Fruit Detection and Ripeness Analysis in Mango Orchards

Pavan Kumar Kavvuri

A thesis submitted in partial fulfillment
of the requirements of the degree of
Masters in Robotics.

Department of Engineering Mathematics
The University of Bristol

September 17, 2020

Abstract

Optimisation of crop management practices with accurate information based decision making approaches is crucial for precision agriculture. An accurate fruit detector framework complemented by yield estimation and ripeness analysis can help growers to efficiently target various in-farm operations including quality of crops, efficient use of resources and labour force, and harvest management. Traditionally these tasks were held based on the manual decisions on a sample of the crops which tend to be inaccurate and inclined to the subjective biases.

With the advancements in the agri-robotics, precise measurements in each square foot of cultivable land can be obtained on a per plant level. This revolution in the data availability to the growers, in contrast to the past, can help them capture the detailed representations of their farm to act wisely. This thesis incorporates robust and accurate data processing techniques to revolutionise the farm operations with data-driven approaches by presenting state-of-the-art object detection frameworks, YOLO v4 and YOLO v4 tiny in the context of fruit detection in mango orchards. The results show a high F1-score of 81.13 % and mAP value of 82.38% for YOLO v4. Analysis on the ripeness of the fruit is presented with Gaussian Mixture Models (GMM) by reflecting the potential future implications that influence the harvest management.

Acknowledgements

I would like to express my heartfelt thanks to the Centre for Machine Vision, a part of the Bristol Robotics Laboratory for accepting my co-creation proposal on Agri-Technology.

To my supervisor, Dr.Mark Hansen, you have been an excellent mentor. I greatly admire your proficiency in providing guidance and support through all aspects of this work. I have also always appreciated your constructive feedback on my work, writing and approach to research. The skills that I have learnt under your supervision will continue to play a big role well into the future.

Declaration

I hereby declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

Pavan Kumar Kavvuri

September 17, 2020

Table of Contents

List of Tables

Table **Page**

List of Figures

Nomenclature

List of Acronyms

$2D$ two dimensional

$3D$ three dimensional

$AIMD$ angular invariant maximal detector

ANN artificial neural network

AP average precision

$CLAHE$ contrast limited adaptive histogram equalization

CNN convolutional neural network

DNN deep neural network

GMM gaussian mixture model

LBP local binary pattern

$LIDAR$ light detection and ranging

mAP mean average precision

$RCNN$ region-based convolutional neural network (CNN)

RGB red, green and blue

RST radial symmetry transform

SAE sparse autoencoder

$SIFT$ scale invariant feature transform

$SURF$ speeded-up robust features

SVM support vector machine

UGV unmanned ground vehicle

WS watershed segmentation

$YOLO$ you only look once

Chapter 1

Introduction

The aim of this thesis is to develop machine vision and AI frameworks to optimise a variety of in-field or in-farm operations that will facilitate decision-making for orchard growers to meet their demand-supply needs and to estimate the time to harvest. This work utilises state-of-the-art image analysis algorithms to conduct fruit detection for estimating the fruit count and ripeness analysis tasks to plan the harvest time.

1.1 Background and Motivation

Rapid changes in global demographics from nearly seven billion today to eight billion by 2030, and apparently to over nine billion by 2050 creates a remarkable confluence of pressures. This demand the increase in food production by 70% to feed two more billion mouths [6]. On the production side, due to resource constraints and environmental concerns, competition for land, energy and water will rise while the effects of climate change will become increasingly obvious [56]. Hence the need to adapt to climate changes by reducing fossil-fuel emissions will become more crucial. Sustainable agricultural practices with the optimal usage of limited resources like land, water, fertilisers and agricultural labour force is the key to meet these needs of the critical food chain transformation. One way to accomplish this is to closely monitor and measure crops, during the whole breeding process, from seed to harvest[61] .

Precision agriculture is a field of research that aims to develop decision support systems targeting the farm efficiency through the deployment of in-farm site specific operations[3]. Such systems show the ability to process and interpret sensory information on-field to facilitate the decision-making allowing the farmers to efficiently direct a wide range of farming operations including spraying, thinning, harvesting, labour management,

crop storage and marketing. Precision agriculture enables them with the better control over farm, agricultural productivity and environmental implications [26]

Currently, farmers and agronomists manually inspect the field and take key measurements on yield, quality and crop health which is to be labor intensive and inaccurate. Unmanned autonomous or semi-autonomous robots equipped with LIDARs, 2D, 3D and thermal vision camera sensors are ideal for perceiving the on-field data by traversing between the orchard rows and perform qualitative and quantitative analysis on fruits. This analysis should help the growers to make better decisions on various steps in the supply chain. Robust and accurate machine vision algorithms are required to develop such support systems.

However, with unconstrained and unstructured agricultural settings, the challenges in bringing these systems into reality are countless. Heavy leaf occlusions, dynamic illuminations, myriad shapes, colors, textures, sizes etc. mark this area of research as a field high degree of uncertainty and complexity. Hence the present-day success is still limited, leaving agriculture as an important frontier of applied computer vision [26].

While a myriad of applications of precision agriculture are being researched, this thesis focusses specifically on mango fruit detection, yeild estimation and ripeness analysis in orchards. It therefore incorporates the algorithms and frameworks that decipher sensor information to address the fruit detection problem in orchards with heavy leaf occlusions.

1.2 Thesis Objectives

The objective of this thesis is to develop a robust machine vision framework that interprets raw digital images of the mango orchards to produce an estimate of the yield and ripeness analysis of the each detected fruit. This framework consists of

1. Localising and detecting the fruits from digital images.
2. Counting the detected fruits from all the images(from test-set) to provide a rough estimate of the yield.
3. Assess the ripeness probability of each detected fruit.

1.3 Research Questions

The following research questions are answered through the remainder of this thesis.

- **RQ1**: How can we develop a robust and accurate object detector that suits to highly dynamic agricultural settings for fruit detection and yield estimation tasks?
  - **Motivation:** Answering this question is necessary to fulfill the first two objectives of this research that will help the growers to take better decisions on their in-farm operations.
- **RQ2**: How can we assess the probability of ripeness for the detected mangos in RQ1?
  - **Motivation:** While planning to export to the distant markets, Mango is generally harvested at the mature green stage [42] to avoid over-ripening and transportation losses. Answering this research question will help growers to understand the maturity of the fruit to decide on the time to harvest.

1.4 Principle Contributions

Firstly, this thesis presents the investigation for the robust and reliable image based fruit recognition system by considering state-of-the-art object detection and machine learning approaches. The proposed framework performs pixel-wise classification of images for fruit detection. The output of this framework plays a key role not only in estimating the yield of the orchard but also analysing the maturity of the fruits.

Secondly, this research adopts the idea of Contrast Limited Adaptive Histogram Equalisation (CLAHE) from the literature [66] that enhances the contrast in badly illuminated images to reduce the effect of occlusions in the cluttered scenes.

Thirdly, all the scripts for accomplishing these tasks are open-sourced and are publicly available to access through the github repository[1].

1.5 Scope and Limitations

Due to the fixed time frame and data related constraints, this project limits it's scope to the following

1. Fruit detection and counting is performed merely on raw images but doesn't associate those images to individual orchard rows and trees due to the lack of

[1] Accessible at https://github.com/PavanproJack/Fruit-Detection-in-Orchards

orientation of image capturing. This limits the ability to estimate the yield of an orchard accurately.

2. Appearance variabilities in the data due to sun's position(sun-behind or sun-infront of the camera sensor) are not explicitly addressed in this thesis. The influence of this problem on the fruit count is discussed in detail in Section 5.2.2.

3. Ripeness of the images in the data set is not labelled. So we limit us ripeness assessment methodology to merely unsupervised learning tasks.

4. Images are not geo-tagged to a tree, so ripeness analysis methods give a global analysis on fruit maturity without associating it to fruits on individual trees.

1.6 Thesis Structure

The core components of this thesis are fruit detection and ripeness analysis frameworks. This thesis is structured in the following chapters:

Chapter 2 presents a survey of fruit detection techniques in the computer vision community that suit the orchards. Data capturing methods and a variety of object perception algorithms are investigated, extending from the on-going research in agriculture to the cutting edge advancements in computer vision. Following this, techniques for analysing the maturity of a fruit are presented and summarises with the key gaps identified in the literature thus motivating the further chapters.

Chapter 3 addresses the gaps in Chapter 2 with a research design, detailing methods chosen for this study. It briefs the experimental setup needed to successfully replicate this work and puts forward the metrics for evaluating it.

Chapter 4 presents the results of the experiments for each research question that is set out in the Chapter 1. It compares the methods chosen in the light of the evaluation metrics mentioned in the Chapter 2 and summaries with a discussion on the outcomes of the experiments.

