6). This is enabled by the very high performance of modern convolutional neural networks (CNNs) such as Mask R-CNN (the network used in this work). The literature already contains many reports of the high performance of CNNs for fruit and other object detection, especially in varying weather and imaging conditions. The success of the kiwifruit detection system is yet another conformation of the suitability of CNNs for fruit detection over conventional image processing techniques used in the past.

The stereo vision system used in this work is able to localise fruit accu-

rately (Chapter 7). Compared to the monocular vision approach of many previous works, the addition of a second camera provides 3D localisation of fruit. Positioning the fruit in 3D aids the fruit tracking system, increasing overall accuracy. Full 3D tracking of fruit also allows more granular information to be provided to the grower, increasing the overall utility of the system.

The calyx comparison system employed in this work is both highly accurate and fast (6.4% error rate, 122 µs per comparison, see Section 8.5). These qualities make it well suited to aiding a fruit tracking system. Although the deficiencies of the SLAM system prevented a full fruit tracking system from being implemented, the calyx comparison system is a highly valuable addition to fruit tracking. Future improvements to the stereo localisation system (Section 12.3.2) will further increase the accuracy of the calyx comparison system and could allow further optimisations to reduce calyx comparison processing time.

The accuracy of the yield estimation system presented in this work is very high with the baseline model achieving 0.63% mean absolute percentage error on the test dataset (Section 10). Compared to the other fruit yield estimation systems presented in the literature, this result is very good. The thorough approach of both localising fruit in 3D and tracking tracking them through multiple overlapping images to reduce the effect of occlusion is responsible for this high performance.

11.2 Weaknesses Of The Work

As the yield estimations conducted in this work are orchard scale rather than plant scale, capturing a large enough dataset to draw strong conclusions is very resource intensive (Chapter 10). Efforts were made to capture as much data as possible, but with only 16 maturity areas covered, there is simply not enough data to conclusively quantify the performance of the system. However, the system has shown the potential to be of use to the kiwifruit industry. The

commercial partners associated with this work intend to continue investment in the system with the aim to commercialise it in the coming years.

The biggest weakness of the yield estimation system is the performance of the SLAM system (Chapter 4). The positional errors introduced by the SLAM system (particularly between passes) are too great for the fruit tracking system to overcome. Also the lidar units lack of ability to see through shade cloth placed between orchard rows limits the practical utility of the current system. The author believes both of these issues can be overcome with relatively small changes to the hardware used on the data capture system (Section 12.2).

By using ground truth data obtained from packhouses, all control of the quality of that data is lost (Section 3.6). The loss of control is highlighted by the potential error in the ground truth data discussed in Section 10.2.1. However, it is this reliance on outside parties that allows the very large scale of this work. Perhaps a hybrid approach could allow both the large scale and high confidence in the quality of the data. The labourers harvesting the fruit could be supervised by researchers, who verify that the fruit is correctly labelled, transported and fed into the packhouse system. An alternative approach is to increase the number of samples (more orchards) so that errors with any one orchard, do not have a large influence on the overall system.

All of the orchards included in this study are located in a small geographic region around Tauranga, New Zealand (Section 3.5). The orchards are also all managed by the same orchard management company (Gro Plus Ltd.). These orchards were used due to the ease of access and logistical constraints around capturing data. Finally, all data used is from the same growing season. The resulting lack of diversity in the dataset may have introduced bias and may result in the conclusions not being true for the wider kiwifruit industry, or subsequent growing seasons.

More details on the weaknesses of this work and how they can be fixed/mitigated are discussed in Chapter 12.

Chapter 12

Future Work

The following section is intended as a guide for the direction of future research in this area. It is based on the experience of the author while designing and developing the yield estimation system. Some of the suggestions apply only to the yield estimation system presented throughout this work, while others could be applied to other systems and/or crops.

12.1 Data

The yield estimation system presented requires more data to properly verify its performance. That data should include as much diversity as possible. Orchards from different geographical regions, managed by different orchard managers, over multiple growing seasons, grown with different practices (such as organic) should all be included. More data should also increase the accuracy and generalisability of the yield estimation model as the current model is based on only 10 maturity areas.

Currently, the system has only been trained and applied to green kiwifruit. Extending/adapting the system to provide yield estimation for gold fruit would significantly increase the utility of the system. The New Zealand gold crop is expected to more than double in volume between 2018 and 2027, while the green crop is expect to decline by a quarter over the same period [11]. Adding gold fruit would require either retraining of the detection and yield prediction

models, or separate gold kiwifruit specific models to be built and trained.

Data was captured before flower opening as part of this work. However, the data was not used due to time constraints. Adding the capability to count buds and/or flowers would increase the utility of the system as it would provide more information to growers and potentially allow targeted bud/flower thinning (Section 12.4.4). Both the detection model and the calyx comparison system would need to be retrained/reworked to enable accurate bud/flower counting.

Adding more sources of data to the yield estimation model could increase accuracy. The following is a list of data sources that could be added/tracked to gain further insight into orchard performance:

- Weather data such as rainfall, wind speed, sunlight, temperature, humidity, carbon dioxide concentration etc.
- Previous seasons yield
- Plant spacial distribution
- Plant male to female ratio
- Pollination method and application rate
- Soil data such as moisture content, nutrient levels, density etc.
- Spray and fertiliser applications
- Plant information such as age, rootstock etc.
- Plant training characteristics such as cane spacing, cane diameter, cane length etc.
- Fruit information such as dry matter, sugar content etc.

Increasing the number and diversity of data channels tracked is a big step towards being able to apply machine learning to optimise growing practices.

12.2 SLAM

The weakest component of the yield estimation system presented is the SLAM system. There are several aspects that could be changed to improve the quality of SLAM maps produced.

The M8-1 (Quanergy, Sunnyvale, California, USA) lidar that is used, has only 8 layers. Other modern lidar units (such as those from Velodyne and Ouster) feature 16, 32 or 64 layers. The increased resolution would provide more data to the SLAM algorithm, which should result in higher quality SLAM output.

The lidar on the data capture system is mounted such that it has a 212° horizontal field of view. In orchards that have small headlands bordered by shelter belts, turning at the end of a row can result in the lidar's field of view containing little more than the shelter belt. The shelter belt looks essentially like a large vertical plane, which provides no distinguishable features for scan matching. This has been observed to cause errors in positioning on multiple occasions. Therefore, the lidar should be mounted such that it has an increased horizontal field of view. If the lidar cannot be mounted in a position where it has a larger horizontal field of view, adding a second lidar to the rear of the vehicle would also solve the problem. This will increase the hardware cost significantly though.

The lidar used on the data capture system is not suitable for use in orchards with shade cloth between the rows (Section 4.2.2). Shade cloth is common in gold orchards more so than green orchards. As discussed in Section 12.1, gold fruit should be a target for future work in this area, therefore the shade cloth issue will need to be solved. The preliminary experiment discussed in Section 4.2.2 shows that the VLP16 (Velodyne LiDAR, San Jose, California, USA) lidar is better suited to seeing through shade cloth than the M8-1. A further preliminary study (not discussed in this work) showed that an OS-1 (Ouster, San Francisco, California, USA) lidar may also be suited to seeing through shade cloth. However, these claims about both the VLP-16 and OS-1 require

further verification and testing in a wider variety of orchard configurations. For commercial deployment of a yield estimation system that utilises lidar based SLAM, solving the shade cloth issue is of high importance.

The data capture system used provided no odometry feedback for the SLAM system. Adding some form of odometry could significantly improve the quality of SLAM output as it would provide an additional data source to the lidar. The odometry could be added via encoders on the vehicle wheels, visual odometry using a camera pointed at the ground, or by using the upwards facing cameras to perform visual odometry on the canopy. Wheel encoders would be the most reliable form of feedback, but require additional hardware.

The ATV used as the base of the data capture system has a relatively small footprint. It also has soft suspension with large travel. These characteristics result in a platform that is susceptible to significant roll and pitch due to uneven terrain and the rider shifting their bodyweight to avoid low hanging branches. Using a vehicle with a larger footprint and stiffer suspension would decrease the magnitude of pitch and roll experienced by the lidar, improving SLAM performance.

The SLAM system (Cartographer) is configured to run in 2D mode. This mode assumes that the ground is a flat plane, which is not true in kiwifruit orchards. Cartographer can also be run in 3D mode. Running in 3D mode would take into account the rolling and pitching of the vehicle as well as the contours of the ground through the orchard. This change would incur a significant computational cost but could improve quality of SLAM output.

Improvements to the accuracy of the SLAM system will have flow on effects to the other components of the yield estimation system. For example, the fruit tracking system should be both more accurate and require less computation time if the initial estimates of fruit positions are more accurate. If large enough improvements are made to the accuracy of the SLAM system, the fruit tracking system could be extended to operate between passes. Adding inter-pass fruit tracking would reduce the number of double counted and re-

jected fruit, improving the overall accuracy of the system. Visualisations of the orchards fruit would also benefit from the added accuracy as they would be more representative of the actual orchards. This would in turn increase the confidence in the system of growers and managers.

12.3 Incremental Improvements

12.3.1 Vehicle

Ideally, the yield estimation system would not be based on an ATV ridden by a human. It would be based on an autonomous vehicle such as the modular autonomous vehicle for kiwifruit orchards presented by Jones et. al [75]. Removing the human would decrease health and safety risks which have become a growing concern in the industry after the death of an orchard worker in 2016 [123]. Orchard coverage efficiency could also be increased by having one worker deploy multiple autonomous vehicles.

12.3.2 Stereo Localisation

The accuracy of the stereo localisation system could be improved in multiple ways. The stereo matching system (Chapter 7) could be modified to use the calyx comparison technique described in Section 8.5. This addition could increase the already high correct matching rate of 99.23%.

The stereo triangulation system currently uses the centre of the calyx bounding box (from the detection system) in each image. Any errors in the bounding box positions result in errors in the triangulated position. These errors could be reduced by comparing the calyxes in both images to find a position of best match, as described by Scarfe [13]. To decrease computational load, the comparison could be implemented using the binary technique used in the calyx comparison system (Section 8.5). Improving the stereo localisation system in this way would likely reduce the magnitude of the stereo localisation error, which is one of the three sources of error the fruit tracking system

is designed to correct. The flow on effect would be fewer errors in the fruit tracking system, and hence more accurate counts of the fruit seen in images.

The search window method used in the stereo localisation system will fail to match fruit that are too low or high in the canopy, as seen in Figure 7.8. To solve this the size of fruit in the images could be used to bias the position of the search window. For example, fruit that are large in the image (measured by the size of the fruit mask produced by the detection system) would have their search windows shifted towards a position representing a lower fruit. This biasing could both allow fruit outside of the set geometric bounds to be correctly matched, and reduce ambiguity in some matches. Another variation of this technique is to bias the positions of search windows if the fruit are out of focus in the image. Fruit that are very low or high tend to be out of focus and blurry in the image as they are outside of the depth of field of the camera.

To resolve ambiguity in matching combinations, the stereo localisation system uses the mean height of all previously seen fruit in the orchard block. This leads to errors when fruit are far from the mean height. Figure 8.36 shows that fruit can vary significantly in height across a block, but fruit that are near each other tend to be approximately the same height as each other. Therefore, rather than using a global mean height, a localised mean should be used. A well tuned localised mean would likely produce fewer erroneous matches that the global mean. In addition to using a localised mean, the height of a fruit could be estimated based on the proximity of the fruit to beams and leaders, by adding beam and leader detection to the detection system. Fruit near beams and leaders tend to be higher than fruit it the middle of bays, as shown in Figure 8.36. An estimated fruit height based on proximity to beams and leaders could be used for both biasing of the search window and resolving ambiguous matching combinations.

12.3.3 Row Detection

The yield estimation system presented uses annotations provided by the operator at the time of data capture to identify rows. These annotations both inform the system of which row and pass is being conducted, and if the system is currently within a row or not. Having the operator provide these annotations is not a sustainable practice as it is easy for the operator to make an error. Operators may also be replaced by autonomous navigation systems in future implementations of the system (Section 12.3.1). Therefore, automatic identification of the presence of a row, which row it is and the current pass is required. This could be done based on the map produced by the SLAM system. The captured images could also be used to identify the start and end of each row by detecting the presence of the canopy.

12.3.4 Detection

The detection system demonstrated impressive performance with a mean average precision of 0.90 (Chapter 6). However, a relatively small number of images (69) were used for training due to the time consuming nature of producing annotations. Using more images for training would likely improve the accuracy of the detection system.

The only network tested for this work is Mask R-CNN with Resnet101 as the backbone. Other networks, such as YoloV3 or RetinaNet or alternative backbones for Mask R-CNN may offer higher performance for kiwifruit detection.

Two of the main causes of false positive detections are the black clips used to tie canes to wires and small holes in leaves (Figure 6.8). In most of these cases, there is a calyx detected, but no fruit. Rejecting detected calyxes if they are not contained within a detected fruit could eliminate many of these false positive detections. However, it may also reject some true positives, so accuracy would need to be carefully evaluated.

In the detection system, Mask R-CNN is configured to scale the input

images from their original resolution of 1920×1200 to 1024×1024 (Section 6.1.1). Detection accuracy may be improved by modifying the network to accept full resolution images. Alternatively, images could be split into parts and inference performed on each, before recombining the image. However, both of these adjustments would increase computation time significantly, therefore, any improvement in accuracy would need to be evaluated against the increase in computation time.