Finally, Chapter 5 concludes this thesis with the summary of the contributions and recommendations, drawing upon the lessons learned through the research on data collection methods and experiments on various perception algorithms. It comments on the achievements and drawbacks of this work and proposes suggestions for future research.

Chapter 2

Background Research

Given the geometry of the Mango orchards, a ground-based perspective is best suited for fruit detection and counting tasks [3]. Unmanned autonomous or semi-autonomous vehicles equipped with vision and range sensors can capture data across the orchards enabling the growers to plan and organise many high-level agricultural tasks like quality assessment, yield estimation etc. Yield is typically estimated by counting the fruits detected by the detection algorithms that use the raw images captured by the sensors.

Considering the latest developments in the computer vision community, the reviews presented in the surveys of methods for detecting fruits on trees were although very comprehensive, now are obsolete [20, 25, 26, 45] with the advancements in the computer vision techniques. This chapter bridges this gap by not just introducing the most recent advancements in fruit detection but also incorporating the state-of-the-art object detection frameworks in the computer vision literature.

Harvesting is a ceaseless cycle often done 3 to 4 times in the season as the time of fruit maturity is uncertain. Harvesting at the proper stage of maturity is essential for optimum quality and often for the maintenance of this quality after harvest [31]. Of the many factors that can affect the taste quality of a product, ripeness, maturity, cultivar, irrigation, and fertilisation are especially important. The aim of this part of research is to develop a decision-support system for a grower to make valid decision on the fruit maturity.

The first Section 2.1 of this chapter reports the relevant and recent literature that is necessary to address the RQ1 of Chapter 1 by presenting the recent advances in the literature on various feature engineering techniques and sensing modalities that a fruit detection system depend upon, and gives a detailed view on frameworks and associated algorithms that use these techniques. Sections 2.2 presents the state-of-the art machine

learning object detection models that are extendable to orchards. Section 2.3 presents the techniques used in the literature to classify the ripe and unripe fruits that make use of both physical and chemical properties of fruits. We finish up in Section 2.4 with an outline of the current advancements that answer the research questions in Section 1.3 and featuring the constraints, which thus persuade the future work required in the field.

2.1 Fruit Detection

Major works presented in the literature address the problem of fruit detection as an image segmentation problem(i.e. fruit vs background). Due to high variation in the appearance of the fruits in field settings, including colour, shape, size, texture and reflectance properties, developing a fast and reliable fruit detection and segmentation system is challenging. Furthermore, in majority of these settings, the fruits are partially obstructed and subject to continually-changing illumination and shadow conditions making it hard for detection.

Fruit detection is traditionally achieved by processing the raw images captured from the orchards. Detection algorithms can be supervised to learn with annotated examples of these images to output the locations of fruits. Earlier works of computer vision in orchards used hand-engineered features like edges, corners etc. to detect specific types of fruits captured under controlled illumination. Facilitated by the advances in computing and availability of large training datasets, hand-engineered features are outperformed by the feature learning approaches based on Deep learning [3]. Such systems learn generalisable features and have applications spanning in many fields like medical imaging, video analysis, place recognition etc.

Further in this section we start by reporting the feature engineering techniques and representations used to discriminate fruits from the background, and presents the state-of-the-art advanced feature learning approaches in computer vision community.

2.1.1 Sensing Modalities

Perception in orchards can be conducted with a range of sensors such as monocular CCD or CMOS colour cameras, multi/hyper-spectral cameras, ultrasound, thermal sensors, stereo vision and LiDAR. Figure shows how a typical mango orchard looks like and the data we capture from it. The data captured can either be a two dimensional (2D) vision

Figure 2.1: A view of a typical mango orchard with a raw image data collected

based representation of the scene, such as images, or a three dimensional (3D) model, such as point clouds. A combination of these have been used in the literature.

Barnea and Ben-Shahar in [5] accurately localised and detected the red and green sweet peppers in 3D space exploiting the depth information from RGB-D cameras. RGB-D cameras are digital cameras that are equipped with tandard CMOS sensor to capture color information along with the depth of the scene. Shape based local features are used as they are partially translational invariants. Bulanon and Burks in [10] proposed an interesting technique to detect fruits in orchards by fusing thermal image and visible spectrum image captured from thermal camera and digital color camera respectively at different fields of view and spatial resolutions and showed the improved fruit detection results in orange canopies than while using digital images alone.

Payne and Walsh [46] proposed a new data collection scheme to capture mango orchard images at night to allow for the sufficient contrast between foreground and background and used texture filtering and Hessian filtering to rule out trunks, leaves and stems. Capturing data by a focused narrow light beam on a tree at nights, we can eliminate majority of non-fruit pixels from sky, other trees and ground etc. in any given image. This improves the quality of the training data thus increasing the detection accuracy and mitigating the training time required.

2.1.2 Feature Engineering Techniques in Orchards

For discriminating the fruit from non-fruit pixels, raw data needs to be transformed into a new feature space. Mango detection in this context demands this transformation

because raw mangos are hardly distinguishable from the leaves while ripe mangos appear very close to the color of twigs and branches as obvious from the Figure 2.1. The developments in this context are discussed in detail below by presenting commonly used feature representations till date.

Color

Color images are commonly represented in RGB color space. The components R, G, and B are sensitive to illuminations which can affect the classification and detection performance as brightness and spectrum are not orthogonal in RGB making it hard to distinguish between the colors [13]. A variety of color transformations are proposed in the literature to address this problem. Akin and Kirci [1] used a color based method to detect a red pomegranate fruit against green leaves background. Hue, Saturation and Value(HSV) color space is widely used for orchard scenes to segment fruits as Hue alone can represent the color. Palaniappan and Won Suk Lee in [2] utilised the color vision alone to estimate the yield of citrus grove in real-time by segmenting out the citrus fruit with a threshold determined by the pixel scatter plot of collected images in HSI plane. In addition to these, Adaptive Histogram Equalisation techniques are used to contrast adjustments using histograms and can be applied on badly illuminated images to enhance the color for detection. A variant of Adaptive Histogram Equalisation technique, CLAHE is used to reduce noise amplification during data capturing of color retinal image [57].

Further, the ripeness analysis section 2.3 details the adoption of this idea in calculating the probability of ripeness in the detected fruit.

Another important color space used is the L*a*b*, that is developed to more closely model human perception by representing color as lightness from black to white, green to red, and blue to yellow. Color channels of the color space are often normalised to reduce the changes in illuminations on the raw images. This technique was adopted to identify lemons in [22] and red apples in [60].

For the orchards with fruits similar to the color of background leaves/trunk/twigs, color alone is not a sufficient factor to discriminate the background from fruit. The following section section discusses the extraction of further features along with color.

Shape and Texture

Features like smoothness, regularity and coarseness often differ between the fruits and background. Adapting to the changing color of fruit in different growth stages, Stajnko and Cmelik [60] developed a robust and adjustable algorithm to encompass these variations

in apple orchards but faced challenges in detecting partially hidden/occluded apples. Kelman and Linker [27] demonstrated the shape analysis method using convexity and luminance profile in images to detect the apples. The proposed method showed 94% true positives and 14% false positives due to occlusion. Working against occlusions, Inkyu Sa in [55] proposed a mean average precision of 0.81 in detecting the highly occluded sweet peppers by leveraging the visual-texture features like Histogram of Oriented Gradients (HOG) and Local Binary Patterns (LBPs). While HOG describes the distribution of gradient magnitudes and their directions in a local window size, LBP gives the local binary comaparisions of individual pixels with their neighbors. McCool and Sa in [37] proposed the novel use of local binary patterns(LBPs) to segment out the highly occluded sweet peppers with an accuracy of 69.2% using an evaluation metric based on Bayesian framework.

For round fruits with smooth surface like apples and grapes and mangos where the color of fruits and background is indistinguishable, Zania and Stephen [50] used the Angular Invariant Maximal Detector (AIMD) keypoint detector for the detection of smooth round fruit. They used the intensity and gradient oriented patterns on the fruit surface to rule out other pixels. Akin and Kirci in [1], while detecting pomegranates used shape analysis together with a color based method to segment round and red colored fruits from green backgrounds. Font and Palleja in [19] discussed detecting the spherical shaped grapes based on the specular reflection peaks by the means of a morphological peak detector that showed only a 7% of false positive detections on 75% occluded grape fruits. The proposed method is based on detecting the specular reflection peaks from the spherical surface of the grapes. These intensity peaks are detected by means of a morphological peak detector based on the definition of one central point and several radial points. Fruit-geometry based detection enhanced by color segmentation was introduced by Hannan and Bulanon in their work [22] showed more than 90% accurate detections and only 4% false positives.

On the other hand, Stephen and Kyle in [41] uses a combination of high dimensional gradient based feature descriptors like Speeded-Up Robust Features (SURF), Scale Invariant Feature Transform (SIFT), and Fast Retinal Descriptor (FREAK) as their local texture features to improve accuracy of detection in the grape vineyards. These features have been robust to complex texture variations but only at the expense of high computation costs.

Researchers also tried to exploit the radial features of the fruits like apples, mangos and grapes by making use of Invariant Maximal Transform(IMT) and Radial Symmetry

Transform (RST) in [38, 40, 41, 51]. These combination of features are preferred over SIFT or SURF for thier low computational cost calculations.

2.1.3 Classification and Detection

In the current context, a classifier is an algorithm that distinguishes the image pixels into fruit and non-fruit classes. This can be done in a supervised manner where a classifier is trained on annotated examples of fruit and background and less commonly in an unsupervised way.

Unsupervised method of learning would not require any labelled training examples to learn the features. Unsupervised clustering techniques like k-means clustering is used to detect pomegranates [54] and optimal threshold determining techniques like Otsu thresholding method is used to segment apples [10, 35].

A variety of supervised classifiers can learn the increased dimensionality of input feature representations and map features to output class. The range of classifiers used in the literature are Bayesian classifier for citrus fruit identification [53] and k-Nearest Neighbors (k-NN) for berries detection [41]. Oranges detection with ANNs were reported to have competitive results in the work done by Regunathan and Lee [53]. The same classifier is used to test shape and color features in the apple detection by [64]. ANNs in peach fruit detection were also reported to have the best detection accuracies in the work [30]. SVM classifiers were also used to detect orchard fruits like pineapples, bittermelons and sweet peppers [12, 59].

2.2 Feature Learning

Feature learning frameworks learn all the features from low to high level, like colors, gradients and edges etc. hence offering a more generalisable feature representation when applying to new orchards or fruit types or may be to data under differnt illuminations [3]. But feature engineering techniques would require some non-trivial modifications while doing so.

A variety of feature learning techniques that are used for vision based tasks in orchards are Gaussian Mixture Models (GMM), K-means Clustering, Sparse Auto Encoders (SAEs), Convolutional Neural Networks (CNNs) and Restricted Boltzman Machines (RBMs) out of which CNNs and SAEs are widely used for their robustness to complex shapes and patterns.

In recent literature, Sinno and Qiang in [43] have put forward a survey of transfer learning techniques that learn features of data that are outside the feature space of the data it is already trained on. Knowledge transfer is proved to increase the performance with low training data. YOLO (You Only Look Once), a state-of-the-art deep learning based object detector is based on such techniques. Yolov3 model was used to assess the flowering levels based on number of panicles produced by mango trees [28]. Trees produce hundreds of panicles, out of which only a few flower. Pooja and Priyamm [36] have tracked the vehicles using Yolov4 by optimising the anchor box predictions in the architecture.

Although CNNs are a old techniques [32]., their recent success is due to the advancements in computing using GPUs. They were first tested on MNIST dataset [33]. CNNs with deep residual learning [23] has improved the object detection performance on PASCAL VOC 2007 dataset [17] with an mAP of 85.6% compared to the mAP of 29% with an established and leading hand-engineered approaches [18].

2.3 Ripeness Analysis

Ripeness of a fruit is directly associated with the degree of ethylene content in it [49]. As the fruit matures the chemical content expands and affects the fruit color, taste and smell. These physical and chemical properties are used to classify ripe from unripe fruits. Classification based on chemical properties are destructive but more accurate methods. On the contrast, the classification based on physical properties are non-destructive and less accurate but they can have real-time applications and practically easy to use [49]. This section explores the ripeness analysis techniques based on all mentioned physical properties in general and fruit color in specific.

Advocating the importance of sensing modalities (presented in the Section 2.1.1) in the data collection, Zhang and Guo through their work [21], published that fusing spectral and textural features in the data collected by hyperspectral imaging over starwberry orchards, produced better results for ripeness. They showed that Support Vector Machine (SVM) performed better classification on the fused data than either of them considered alone. Manish and Abhishek in [14] detected the ripe and unripe mangoes using watershed algorithms that isolate the overlapped mangoes and later segmentation using the watershed transform to mark fruit and background objects.

Outside of computer vision, methods like finite element model analysis are used distinguish ripe and unripe orange fruits by their physical property called natural

frequencies [39], as stiffness of the texture for ripe and unripe fruits differ. This work is based explicitly on the idea that agricultural products show natural frequencies that differ during maturation to ripening. The idea seems promising but lacks a concrete argument to prove that it is extendable to other fruits. Besides this, techniques like artificial olfactory system that assess the ripeness quality by the odour are also explored in [9].

2.4 Summary

After a careful study of literature, we understood that there is limited work reported/published on the fruit detection in mango orchards. Most of the fruit detection systems based their analysis on fruits like apples, sweet peppers, cirus fruits, pomegranates, grape vineyards, strawberries etc. They used a variety of techniques that to address the problem of occlusion. Very few of them [3, 4], have applied the transfer learning techniques to orchard data for detection and yield estimation tasks but limited their scope to fruit detection and counts. This thesis addresses these gaps by presenting a research design that encapsulates transfer learning techniques for fruit detection with ripeness assessment on detected fruits which enables the growers to have more control over their farm operations.

Further, the sensing modalities addressed in the Section 2.1.1 inspired us to use a range of vision sensors for data collection [5, 45] and data fusing techniques [10] to improve the models performance on orchard data. With time constraints in place, we limit our data collection to the secondary sources of data presented in the Section 3.4.1 and look forward to explore these techniques in near future.

Chapter 3

Research Methodology

The main purpose of this chapter is to discuss the research methodology followed while conducting this study. This research used the quantitative methods and secondary sources of data to address the key research objectives mentioned in the Chapter 1. Open source object detection frameworks available on Github are used experimentation and evaluation.

There have been many external factors that the orchard image data is often subject to, that instigate undesirable variability in the images. Variable illuminations in different seasons and irregular shadows due to tree geometry are the major factors of the variability making it challenging for a detection architecture presented in Section 2.1 to engineer generalisable feature representations that cater for all the conditions. This chapter documents the feature learning based state-of-the-art object detection approaches that are robust to external any factors.

For identifying a fruit from the background, the detection algorithms must be invariant to illuminations. Data pre-processing techniques are introduced to enhance the contrast and quality of images without any loss of information. Contrast Limited Adaptive Histogram Equalisation is applied on images to enhance the visibility level of cluttered/unclear scenes [67]. CLAHE is reported to have improved the F1-scores of the model used to estimate the age from RSNA Bone age dataset [66].

The Section 3.1 presents the research design followed for this project by introducing the methods chosen for this study. The following Section 3.2 develops a comprehensive argument to justify the necessity and suitability of the chosen methods over other to the current context. Section 3.3 gives a comprehensive breakdown on the architectural level of these methods. Then Section 3.4 details the step-by-step procedure carried out while experimenting on the methods, from data collection to training the models. Following

the experimental setup, Section 3.7 briefly introduces the metrics used in the computer vision community to evaluate the performance of the models. Finally, Section 3.8 details the procedure to carry out inference on the test image set.

3.1 Research Design

This research employed mixed type of strategies/procedures and evaluation metrics to answer the research questions proposed in Chapter1.3. Having inspired from the literature presented in the Chapter 2, this part of study started to explore the suitable methods for detecting mangos. Yolov4 and Yolov4 tiny outperformed all the other models interms of their accuracy and inference speeds. This study evaluates these methods testify this with mango image dataset. Later analysis of ripeness is presented on the detections obtained in the previous part by using Gaussian Mixture Models clustering techniques.

3.2 Research Methods

The methods chosen for this research are Yolov4 and Yolov4 tiny which are state-of-the-art object detection models in the computer vision literature. Here are the performance illustration of Yolov4 against other object detection models in Figure 3.1.

Yolov4 offers faster and more accurate object detection results on MS COCO dataset [34] with high Average Precision (AP) processing more Frame Per Second (FPS) complemented by the low computational resources required for running as it can run on conventional GPU with 8-16 GB-VRAM [7]. So, is the case for Yolov4 tiny making them methods of high priority for this study.

Real-time detection is particularly important for object detection models that operate on video feeds, such as a model that counts potatoes while harvesting. Yolov4 and v4 tiny prioritizes the real-time detection with both accuracy and inference speed and conducts training on a single GPU.

For ripeness analysis tasks, clustering techniques are best suited with data being un-labelled for ripeness. K-means clustering and Gaussian means clustering techniques are popular choices for clustering unsupervised data. With a motivation to find out the probability that a fruit is in a stage of ripeness, K-means clustering is not a valid choice with its hard-assignment of clusters. With hard-assignment, K-means tends to strictly assign a pixel/data point to a cluster rather than assigning the belongingness likelihood which otherwise is called soft-assignment.