12.3.5 Calyx Comparison

The calyx comparison system uses a hyperbolic tangent function (Equation 8.3) to normalise the raw differences between binary calyx images (Section 8.6.1). The parameters of this hyperbolic tangent are tuned by hand and are thus likely not optimal. The use of the hyperbolic tangent is also likely sub-optimal. Ideally a function that closely mimics the probability that any given raw calyx comparison score is obtained from two views of the same calyxes or not should be used.

When comparing two calyxes, the current system will compare every available combination of calyx images. For example, if a newly seen calyx is being compared with a calyx that has been seen in 3 previous image pairs, there will be 12 total comparisons made (2 images of the new calyx, 6 images of the previously seen calyx). The score used is the lowest of the 12 raw scores, which will then be normalised using the hyperbolic tangent function. As only the lowest raw score is used, one low outlier can result in the calyxes falsely having a much higher similarity score than they should. Therefore, more than just the lowest score should be taken into account. Either a weighted average of all of the scores or the mean of the lowest 50% (or another proportion) of the scores could be a better measure.

FaceNet was tested as a potential method for performing calyx comparisons (Section 8.5.1). The results were promising, however flaws discovered in the testing methodology, removed all confidence in those results. FaceNet (or

another facial recognition system) could prove to be a high performing option for identifying different views of the same fruit. However, further evaluation to quantify performance on kiwifruit as opposed to human faces is required. It is likely that to achieve high accuracy with FaceNet, a larger training dataset is needed. The barrier to creating a larger dataset is the time required to identify the same fruit in multiple images. The fruit tracking system can be used to identify the same fruit across multiple images, with a human to verify the results. This approach should significantly reduce the time required to form a large dataset for training and evaluation.

With the current calyx comparison system, the scale of calyxes is not taken into account. A calyx that is in the centre of an image, and hence at its closest point to the camera (assuming flat ground), can be compared to the same calyx, but in the corner of another image, where it is significantly further from the camera. This is likely to produce a lower similarity score than if the calyxes appeared the same size in both images. The calyx images could be scaled to correct for this mismatch, which could improve the accuracy of the system.

12.3.6 Model Parameters

The fruit height standard deviation is intended to measure the variation in fruit height (Section 9.2). The theory that high fruit hight variation would result in higher occlusion rates makes intuitive sense. Very high fruit are likely to be occluded by leaves, lower fruit or branches, and very low fruit are likely to not be localised by the stereo localisation system. However, when a canopy has fruit growing in it, there is significant sag in the centre of bays as they are the least supported sections of the canopy. Areas near beams and leaders are well supported and are hence higher. Therefore, the fruit height standard deviation becomes a measure of canopy sag. To quantify the intended variable, the height of fruit should be compared to the height of other fruit in the immediate vicinity, not all fruit.

The measure of fruit distribution across a row does not take into account

varying row widths (Section 9.4). It takes the position of each fruit within its row and compares the resulting distribution to a normal distribution with the same mean and standard deviation. Instead, this should be calculated on a per row basis and then the mean of all the rows taken. Alternatively, the position of each fruit across a row could be divided by the width of the row it is in to normalise it. That way, the whole orchard block can be calculated as one. The result of either of these methods would be a better measure of the distribution of fruit across the row in orchards with varying row width.

12.4 New Features

12.4.1 Fruit Size Estimation

Fruit yield is typically measured in trays of fruit rather than number of fruit. This is because the trays measurement accounts for the size of fruit. The yield estimation system presented predicts only the number of fruit, and not the size. Mask-RCNN is used for the detection system as it provides not only bounding boxes, but masks for detected objects. The training data consists of masks for fruit and bounding boxes for calyxes. The intention of using Mark-RCNN with fruit masks, is to add fruit size measurements to enable prediction of average fruit size. This fruit size estimate would allow estimation of the number of trays. Additional insight could also be gained into the variability across an orchard by producing spacial plots of fruit size along with fruit density.

Addition of fruit size estimation is not a trivial task due to partial occlusion. Accurate estimation of the size of every fruit is likely not possible. Only taking size estimates from fruit with no or low levels of occlusion may be a viable solution. Occlusion could be estimated by the convexity of the fruit mask outline. For example, a fruit with a fully convex mask outline is likely unoccluded, whereas a mask outline that has multiple concave segments is likely partially occluded (Figure 12.1). However, the pose of the fruit relative to the camera must also be taken into account as kiwifruit are shaped more like

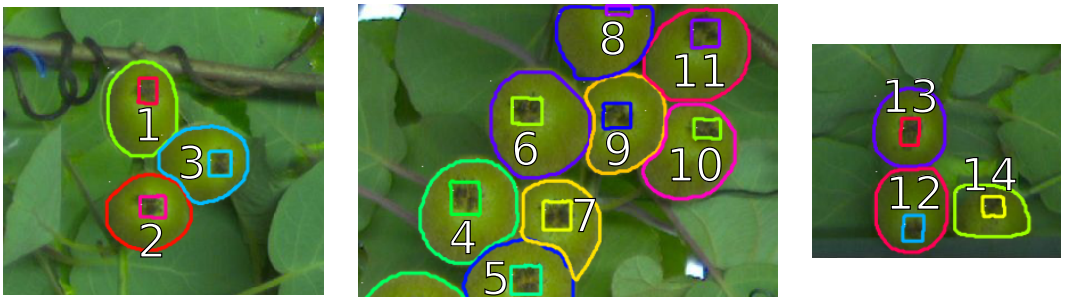


Figure 12.1: Examples of detected fruit masks. Fruit 1 and 2 have a fully convex mask outline, and are unoccluded. Fruit 3, 4, 7, 9 and 10 have partially concave mask outlines and are partially occluded. However, fruit 14 has a fully convex mask outline but is partially occluded. The calyx of fruit 1 is positioned above the centre of the fruit mask, showing that the fruit is not being viewed along its major axis. The calyx of fruit 2 is close to the centre of the fruit mask, showing that it is being view from close to its major axis.

an ellipsoid than a sphere. This results in the fruit appearing to be a different size depending on its pose relative to the camera. There are multiple methods that could be used to estimate the pose of the fruit relative to the camera. The first of which is the position of the fruit within the image. Generally, fruit are oriented so the major axis of the fruit is vertical, however there are many exceptions to this, particularly in clusters where fruit are in contact with each other. Fruit that are in the centre of the image are likely being viewed along the major axis of the fruit. Whereas fruit near the edges of the image are likely being viewed from an oblique angle. Another method is to analyse the position of the calyx within the fruit mask. If the calyx is centred within the fruit mask, the fruit is likely being viewed along its major axis (Figure 12.1).

To ensure consistency of size estimates the best image of each fruit should be used. The vast majority of fruit are seen in multiple pairs of images. A score estimating the proximity of the camera to the major axis of the fruit could be developed. This score could use factors such as the convexity of the fruit mask, the position of the calyx within the fruit mask and the position of the fruit within the image. Then the fruit size could be estimated from the view with the highest score, and hence the best view of the fruit.

Once the best view of a fruit is selected, a circle or ellipse can be fitted to

the fruit mask outline. Alternatively, the area of the mask can be measured, however this would not account for partial occlusions like a circle/ellipse fitting method would. The distance of the fruit from the camera (obtained from stereo vision as discussed in Chapter 7) can then be used to convert the circle/ellipse/mask area from image area to real world area. This will give an estimate of the actual size of the fruit.

An alternative to hand engineering an algorithm for finding the best view of a fruit may be to train a neural network. For each detected fruit, a section of the image around the fruit could be cropped out (maybe 150×150 pixels). This could then be fed into a neural network that estimates how occluded the fruit is. The pose of the fruit relative to the camera could also be estimated in this way, either with the same network or a second neural network.

There may be also be patterns in fruit size that could be exploited. For example, it may be true that there is very little variation in size between fruit in the same cluster. That would mean that only the least occluded fruit in a cluster would need to be measured, all others could be assumed to be very similar in size.

12.4.2 Psa Detection

Pseudomonas syringae pv. actinidiae (Psa) is a bacteria that is harmful to kiwifruit plants. It was found in New Zealand orchards in 2010 and spread quickly throughout the North Island, causing large amounts of damage to crops. It is still present in orchards nine years later. The effects of Psa have largely been mitigated by spraying of bactericides [124] and the introduction of the Psa resilient Gold3 variety of fruit. The effects of Psa on kiwifruit plants can be seen as brown spots on the leaves as seen in Figure 12.2. Detecting the brown spots in the yield estimation images to produce maps highlighting the most effected areas of orchards may provide additional value to growers. It could also be used to quantify the levels of Psa in orchards and provide more granular information on the effect of mitigation techniques.

Figure 12.2: An example of the brown spotting on leaves caused by Psa.

12.4.3 Fruit to Plant Correlation

The pergola growing system has canes coming from both sides of the row, usually in an alternating pattern (Figure 1.5). This arrangement means that a fruit cannot be correlated to a specific plant based solely on its location in the orchard. The best estimate would be a pair of plants. The addition of cane detection and tracking in addition to fruit detection and tracking could allow fruit to be attributed to specific plants. Knowing which fruit belong to which plant allows quantification of the performance of each plant. This increased granularity of information could allow for changes in the way the plants are managed.

12.4.4 Targeted Labour

Thinning of buds, flowers and fruit is often conducted in orchards to reduce the number of fruit and allow the remaining fruit to grow to full size. Thinning is a labour intensive task accounting for 27% of total on-orchard labour requirements in the 2017/18 season [11]. Variation within orchards means that some areas require significant thinning while other may require no thinning. The current method of thinning is to send orchard workers into an orchard and instruct them to remove a fixed percentage of the buds/flowers/fruit. A modified yield estimation system could provide a map of the orchard with each bay (small area approximately 5 m x 5 m) marked with the number of buds/flowers/fruit to remove. This would avoid excess thinning in areas that do not require thinning and ensure dense areas are adequately thinned. The result would be saved labour and an increase in crop uniformity, both decreasing costs and increasing yield.

If cane detection is added to the system, cane diameter estimates could be made and used to better inform thinning decisions. Thicker canes can carry more nutrients and thus support more fruit. Knowing cane diameters could also inform other orchard management decisions.

12.4.5 Fruit Maturity Prediction

Currently, before an orchard can be harvested, a sample of fruit must meet a set of maturity criteria. The criteria includes the sugar content, dry matter content, seed colour, flesh colour, firmness and more [125]. Generally, fruit samples are collected by an orchard worker and the tests are conducted over night so the results are ready for the next morning. Often orchard managers are waiting on maturity testing results so that harvesting can be conducted as soon as possible. Harvesting time is the busiest time of year for the orchard workers, managers and packhouses. Logistics have to be coordinated between parties to ensure workers and equipment are in the right place at the right time. This is further complicated by weather as kiwifruit cannot be picked when wet.

If fruit maturity data is included in the yield prediction system, predictions of fruit maturity could be produced. Better predictions of fruit maturity could allow for more efficient planning of logistics, reducing labour demand. Particularly early in the season when there are monetary incentives to harvest fruit early.

References

[1] Plant and Food Research New Zealand, "Fresh Facts 2016." https://www.freshfacts.co.nz/files/freshfacts-2016.pdf [Online, Accessed 2019-09-24], 2017.

[2] Plant and Food Research New Zealand, "Fresh Facts 2017." https://www.freshfacts.co.nz/files/freshfacts-2017.pdf [Online, Accessed 2019-09-24], 2018.

[3] Plant and Food Research New Zealand, "Fresh Facts 2018." https://www.freshfacts.co.nz/files/freshfacts-2018.pdf [Online, Accessed 2019-09-24], 2019.

[4] Zespri, "Zespri Annual Report 2017/18." https://www.zespri.com/ZespriInvestorPublications/Annual-Report-2017-18.pdf [Online, Accessed 2019-09-28], 2018.

[5] NZKGI, "2018 Kiwifruit Book." https://www.nzkgi.org.nz/wp-content/uploads/2018/12/2018-Kiwifruit-Book.pdf [Online, Accessed 2019-09-23], 2018.

[6] Zespri, "Zespri Annual Report 2016/17." https://www.zespri.com/ZespriInvestorPublications/Annual-Report-2016-17.pdf [Online, Accessed 2019-09-24], 2017.

[7] Trevelyan's, "Kiwifruit Packing Agreement 2019." https://trevelyan.co.nz/wp-content/uploads/2019/06/2019-Packing-Agreement.pdf [Online, Accessed 2019-09-29], 2019.

[8] G. Hutching, "Zespri minimises fungus threat to other exports." https://www.stuff.co.nz/business/farming/83013296/zespri-minimises-fungus-threat-to-other-exports [Online, Accessed 2019-09-20], aug 2016.

[9] Trevelyan's, "Kiwifruit News." https://trevelyan.co.nz/wp-content/uploads/2018/10/TrevelyanSept2018KiwifruitNews.pdf [Online, Accessed 2019-09-26], 2018.

[10] D. Picken, *Inside the Bay of Plenty's kiwifruit labour shortage*. Tauranga, New Zealand: Bay of Plenty Times, 2018.

[11] NZKGI, "New Zealand Kiwifruit Labour Shortage." https://nzkgi.org.nz/wp-content/uploads/2018/07/NZ-Kiwifruit-Labour-Shortage-July-2018.pdf [Online, Accessed 2019-09-24], 2018.

[12] A. J. Scarfe, R. C. Flemmer, H. H. Bakker, and C. L. Flemmer, "Development of an autonomous kiwifruit picking robot," in *ICARA 2009 - Proceedings of the 4th International Conference on Autonomous Robots and Agents*, (Wellington, New Zealand), pp. 380–384, 2009.

[13] A. J. Scarfe, *Development of an Autonomous Kiwifruit Harvester*. Doctoral thesis, Massey University, 2012.