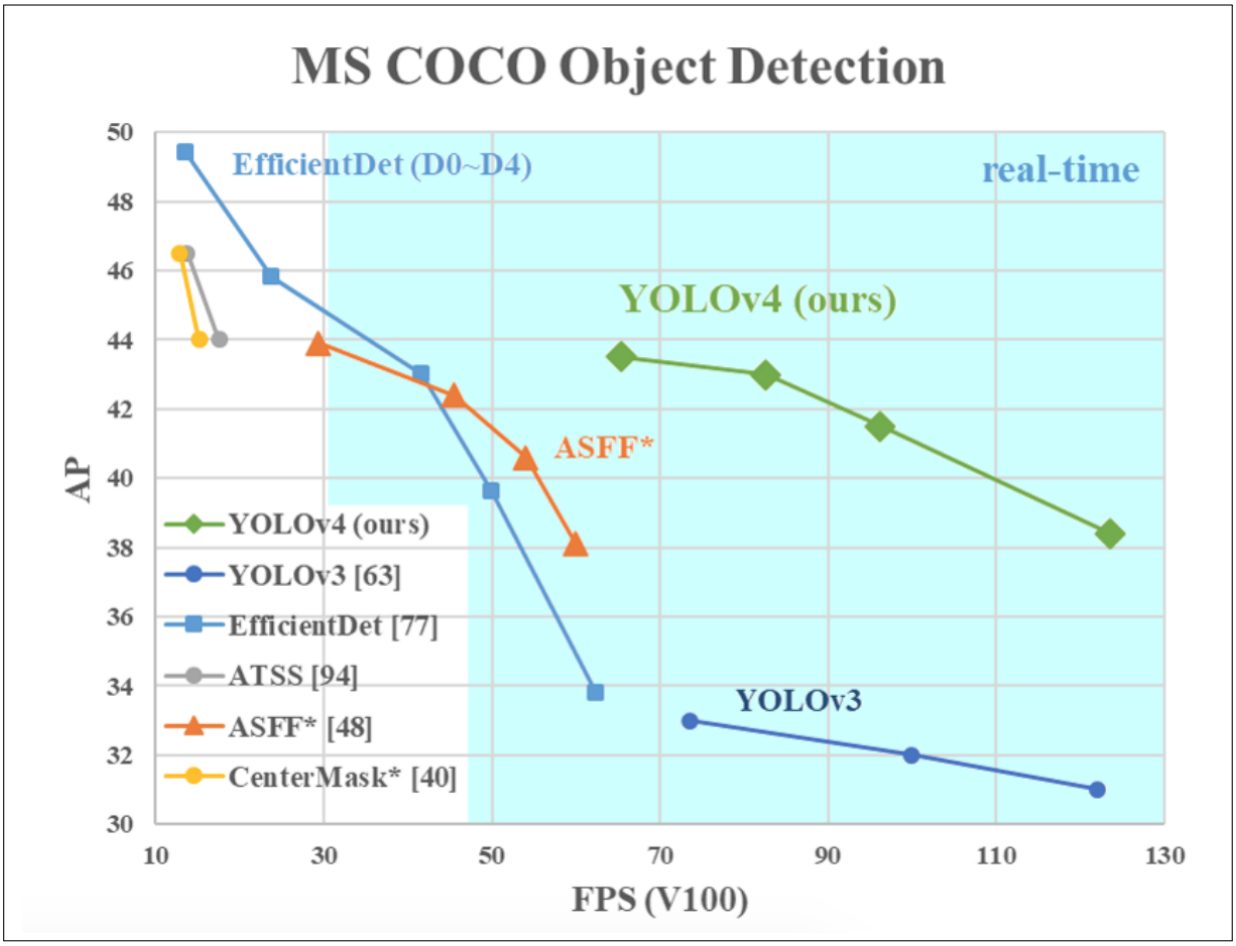


Figure 3.1: Comparision of Yolo v4 and other state-of-the-art object detectors, referenced in [7]

Gaussian Mixture Models clustering is a soft-clustering technique that gives the pixel belonging probability to a cluster. This notion of probability of belongingness becomes more obvious in the Section 4.2

3.3 Technical Background

In this section, we discuss technical details of each of the components used for the fruit detection and ripeness analysis framework mentioned in the previous section.

3.3.1 The Anatomy of Yolov4 and Yolov4 tiny

The architectures discussed here are evaluated in the Section 4.1 to answer RQ1(of Section 1.3).

Object detection models that are trained and evaluated on COCO dataset are assumed to generalise to new object detection tasks with new training data without training from scratch [58]. Yolov4 and v4 tiny are built on Darknet which makes them incredibly faster. The following section detail the background of Darknet and give a comprehensive overview on the architectures of Yolov4 and v4 tiny.

Darknet

Darknet is a custom object detection framework written in C and CUDA by Joseph Redmon that outperforms state-of-the-art object detection results. It offers better computational speed with bot CPU and GPU [52]. Yolov4 and Yolov4 tiny are implemented using Darknet.

Yolov4 and Yolov4 tiny tend to have similar architectures with primary differences in network size. From the config files of Yolov4 and Yolov4 tiny, we can see that there is a remarkable decrease in the network size for tiny as the number of convolutional layers in the backbone(CSPDarknet53) are constricted. Besides, tiny has got only two YOLO layers instead of three and fewer anchor boxes for prediction. Between Yolov4 and Yolov4 tiny, the latter suits better for tasks requiring faster inference speeds, faster training, low computation power. The comparision of Yolov4 and Yolov4 tiny is illustrated in the Figure 3.2.

Yolo is a single stage detector that does both object localisation and classification simultaneously. Traditional object detectors take image as input and compress the features using convolutional backbone layers. While solving classification problem, these layers are at the end of the network and perform the prediction tasks. Now in object detection, bounding boxes are needed to be drawn on images while also classifying fruit and non-fruit pixels. Hence the feature layers of convolutional backbone should be blended and held up considering each other [58]. This combination of feature layers of the backbone happens in the neck.

Backbone Network - Feature Formation

Yolov4 considers the following backbone networks for object detection.

- CSPDarknet53
- EfficientNet-B3
- CSPResNext50

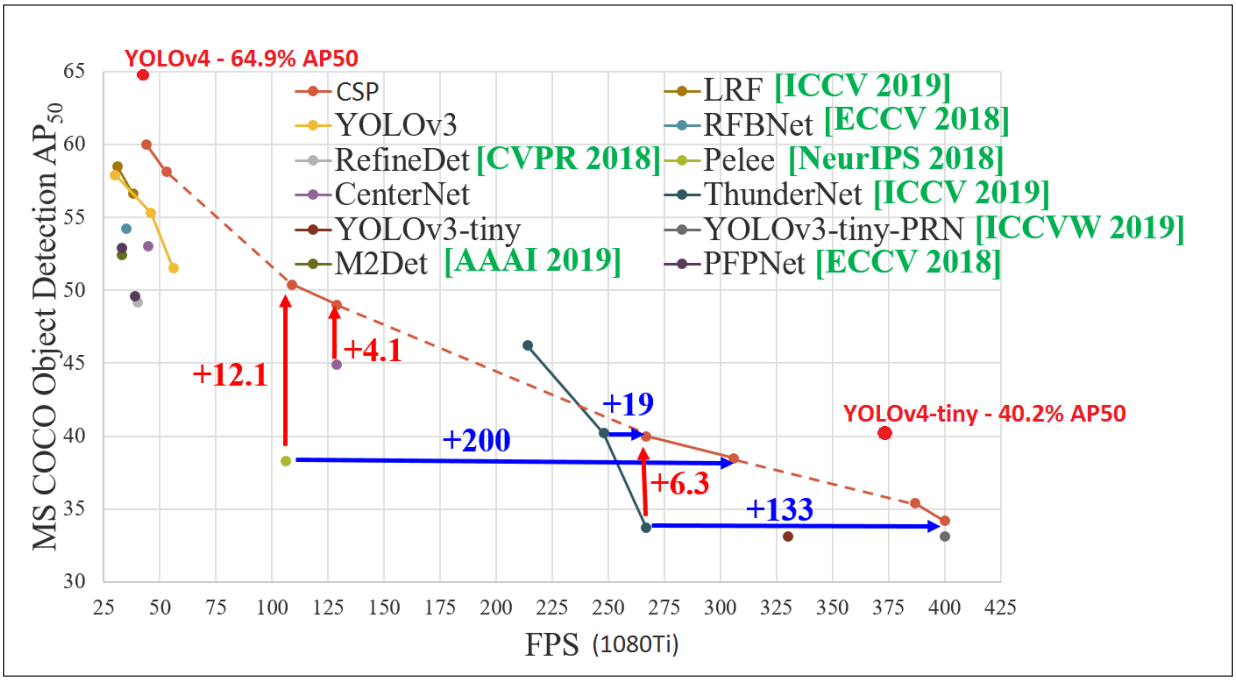


Figure 3.2: Comparision of Yolov4 and Yolov4 tiny models [7]