[14] L. Mu, Y. Liu, Y. Cui, H. Liu, L. Chen, L. Fu, and Y. Gejima, "Design of End-effector for Kiwifruit Harvesting Robot Experiment," in *Proceedings of the 2017 ASABE Annual International Meeting*, (Spokane, Washington, USA), pp. 1–8, American Society of Agricultural and Biological Engineers, 2017.

[15] H. A. Williams, M. H. Jones, M. Nejati, M. J. Seabright, J. Bell, N. D. Penhall, J. J. Barnett, M. D. Duke, A. J. Scarfe, H. S. Ahn, J. Lim, and B. A. MacDonald, "Robotic kiwifruit harvesting using machine vision, convolutional neural networks, and robotic arms," *Biosystems Engineering*, vol. 181, pp. 140–156, may 2019.

[16] H. Williams, C. Ting, M. Nejati, M. H. Jones, N. Penhall, J. Lim, M. Seabright, J. Bell, H. S. Ahn, A. Scarfe, M. Duke, and B. MacDonald, "Improvements to and largescale evaluation of a robotic kiwifruit harvester," *Journal of Field Robotics*, pp. 1–15, jul 2019.

[17] L. Mu, G. Cui, Y. Liu, Y. Cui, L. Fu, and Y. Gejima, "Design and simulation of an integrated end-effector for picking kiwifruit by robot," *Information Processing in Agriculture*, may 2019.

[18] A. D. Aggelopoulou, D. Bochtis, S. Fountas, K. C. Swain, T. A. Gemtos, and G. D. Nanos, "Yield prediction in apple orchards based on image processing," *Precision Agriculture*, vol. 12, no. 3, pp. 448–456, 2011.

[19] Q. Wang, S. Nuske, M. Bergerman, and S. Singh, "Automated Crop Yield Estimation for Apple Orchards," in *Proceesings of The 13th International Symposium on Experimental Robotics*, (Québec City, Canada), pp. 754–758, Springer, 2012.

[20] Q. Wang, S. Nuske, M. Bergerman, and S. Singh, "Design of Crop Yield Estimation System for Apple Orchards Using Computer Vision," *2012 ASABE Annual International Meeting*, vol. 7004, no. 12, 2012.

[21] H. Cheng, L. Damerow, Y. Sun, and M. Blanke, "Early Yield Prediction Using Image Analysis of Apple Fruit and Tree Canopy Features with Neural Networks," *Journal of Imaging*, vol. 3, no. 1, p. 6, 2017.

[22] S. Bargoti and J. Underwood, "Deep fruit detection in orchards," in *Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA)*, (Singapore), pp. 3626–3633, IEEE, may 2017.

[23] R. Zhou, L. Damerow, and M. M. Blanke, "Recognition Algorithms for Detection of Apple Fruit in an Orchard for early yield Prediction," in *The 11th International Conference on Precision Agriculture (ISPA)*, vol. 13, (Indianapolis, Indiana, USA), pp. 568–580, MDPI, 2012.

[24] S. W. Chen, S. S. Shivakumar, S. Dcunha, J. Das, E. Okon, C. Qu, C. J. Taylor, and V. Kumar, "Counting Apples and Oranges With Deep Learning: A Data-Driven Approach," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 781–788, 2017.

[25] R. Črtomir, C. Urška, T. Stanislav, S. Denis, P. Karmen, M. Pavlovič, and V. Marjan, "Application of Neural Networks and Image Visualization for Early Forecast of Apple Yield," *Erwerbs-Obstbau*, vol. 54, pp. 69–76, jun 2012.

[26] J. Qian, B. Xing, X. Wu, M. Chen, and Y. Wang, "A smartphone-based apple yield estimation application using imaging features and the ANN method in mature period," *Scientia Agricola*, vol. 75, pp. 273–280, aug 2018.

[27] D. Stajnko, M. Lakota, and M. Hočevar, "Estimation of number and diameter of apple fruits in an orchard during the growing season by thermal imaging," *Computers and Electronics in Agriculture*, vol. 42, no. 1, pp. 31–42, 2004.

[28] A. Gongal, A. Silwal, S. Amatya, M. Karkee, Q. Zhang, and K. Lewis, "Apple crop-load estimation with over-the-row machine vision system," *Computers and Electronics in Agriculture*, vol. 120, pp. 26–35, 2016.

[29] S. Nuske, S. Achar, T. Bates, S. Narasimhan, and S. Singh, "Yield Estimation in Vineyards by Visual Grape Detection," in *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (San Francisco, CA, USA), pp. 2352–2358, IEEE, 2011.

[30] M. P. Diago, C. Correa, B. Millán, P. Barreiro, C. Valero, and J. Tardaguila, "Grapevine yield and leaf area estimation using supervised classification methodology on RGB images taken under field conditions," *Sensors (Switzerland)*, vol. 12, no. 12, pp. 16988–17006, 2012.

[31] B. Millan, S. Velasco-Forero, A. Aquino, and J. Tardaguila, "On-the-Go Grapevine Yield Estimation Using Image Analysis and Boolean Model," *Journal of Sensors*, vol. 2018, pp. 1–14, dec 2018.

[32] A. B. Payne, K. B. Walsh, P. P. Subedi, and D. Jarvis, "Estimation of mango crop yield using image analysis - Segmentation method," *Computers and Electronics in Agriculture*, vol. 91, pp. 57–64, 2013.

[33] A. Payne, K. Walsh, P. Subedi, and D. Jarvis, "Estimating mango crop yield using image analysis using fruit at 'stone hardening' stage and night time imaging," *Computers and Electronics in Agriculture*, vol. 100, pp. 160–167, 2014.

[34] W. S. Qureshi, A. Payne, K. B. Walsh, R. Linker, O. Cohen, and M. N. Dailey, "Machine vision for counting fruit on mango tree canopies," *Precision Agriculture*, vol. 18, no. 2, pp. 224–244, 2017.

[35] A. Koirala, K. B. Walsh, Z. Wang, and C. McCarthy, "Deep learning for real-time fruit detection and orchard fruit load estimation: benchmarking of MangoYOLO'," *Precision Agriculture*, vol. February, pp. 1–29, feb 2019.

[36] M. Stein, S. Bargoti, and J. Underwood, "Image Based Mango Fruit Detection, Localisation and Yield Estimation Using Multiple View Geometry," *Sensors*, vol. 16, no. 11, p. 1915, 2016.

[37] Z. Wang, K. Walsh, and B. Verma, "On-Tree Mango Fruit Size Estimation Using RGB-D Images," *Sensors*, vol. 17, no. 12, p. 2738, 2017.

[38] P. Wijethunga, S. Samarasinghe, D. Kulasiri, and I. Woodhead, "Digital image analysis based automated kiwifruit counting technique," in *2008 23rd International Conference Image and Vision Computing New Zealand*, (Christchurch, New Zealand), pp. 1–6, IEEE, nov 2008.

[39] I. Sa, Z. Ge, F. Dayoub, B. Upcroft, T. Perez, and C. McCool, "DeepFruits: A Fruit Detection System Using Deep Neural Networks," *Sensors*, vol. 16, no. 8, p. 1222, 2016.

[40] C. McCool, I. Sa, F. Dayoub, C. Lehnert, T. Perez, and B. Upcroft, "Visual detection of occluded crop: For automated harvesting," in *2016*

IEEE International Conference on Robotics and Automation (ICRA), no. May, (Stockholm, Sweden), pp. 2506–2512, IEEE, may 2016.

[41] J. P. Underwood, C. Hung, B. Whelan, and S. Sukkarieh, "Mapping almond orchard canopy volume, flowers, fruit and yield using lidar and vision sensors," *Computers and Electronics in Agriculture*, vol. 130, pp. 83–96, 2016.

[42] C. C. Tran, D. T. Nguyen, H. D. Le, Q. B. Truong, and Q. D. Truong, "Automatic dragon fruit counting using adaptive thresholds for image segmentation and shape analysis," in *Proceedings of the 2017 4th NAFOSTED Conference on Information and Computer Science, NICS 2017*, (Hanoi, Vietnam), pp. 132–137, 2017.

[43] P. Wijethunga, S. Samarasinghe, D. Kulasiri, and I. Woodhead, "Towards a generalized colour image segmentation for kiwifruit detection," in *24th International Conference Image and Vision Computing New Zealand, IVCNZ 2009 - Conference Proceedings*, (Wellington, New Zealand), pp. 62–66, 2009.

[44] Q. U. Zaman, D. C. Percival, R. J. Gordon, and A. W. Schumann, "Estimation of wild blueberry fruit yield using digital color photography," *Acta Horticulturae*, vol. 824, no. 5, pp. 57–66, 2009.

[45] J. Moonrinta, S. Chaivivatrakul, M. N. Dailey, and M. Ekpanyapong, "Fruit Detection, Tracking, and 3D Reconstruction for Crop Mapping and Yield Estimation," in *11th International Conference on Control, Automation, Robotics and Vision, ICARCV 2010*, (Singapore), pp. 1181–1186, 2010.

[46] S. Nuske, K. Wilshusen, S. Achar, L. Yoder, S. Narasimhan, and S. Singh, "Automated Visual Yield Estimation in Vineyards," *Journal of Field Robotics*, vol. 31, pp. 837–860, sep 2014.

[47] Z. S. Pothen and S. Nuske, "Texture-based fruit detection via images using the smooth patterns on the fruit," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, (Stockholm, Sweden), pp. 5171–5176, IEEE, may 2016.

[48] H. Patel, R. Jain, and M. Joshi, "Automatic Segmentation and Yield Measurement of Fruit using Shape Analysis," *International Journal of Computer Application*, vol. 45, no. 7, pp. 19–24, 2012.

[49] P. S. Nandyal and M. Jagadeesha, "Crop Growth Prediction Based On Fruit Recognition Using Machine Vision," *International Journal of Computer Trends and Technology (IJCTT)*, vol. 4, no. 9, pp. 3132–3138, 2013.

[50] U. O. Dorj, M. Lee, and S. Han, "A comparative study on tangerine detection, counting and yield estimation algorithm," *International Journal of Security and its Applications*, vol. 7, no. 3, pp. 405–412, 2013.

[51] U.-O. Dorj, M. Lee, and S. Han, "A Counting Algorithm for Tangerine Yield Estimation," vol. 21, pp. 279–282, 2013.

[52] U.-o. Dorj, M. Lee, and S.-s. Yun, "An yield estimation in citrus orchards via fruit detection and counting using image processing," *Computers and Electronics in Agriculture*, vol. 140, pp. 103–112, aug 2017.

[53] S. Liu, S. Marden, and M. Whitty, "Towards automated yield estimation in viticulture," in *Proceedings of the Australasian Conference on Robotics and Automation, ACRA*, (Kensington, NSW, Australia), pp. 2–4, 2013.

[54] W. S. Qureshi, S. Satoh, M. N. Dailey, and M. Ekpanyapong, "Dense Segmentation of Textured Fruits in Video Sequences," in *Proceedings of the 9th International Conference on Computer Vision Theory and Applications*, (Lisbon, Portugal), pp. 441–447, SCITEPRESS - Science and and Technology Publications, 2014.

[55] C. Hung, J. Underwood, J. Nieto, and S. Sukkarieh, *A Feature Learning Based Approach for Automated Fruit Yield Estimation*, pp. 485–498. Brisbane, Australia: Springer International Publishing, 2015.

[56] S. Bargoti and J. P. Underwood, "Image Segmentation for Fruit Detection and Yield Estimation in Apple Orchards," *Journal of Field Robotics*, vol. 34, pp. 1039–1060, sep 2017.

[57] R. Linker, O. Cohen, and A. Naor, "Determination of the number of green apples in RGB images recorded in orchards," *Computers and Electronics in Agriculture*, vol. 81, pp. 45–57, 2012.

[58] R. Linker and E. Kelman, "Apple detection in nighttime tree images using the geometry of light patches around highlights," *Computers and Electronics in Agriculture*, vol. 114, pp. 154–162, jun 2015.

[59] R. Linker, "A procedure for estimating the number of green mature apples in night-time orchard images using light distribution and its application to yield estimation," *Precision Agriculture*, vol. 18, no. 1, pp. 59–75, 2017.

[60] X. Liu, S. W. Chen, S. Aditya, N. Sivakumar, S. Dcunha, C. Qu, C. J. Taylor, J. Das, and V. Kumar, "Robust Fruit Counting: Combining Deep Learning, Tracking, and Structure from Motion," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Madrid, Spain), pp. 1045–1052, IEEE, oct 2018.

[61] Z. Malik, S. Ziauddin, A. R. Shahid, and A. Safi, "Detection and Counting of On-Tree Citrus Fruit for Crop Yield Estimation," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 5, pp. 519–523, 2016.

[62] W. Maldonado and J. C. Barbosa, "Automatic green fruit counting in orange trees using digital images," *Computers and Electronics in Agriculture*, vol. 127, pp. 572–581, sep 2016.

[63] P. Annamalai and W. S. Lee, "Citrus Yield Mapping System Using Machine Vision," in *Proceedings of the 2003 ASAE Annual Meeting*, (Las Vegas, NV, USA), 2003.

[64] D. Font, M. Tresanchez, D. Martínez, J. Moreno, E. Clotet, and J. Palacín, "Vineyard Yield Estimation Based on the Analysis of High Resolution Images Obtained with Artificial Illumination at Night," *Sensors*, vol. 15, pp. 8284–8301, apr 2015.