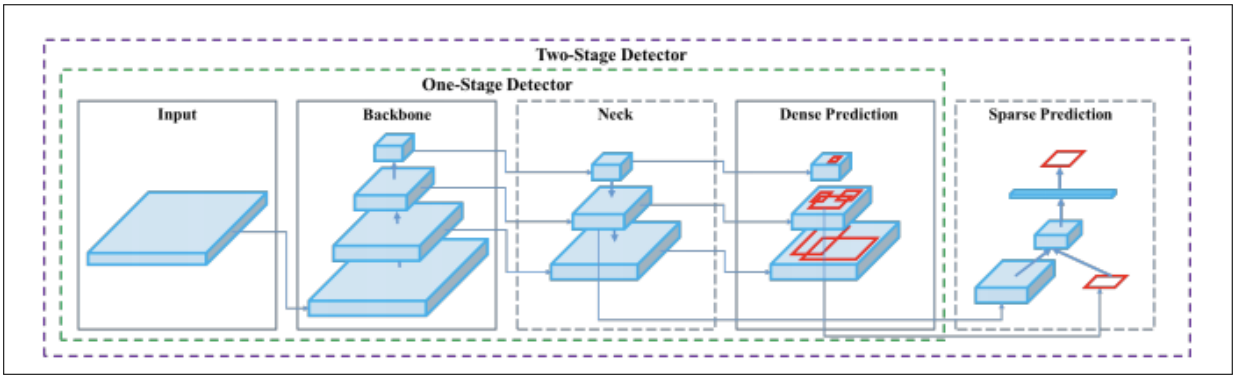


Figure 3.3: A typical build of an object detector referenced in [7]

Table 1: Parameters of neural networks for image classification.

Backbone model	Input network resolution	Receptive field size	Parameters	Average size of layer output (WxHxC)	BFLOPs (512x512 network resolution)	FPS (GPU RTX 2070)
CSPResNext50	512x512	425x425	20.6 M	**1058 K**	31 (15.5 FMA)	62
CSPDarknet53	512x512	725x725	**27.6 M**	950 K	52 (26.0 FMA)	**66**
EfficientNet-B3 (ours)	512x512	**1311x1311**	12.0 M	668 K	11 (5.5 FMA)	26

Figure 3.4: Three different backbone models comparision referenced in [7]

These networks are pre-trained on Imagenet classification [15]. Yolov4 uses CSP-

Darknet53 that is an edited version of DenseNet. The main idea behind this is to send create a copy of feature map from the base layer, pass one through the partial dense network while passing the other straight to the partial transition layer(the next block) this achieved improved learning [7].

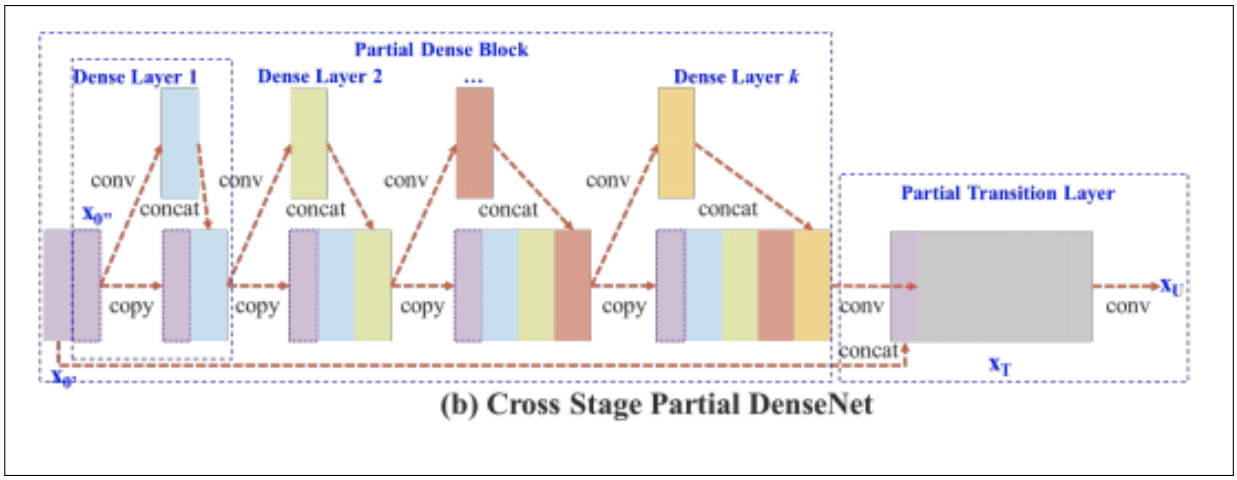


Figure 3.5: referenced in [65]

Neck - Feature Aggregation

The neck step combines the features formed in the backbone convolutional layers. Yolov4 uses PANet in the Neck stage for feature aggregation . Each node with P_i is the feature layer of the Yolov4 backbone network. The figure below shows the PANet architecture.

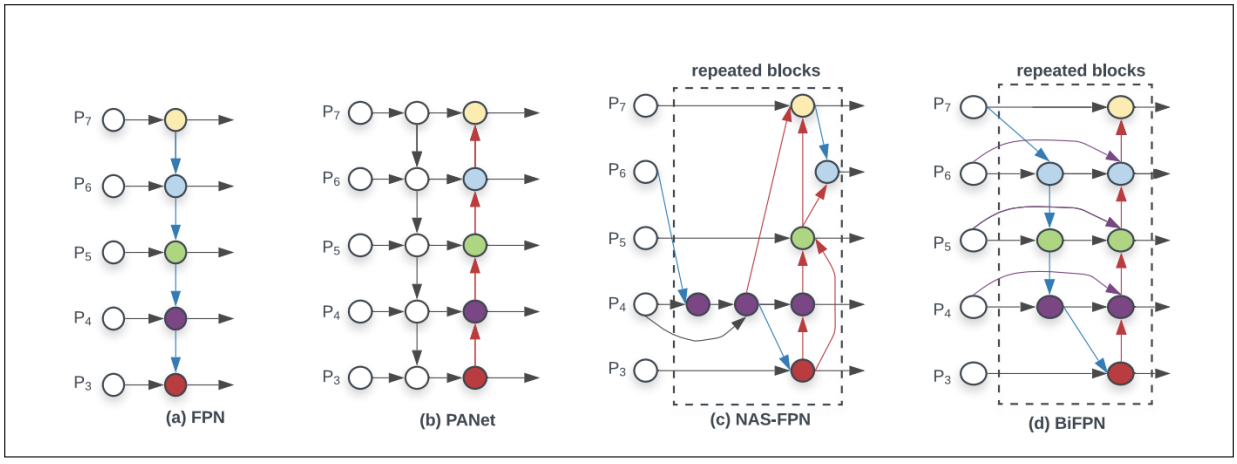


Figure 3.6: PANet architecture referenced in [62]

Head - Detection

Yolov4 adopts Yolo head from Yolov3 that uses anchor based detection steps, and three levels of detection granularity [58].

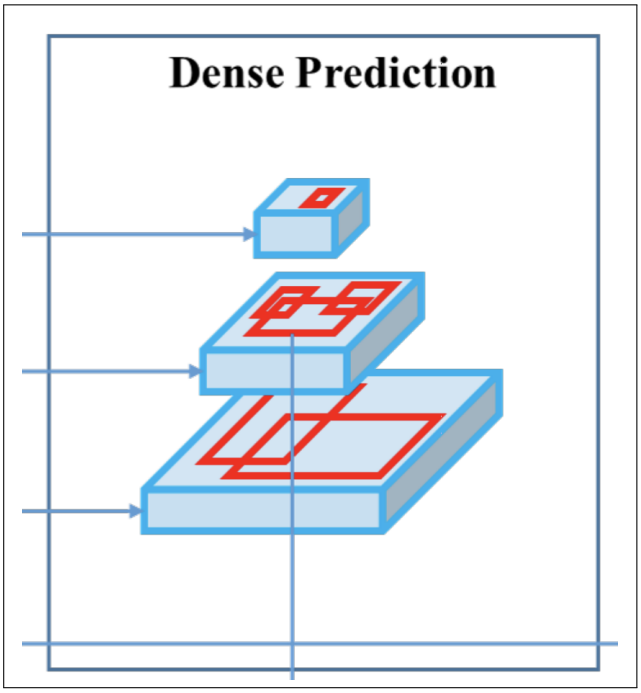


Figure 3.7: 3 levels of dense prediction in Yolov4 referenced in [7]

3.3.2 Gaussian Mixture Models

The architecture discussed here is evaluated in the Section 4.2 to answer RQ2(of Section 1.3).

K-means is the most frequently applied clustering algorithm for its convinience to use with the most software packages. The problem with the K-means is that it tends to create spherical clusters of data points making the solution biased when the data distribution is not conforming to this behaviour. This feature of k-means is not ideal for mango image data clustering with high variances in shape and volume.

Overcoming this limitation, GMM allows the clusters of variable shape, volume and orientation. Gaussain Mixture Model (GMM) which is also known as **EM Clustering** is an unsupervised clustering algorithm that is used to model multi-dimensional data

each with it's own distribution. Hence, GMM is a probability distribution that consists of multiple probability distributions.

It observes the data as generated by multiple gaussian distributions each of n-dimensional representing different clusters.

Because we have many stages of maturity for a fruit, soft clustering method that outputs the probability of a fruit in a particular stage should make more sense to a grower to plan his operations according to the market needs.

3.4 Experimental Setup

3.4.1 Data Sources and Collection

Desk research is conducted to collect data from secondary data sources like literature, Kaggle, a data science community, Google's Open Images Datasets, and online with keywords, orchard mango datasets etc. For this research data has been obtained from the study conducted by the Bargoti and Underwood [4] at the Australian Centre for Field Robotics, The University of Sydney, using a tele-operated ground vehicle called *Shrimp*, that collected the data by traversing between the rows of mango orchards during daylight hours. At the time of collecting the data, *Prosilica* camera sensor and *Velodyne* LiDAR sensors are used. This data is now open sourced and publicly accessible from here[1]. Images were manually labelled as rectangular annotations using Pychet Labeller annotation toolbox which is not in the scope of this study.

3.4.2 Data Quality

In total, 1964 images are used in this study (before augmentation) where each image corresponds to 500 x 500 pixels. Figure 3.8 illustrates the illumination variance on the raw images that affect the detection count (as discussed in the Section 4.1.2), analysis of ripeness (as discussed in the Section 4.2). Necessary pre-processing techniques should be incorporated into the study to explicitly address this problem or may be explore Active Illumination techniques for for data collection that enable night time operations avoiding any variable background light. Color constancy systems could also be developed that are invariant to illuminations.

[1] http://data.acfr.usyd.edu.au/ag/treecrops/2016-multifruit/

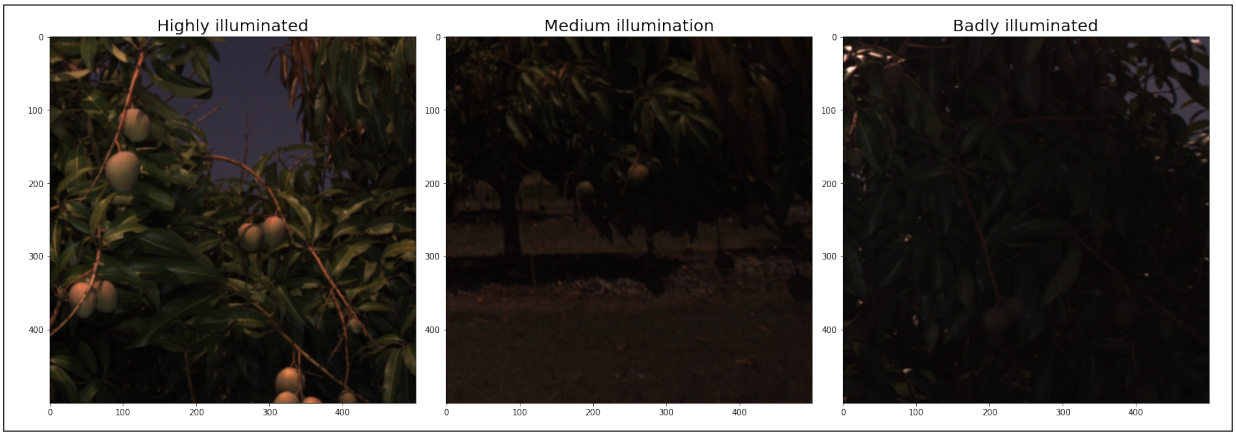


Figure 3.8: Image tiles showing different augmentation techniques applied on the dataset

3.4.3 Data Distribution and Augmentation

Out of total 1964 mango orchard images, only 70% of them contain at least one mango. Here is the distribution chart of the data collected. The performance of a detector is proportional to the number of training examples we show to it while training. The following Chart 3.9 shows the distribution of data from the source. With only 1964 images we should address the problem of over-fitting to develop a robust model for detection. The next section introduces techniques for avoiding this problem.

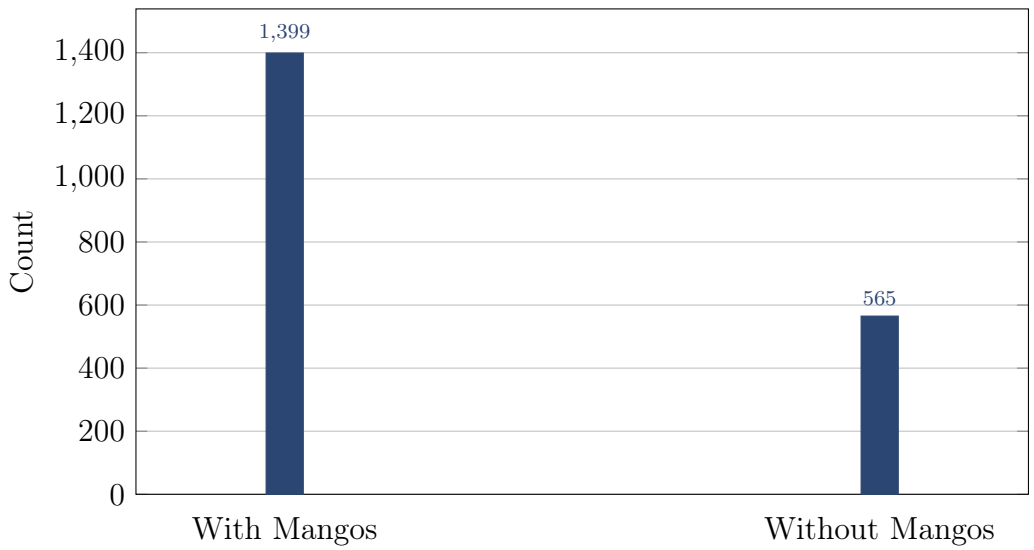


Figure 3.9: Data distribution chart

Data Augmentation

Introducing diversity in training examples would make the model more robust on the test dataset. Without actually collecting new data on-field, data augmentation techniques expand the data using label preserving transformations and increase the model's capability to reduce over-fitting. The techniques used and their significance is mentioned below:

Flipping to be robust to image orientations.

Rotation to be insensitive to camera roll.

Shear to be insensitive to camera and subject pitch and yaw.

Exposure to be robust towards lighting and camera setting changes.

Noise to be robust to camera artifacts.

Data Splitting

After augmenting the data from 1964 to 4016 images, the data is split in 3099 : 442 : 475 ratio for training, validation and testing tasks. Table 3.1 shows the partitions of the data after augmentation.

Table 3.1: Table showing the data partitions after augmentation

#	Total Images count	4016
1	Training	3099
2	Validation	442
3	Testing	475

3.4.4 Data Pre-processing

Adaptive Histogram equalization (AHE) is a contrast enhancement algorithm especially for low resolution medical images [48]. CLAHE stands for Contrast Limited Adaptive Histogram equalisation, a variant of AHE that limits the contrast amplification to limit the noise amplification.

Due to large variance in the brightness and contrast of the images in the dataset, we tried applying CLAHE on both train and test sets to see if it improves the model performance. A custom python script using opencv library is written to apply CLAHE on training and testing images. Figure 3.11 shows an example of this transformation on an orchard image. Originally, images in RGB color space are converted to L*a*b*

Figure 3.10: Image tiles showing different augmentation techniques applied on the dataset

color space and split into three separate channels L*, a*, and b*. Then using the opencv function *createCLAHE* function with a clip limit of 3.0 and grid size of 8x8 is used and apply this transformation on L* channel. Finally merge the three channels and convert back to original RGB color space. The script to carryout these steps is detailed in the appendix Section A.1