[65] X. Liu, D. Zhao, W. Jia, C. Ruan, S. Tang, and T. Shen, "A method of segmenting apples at night based on color and position information," *Computers and Electronics in Agriculture*, vol. 122, pp. 118–123, mar 2016.

[66] N. Behroozi-Khazaei and M. R. Maleki, "A robust algorithm based on color features for grape cluster segmentation," *Computers and Electronics in Agriculture*, vol. 142, pp. 41–49, 2017.

[67] N. Lamb and M. C. Chuah, "A Strawberry Detection System Using Convolutional Neural Networks," in *2018 IEEE International Conference on Big Data (Big Data)*, (Seattle, Washington, USA), pp. 2515–2520, IEEE, dec 2018.

[68] E. Bellocchio, T. A. Ciarfuglia, G. Costante, and P. Valigi, "Weakly Supervised Fruit Counting for Yield Estimation Using Spatial Consistency," *IEEE Robotics and Automation Letters*, vol. 4, pp. 2348–2355, mar 2019.

[69] N. Habibie, A. M. Nugraha, A. Z. Anshori, M. A. Ma'sum, and W. Jatmiko, "Fruit mapping mobile robot on simulated agricultural

area in Gazebo simulator using simultaneous localization and mapping (SLAM)," in *Proceedings of the 2017 International Symposium on Micro-NanoMechatronics and Human Science (MHS)*, (Nagoya, Japan), pp. 1–7, IEEE, dec 2017.

[70] P. A. Dias, A. Tabb, and H. Medeiros, "Apple flower detection using deep convolutional networks," *Computers in Industry*, vol. 99, no. August, pp. 17–28, 2018.

[71] A. Gong, J. Yu, Y. He, and Z. Qiu, "Citrus yield estimation based on images processed by an Android mobile phone," *Biosystems Engineering*, vol. 115, no. 2, pp. 162–170, 2013.

[72] A. Aquino, M. P. Diago, B. Millán, and J. Tardáguila, "A new methodology for estimating the grapevine-berry number per cluster using image analysis," *Biosystems Engineering*, vol. 156, pp. 80–95, 2017.

[73] A. Aquino, B. Millan, M.-p. Diago, and J. Tardaguila, "Automated early yield prediction in vineyards from on-the-go image acquisition," *Computers and Electronics in Agriculture*, vol. 144, pp. 26–36, jan 2018.

[74] M. Halstead, C. Mccool, S. Denman, T. Perez, and C. Fookes, "Fruit Quantity and Ripeness Estimation Using a," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2995–3002, 2018.

[75] M. Jones, J. Bell, D. Dredge, M. Seabright, A. Scarfe, M. Duke, and B. MacDonald, "Design and Testing of a Heavy-Duty Platform for Autonomous Navigation in Kiwifruit Orchards," *[Preprint, submitted to Biosystems Engineering]*, 2019.

[76] H. Williams, M. Nejati, S. Hussein, N. Penhall, J. Y. Lim, M. H. Jones, J. Bell, H. S. Ahn, S. Bradley, P. Schaare, P. Martinsen, M. Alomar, P. Patel, M. Seabright, M. Duke, A. Scarfe, and B. MacDonald, "Autonomous pollination of individual kiwifruit flowers: Toward a robotic kiwifruit pollinator," *Journal of Field Robotics*, pp. 1–17, jan 2019.

[77] M. Duke, J. Barnett, J. Bell, M. H. Jones, P. Martinsen, B. McDonald, M. Nejati, A. Scarfe, P. Schaare, M. Seabright, H. Williams, H. Ahn, and J. Lim, "Automated Pollination of Kiwifruit Flowers," in *7th Asian-Australasian Conference on Precision Agriculture (7ACPA)*, (Hamilton, New Zealand), pp. 1–5, 2017.

[78] K. Heinrich, A. Roth, L. Breithaupt, B. Möller, and J. Maresch, "Yield prognosis for the agrarian management of vineyards using deep learning

for object counting," in *14th International Conference on Wirtschaftsinformatik*, (Siegen, Germany), 2019.

[79] M. Rahnemoonfar and C. Sheppard, "Deep Count : Fruit Counting Based on Deep Simulated Learning," *Sensors*, vol. 17, no. 4, p. 905, 2017.

[80] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 32, (Las Vegas, NV, USA), pp. 770–778, IEEE, jun 2016.

[81] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv preprint*, vol. arXiv:1409, apr 2014.

[82] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," in *Computer Vision – ECCV 2014*, pp. 818–833, 2014.

[83] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Las Vegas, NV, USA), pp. 779–788, IEEE, jun 2016.

[84] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9905 LNCS, pp. 21–37, 2016.

[85] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[86] E. Shelhamer, J. Long, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 640–651, apr 2017.

[87] Y. Zhou, Y. Zhu, Q. Ye, Q. Qiu, and J. Jiao, "Weakly Supervised Instance Segmentation Using Class Peak Response," in *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (Salt Lake City, Utah, USA), pp. 3791–3800, IEEE, jun 2018.

[88] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Computers and Electronics in Agriculture*, vol. 147, pp. 70–90, apr 2018.

[89] A. Koirala, K. B. Walsh, Z. Wang, and C. McCarthy, "Deep learning Method overview and review of use for fruit detection and yield estimation," *Computers and Electronics in Agriculture*, vol. 162, pp. 219–234, jul 2019.

[90] A. Palaniappan, Won Suk Lee, and Thomas F. Burks, "Color Vision System for Estimating Citrus Yield in Real-time," in *Proceedings of the 2004 ASABE Annual Meeting*, (Ottawa, Canada), 2004.

[91] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2.1, pp. 83–97, feb 1955.

[92] G. Fielding and M. Kam, "Applying the Hungarian method to stereo matching," in *Proceedings of the 36th IEEE Conference on Decision and Control*, vol. 2, (San Diego, CA, USA), pp. 1928–1933, IEEE, 1998.

[93] A. Yuille and T. Poggio, "A generalized ordering constraint for stereo correspondence," tech. rep., Artificial Intelligence Laboratory, Cambridge, Massachusetts, Cambridge, Massachusetts, USA, 1984.

[94] Y. Si, G. Liu, and J. Feng, "Location of apples in trees using stereoscopic vision," *Computers and Electronics in Agriculture*, vol. 112, pp. 68–74, 2015.

[95] A. Plebe and G. Grasso, "Localization of spherical fruits for robotic harvesting," *Machine Vision and Applications*, vol. 13, pp. 70–79, nov 2001.

[96] R. Xiang, H. Jiang, and Y. Ying, "Recognition of clustered tomatoes based on binocular stereo vision," *Computers and Electronics in Agriculture*, vol. 106, pp. 75–90, 2014.

[97] M. Nielsen, D. C. Slaughter, and C. Gliever, "Vision-based 3D peach tree reconstruction for automated blossom thinning," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 1, pp. 188–196, 2012.