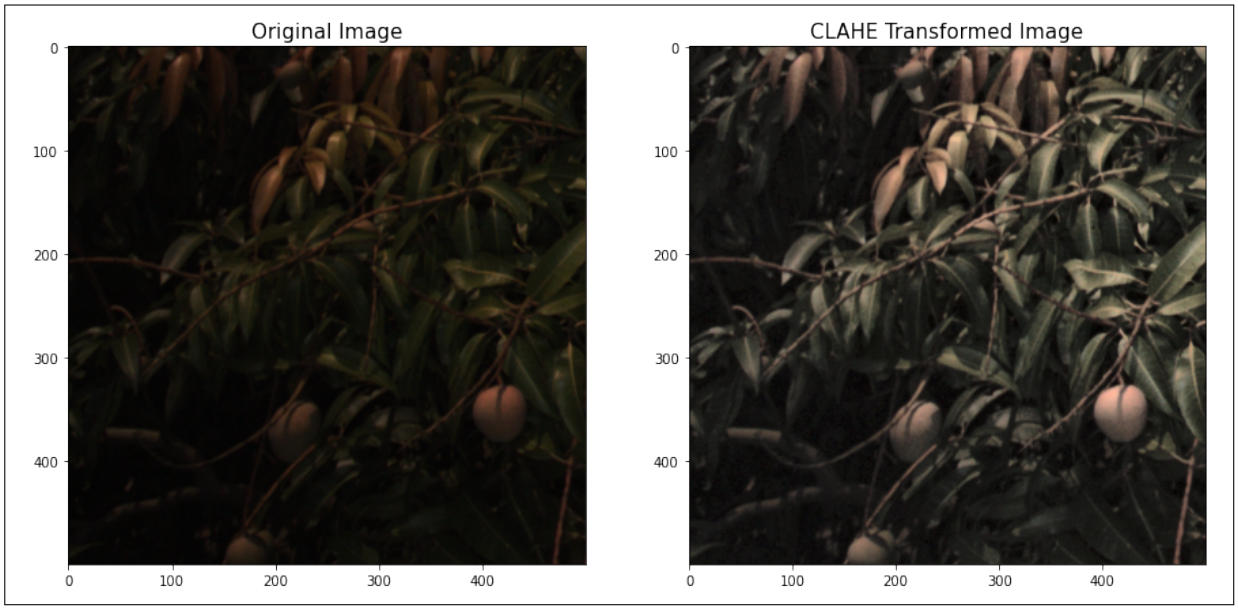


Figure 3.11: Original and CLAHE transformed Image

3.5 Hardware Specifications

Experiments on models are conducted on Goolge Colaboratory (in short Colab) environment that runs entirely on cloud. Colab offers a free GPU that is essential for training Yolov4 and Yolov4 tiny. The following Table 3.2 are the hardware specifications used at the time of running these experiments.

Table 3.2: Table showing the hardware specifications used for experiments

#	Hardware	Specification
1	GPU 0 Name	Tesla T4
2	Model name	Intel(R) Xeon(R) CPU @ 2.00GHz
3	RAM	13GB
3	Hard disk	35GB
3	CPU MHz	2000.176

3.6 Implementation and Running

Irrespective of training on cloud or in local machine, the following things are necessary to train Yolov4 and Yolov4 tiny.

1. GPU with specific GPU drivers installed
2. OpenCV (a computer vision library)
3. NVIDIA cuDNN (a GPU-accelerated library for deep neural networks configured on top of GPU drivers)

Google Colab offers a free GPU and OpenCv library pre-installed, so all that we need to do is configure cuDNN. The following commands will do that for us.

```
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
```

Start by enabling the GPU by changing the runtime type to Hardware accelerator : GPU. Using Git commands, clone the darknet repository from the github here [2] to download the source code of the architecture. Then run the *!make* command to automatically build executable programs and libraries from source code. Then comes the data part and although the data is annotated, the labels are not yet in a format that a Yolo model can digest. A custom routine has to be written to achieve the conversion. Here is the format that Yolo accepts:

Labels Formatting

Yolo expects the labels and images to conform to the following standards:

1. Label file should be in a .txt file format for each .jpg or .png image files in same directory with same name [7].
2. Each object of a class should correspond to a new line in the .txt file in the format: $< object - class >< x >< y >< width >< height >$ [7].

[2] https://github.com/AlexeyAB/darknet

where <object-class> is an integer from 0 to (classes-1) which in this context would be 0 as we are detecting only mangos. <x> and <y> are the centers of the rectangle(bounding box) in the normalised format. For eg. $< x >=< absolutex > / < imagewidth >$ and $< y >=< absolutey > / < imageheight >$. <width> <height> are the length and height of the bounding box.

Now that we have converted the annotations to Yolo standards, split the data paired with annotations in the proportion mentioned in the subsection 3.4.3. Create three folders Train, Validation, Test and paste the partitioned data into respective folders. Upload these folders as a zip file to Colab Files section and unzip it the same location. Then create a folder named *obj* in the data folder of darknet directory and copy the contents of Train and Validation folders into it. This completes the data loading step.

Then create the files *train.txt* and *valid.txt* in the darknet directory and create a routine that writes the names of image files in Train and Validation folders to the respective text files. In this way Yolo distinguishes an image as training or validation data.

Following the previous step, create files *obj.data* and *obj.names* in the darknet directory. In *obj.data* file, add the metadata like location of training data(ie. *train.txt*), validation data training data(ie. *valid.txt*), *obj.names* file and backup folder, the location to which Yolo pushes the learned weight files. *obj.names* just contains name of the class. In this context, it is Mangos. By completing this step, Yolo can read this file to know the location of its training and validation data, name of the target class and location to save the progress by pushing weights file to specified location.

With this step ready, we are set to edit the configuration file.

Customising the Configuration file (.cfg)

Configuration file with extension .cfg can be found under the *cfg* folder of darknet directory. For Yolov4 experiment try editing *yolov4-custom.cfg* file and for Yolov4 tiny experiment try *yolov4-tiny-custom.cfg*. The following are the changes that we have in configuration file customised for our application and remain same for both Yolov4 and Yolov4 tiny models.

1. A *batch* is the number of images per iteration.
2. *subdivisions* are the number of pieces a batch is broken into for GPU memory.
3. *max_batches* is (no.of classes) * 2000 but not less than 6000.

4. *steps* to be set to (80% of *max_batches*), (90% of *max_batches*).

5. *filters* to be set as (no.of classes + 5) * 3, where no.of classes is 1 here.

Table 3.3: Table showing the parameters to tweak in config file

#	Parameter	Value
1	batch	64
2	subdivisions	32
3	width	416
4	height	416
5	max_batches	6000
6	steps	4800, 5400
7	classes	1 (in the three YOLO layers)
8	filters	18 (in the three convolutional layers before the YOLO layers)

Yolo expects these parameters in the configuration file to follw the following calculations: Width and height can be the multiple of 32 and 416 is the standard, max_batches = (no.of classes) * 2000 but not less than 6000, steps = (80% of max_batches), (90% of max_batches) and filters = (no.of classes + 5) * 3.

Download weights and Train

Then download the pre-trained weights for the convolutional layers from the repository here[3]. By using these weights, it helps the custom object detector to be more accurate and not have to train as long and converges faster. The following is the comman used set the detector to start training.

```
!./darknet detector train data/obj.data cfg/yolov4-obj.cfg yolov4.conv.137 -dont_show
-map
```

For 3099 training images and 6000 epochs, Yolov4 took around 24-26 hrs to train and generate mAP on the validation dataset. While Yolov4 tiny finished training in just 6-8 hrs. Architectural differences between the models is the cause for this huge time difference. Weights files will be generated and backed up to the location given in the

[3] https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.conv.137

obj.data file. Furthermore, we can find the Average Loss vs Iterations chart showing how the model performed while training.

3.7 Evaluation Metrics

The detection models that output the detections as a rectangular bounding box co-ordinates can be evaluated using the following metrics.

Intersection Over Union (IoU) IoU is the ratio of area of overlap and area of union between the ground truth bounding box and the detected bounding box. IoU threshold is 50% for true detections in this context.

$$IoU = \frac{Area(B_p \cap B_g)}{Area(B_p \cup B_g)}$$

where and $B_p \cap B_g$ and $B_p \cup B_g$ respectively denotes the intersection and union of the predicted and ground-truth bounding-boxes.

Confidence Threshold Confidence score gives the probability that the bounding-box contains an object of a specific class. Confidence threshold is often set to filter detections with low confidence score.

True Positives A true positive is a prediction that has IoU value greater than a threshold. In our experiment the IoU threshold is 50%. So all the predictions of the detector that match to more than 50% of the ground truth are considered as true positives.

False Positives A fasle positive is an incorrect classification. A detection is termed as false positive when it predicts an object with an IoU less than 50%.

False Negatives When there is no intersection between the predicted and ground truth bounding boxexs, then such predictions are termed as false negatives.

These three measures are fundamental to calculate the performance metrics like Precision, Recall and F1-Score that evaluate the models.

Precision Precision answers the question of total true positive predictions out of total positive predictions by a detector. Precision increases with increase in the confidence threshold. And False negatives are not included in the calculation of precision. Hence this measure is not a complete one to distinguish the models.

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

Recall Recall is also called as sensitivity of the model and gives the probability that our model can successfully detect objects.It answers the question of total true positive predictions out of actual ground truth objects to predict. Recall increases with a lower confidence threshold.

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

The sum $TP + FN$ gives the total positive samples in the dataset.