[98] C. Wang, X. Zou, Y. Tang, L. Luo, and W. Feng, "Localisation of litchi in an unstructured environment using binocular stereo vision," *Biosystems Engineering*, vol. 145, pp. 39–51, may 2016.

[99] P. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 239–256, feb 1992.

[100] R. L. Larkins, *Analysing and Enhancing the Coarse Registration Pipeline by.* Doctral thesis, University of Waikato, 2015.

[101] G. Agamennoni, S. Fontana, R. Y. Siegwart, and D. G. Sorrenti, "Point Clouds Registration with Probabilistic Data Association," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2016-Novem, (Daejeon, Korea), pp. 4092–4098, 2016.

[102] S. Kohlbrecher, O. von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable SLAM system with full 3D motion estimation," in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, (Shanghai, China), pp. 155–160, IEEE, nov 2011.

[103] G. Grisetti, C. Stachniss, and W. Burgard, "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters," *IEEE Transactions on Robotics*, vol. 23, pp. 34–46, feb 2007.

[104] J. M. Santos, D. Portugal, and R. P. Rocha, "An evaluation of 2D SLAM techniques available in Robot Operating System," in *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, (Linköping, Sweden), pp. 1–6, IEEE, oct 2013.

[105] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2016-June, (Stockholm, Sweden), pp. 1271–1278, IEEE, may 2016.

[106] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, (Kobe, Japan), p. 5, 2009.

[107] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[108] Q. U. Zaman, a. W. Schumann, and H. K. Hostler, "Estimation of citrus fruit yield using ultrasonically-sensed tree size," *Applied Engineering in Agriculture*, vol. 22, no. 1, pp. 39–44, 2006.

[109] Z. Pothen and S. Nuske, "Automated Assessment and Mapping of Grape Quality through Image-based Color Analysis," *IFAC-PapersOnLine*, vol. 49, no. 16, pp. 72–78, 2016.

[110] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *2011 IEEE International Conference on Robotics and Automation*, (Shanghai, China), pp. 1–4, IEEE, may 2011.

[111] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-Octob, (Venice, Italy), pp. 2980–2988, IEEE, oct 2017.

[112] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv preprint*, 2018.

[113] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal Loss for Dense Object Detection," in *2017 IEEE International Conference on Computer Vision (ICCV)*, (Venice, Italy), pp. 2999–3007, IEEE, oct 2017.

[114] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," *arXiv preprint*, mar 2016.

[115] W. Abdulla, "Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow," *Github repository*, 2017.

[116] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollr, and C. L. Zitnick, "Microsoft coco: Common objects in context," *Lecture Notes in Computer Science*, p. 740755, 2014.

[117] M. Seabright, L. Streeter, M. Cree, M. Duke, and R. Tighe, "Simple Stereo Matching Algorithm for Localising Keypoints in a Restricted Search Space," in *2018 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, vol. 2018-Novem, (Auckland, New Zealand), pp. 1–6, IEEE, nov 2018.

[118] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, A. Müller, J. Nothman, G. Louppe, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, jan 2011.

[119] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, pp. 509–517, sep 1975.

[120] Y. Tsin and T. Kanade, "A Correlation-Based Approach to Robust Point Set Registration," in *Computer Vision - ECCV 2004, 8th European Conference on Computer Vision*, (Prague, Czech Republic), pp. 558–569, 2004.

[121] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Boston, MA, USA), pp. 815–823, IEEE, jun 2015.

[122] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York: Springer-Verlag, 2006.

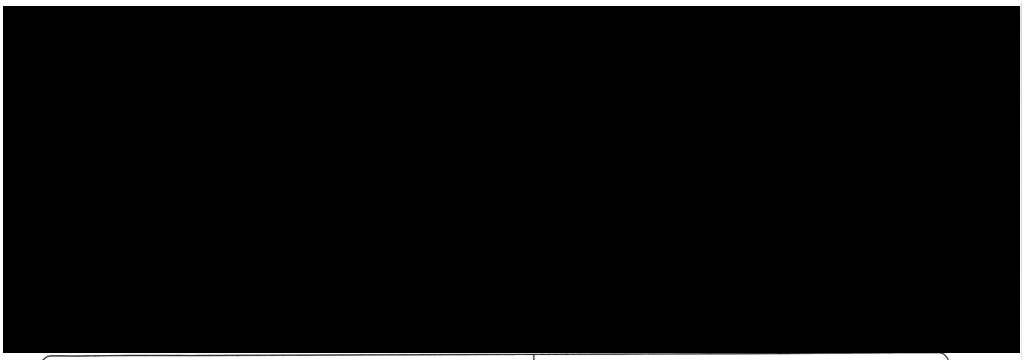
[123] G. Hutching, "Zespri to pay $250,000 after orchard worker dies in quad bike accident." https://www.stuff.co.nz/business/farming/98373542/zespri-to-pay-250000-after-orchard-worker-dies-in-quad-bike-accident [Online, Accessed 2019-09-23], 2017.

[124] P. Jeyakumar, C. W. N. Anderson, A. Holmes, and S. Miller, "Optimising Copper Sprays on Kiwifruit: a Review," tech. rep., Fertilizer and Lime Research Centre, At Palmerston North, New Zealand, Plamerston North, New Zealand, 2014.

[125] Eurofins, "Kiwifruit Maturity Clearance Testing." https://www.eurofins.co.nz/fruit-quality-testing/projects/kiwifruit-maturity-clearance-testing/ [Online, Accessed 2019-09-23], 2018.

Appendices

A Example Packout Report



Description: Combined Packruns
Packrun code: COMBINED

Maturity Area: A

Latest TZG: 0.86
Taste: Y Avg DM: 18.40

Bin Type	Bins Received	Kgs / bin	Total Kgs Received
Large	230.00	383.0	88,090
Total	230.00	383.0	88,090

Trays:

Class	Variety	Size	Grow Method	Trays		Kg each	Kg Total
1	HW	18	CK	226.0	1.0 %	3.615	817
1	HW	22	CK	2,547.0	12.3 %	3.616	9,212
1	HW	25	CK	3,409.8	16.4 %	3.616	12,333
1	HW	27	CK	3,949.2	19.0 %	3.616	14,284
1	HW	30	CK	3,893.0	18.8 %	3.617	14,081
1	HW	33	CK	3,151.8	15.2 %	3.617	11,400
1	HW	36	CK	2,540.6	12.2 %	3.617	9,190
1	HW	39	CK	754.8	3.6 %	3.616	2,730
1	HW	42	CK	238.3	1.1 %	3.616	862
Average Size: 29.14				20,710.7			74,909

Rejects:

	Kgs	
Class 2	5,330.00	6.05 %
NSS	140.00	0.15 %
Waste	7,629.00	8.66 %
Undersize	82.00	0.09 %
Totals	13,181.00	14.96 %

Summary:

Trays / bin: 90.0

Figure 12.3: An example packout report with personally identifying information censored.