Precision-recall Curve If we plot the Precision on y-axis and Recall on x-axis for all values of confidence thereshold that are between 0 and 1, the plot is called Precision-Recall curve. $PR - AUC$ is the are under Precision-recall curve that summarises the curve in single number.

$$PR - AUC = \int_0^1 prec(rec)d(rec)$$

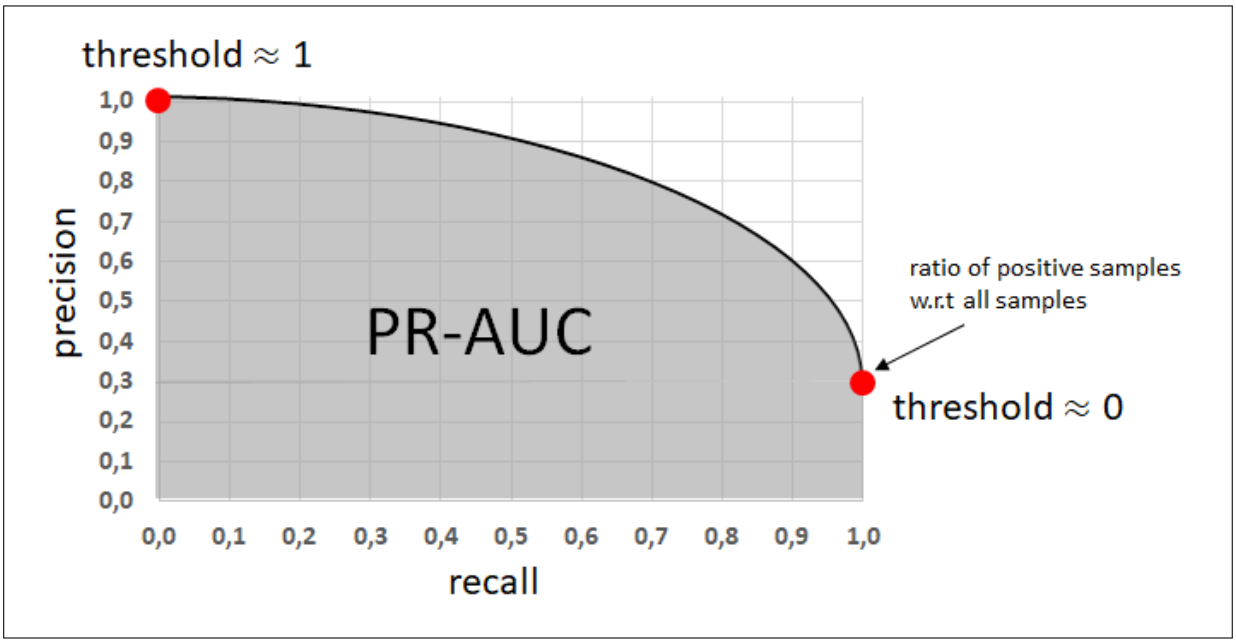


Figure 3.12: Precision-Recall Curve referenced in [29]

F1-Score Another measure that summarises the Precision and Recall is F1-Score. It is calculated as a harmonic mean of precision and recall. With this average, F1-score takes into account both false positives and false negatives.

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Average Precision (AP) AP is calculated by averaging out the Precision values for each recall value in a Precision-Recall curve. This is an important measure for calculating the mean Average Precision.

Mean Average Precision (mAP) For calculating mAP, we plot the Precison-Recall curves at multiple IoU thresholds, and take the mean of Average Precisions (APs) calculated for all the curves.

Log Average Miss Rate Log average miss rate is used to evaluate an object detection model by Piotr and Christian in their work [16]. They preferred this metric to precision recall curves in their work.

Log average miss rate is obtained by plotting the miss rate against number of False Positives Per Image (FPPI) on log axes by varying the confidence thereshold on the detections. Miss rate is defined as the ratio of False Negatives to the actual ground truth positives.

Average Loss vs Number of iterations chart This chart is an indicator for the performance of the model while training.

3.8 Inference on the Testdata

Evaluation of the model on unseen data is an important step to deploy it for real-time applications. In the light of the evaluation metrics mentioned in the Section 3.7, the procedures carried out in successfully testing them for each of the models are discussed in this section.

The evaluation metrics and procedures used here are partially inspired from the work by Cartucho and Ventura [11] that used Yolov2 for object detection tasks. The base code for this sections is taken from the github repository [4] and customised to suit the application in this context.

This part needs the results of detection in a .txt file. Fortunately, the authors of Yolo have provided the facility of saving the test set results to a text file appending the confidence of each detection. This can be achieved by the following command in colab workspace. Running this command generates a *result.txt* file to the darknet directory.

```
!./darknet detector train data/obj.data cfg/yolov4-obj.cfg <yolov4-obj_best.weights location> <test set location>
```

The following are the initial steps need to be carried out to customise the scripts.

[4] https://github.com/Cartucho/mAP

1. Insert test set images into $images - optional$ folder.
2. Insert test set labels into the $ground - truth$ folder
3. Run the script $convert\_gt\_yolo.py$ that converts ground-truth labels into a desired format.
4. Copy the $result.txt$ file to the $extra$ folder under the $scripts$ directory.

The python script $convert\_dr\_yolo.py$ parses the $result.txt$ into a desired format. New text files are created into the $detection - results$ directory in $< class\_name >< left >< top >< right >< bottom >$ format where class name is $Mangos$ and $left, topright, bottom$ are the absolute co-ordinates of the bounding-boxes. To be more specific, the script run in this step de-normalises the normalised detection results from Yolo.

Finally run the $main.py$ script that outputs the mAP measure and creates a folder named $output$ with $precision - recallcurve$, $totalfalsepositives$, $truepositives$ and a graph of $log - averagemissrate$.

3.9 Ripeness Methodology

With the information on probability of a fruit in a particular maturity stage, grower can plan the farm level harvest operations. Here we present the technical procedures to carry out the analysis. Figure 3.13 presents the connection between the outputs of RQ1 and inputs of RQ2. Figure 3.14 presents the flowchart of steps followed for answering RQ2. The associated scripts for ripeness analysis are attached in the Section A.2

3.9.1 Isolating and Pre-processing

For achieving so, we isolate the detections on each test image into a folder called $results\_img$. Having completed the training procedure for a model, create a folder named $result\_img$ under the darknet directory. Tweak the $image.c$ file in the src folder as directed by the authors of Yolo here [5] and recompile Darknet using $!make$ command. This ensures that the bounding-box pixels are saved to $results\_img$ folder for our further analysis.

All detections are rectangular bounding-boxes containing fruit and non-fruit pixels. They are hand-masked to eliminate the background pixels so that we could limit our analysis only to the fruit pixels.

[5] https://github.com/AlexeyAB/darknet/issues/954#issuecomment-393609188

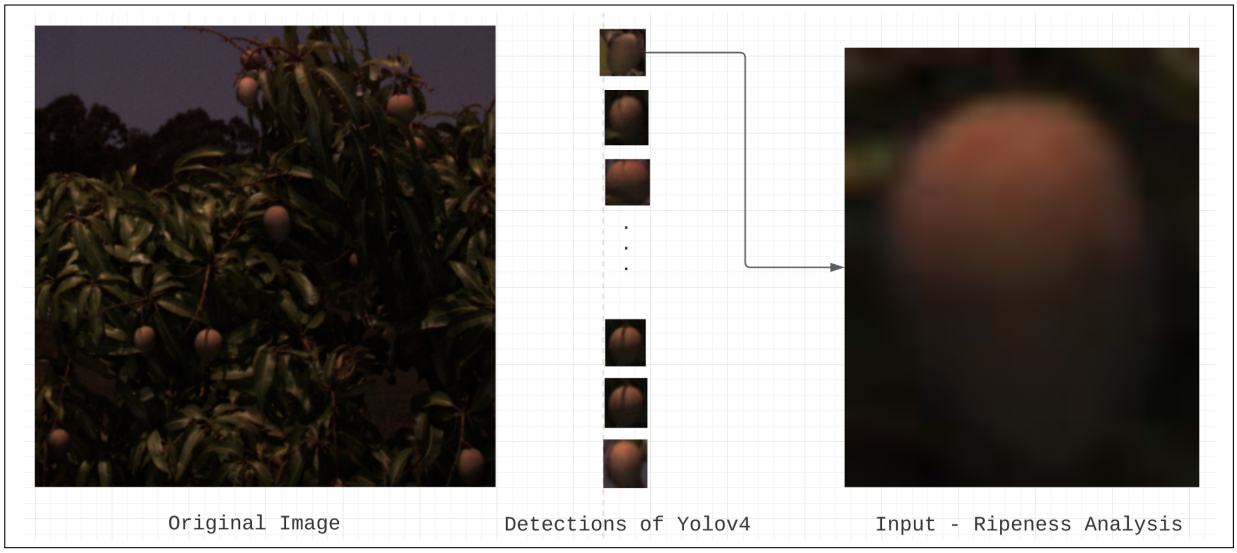


Figure 3.13: A flow chart linking the outputs of RQ1 to inputs of RQ2

3.9.2 Clustering

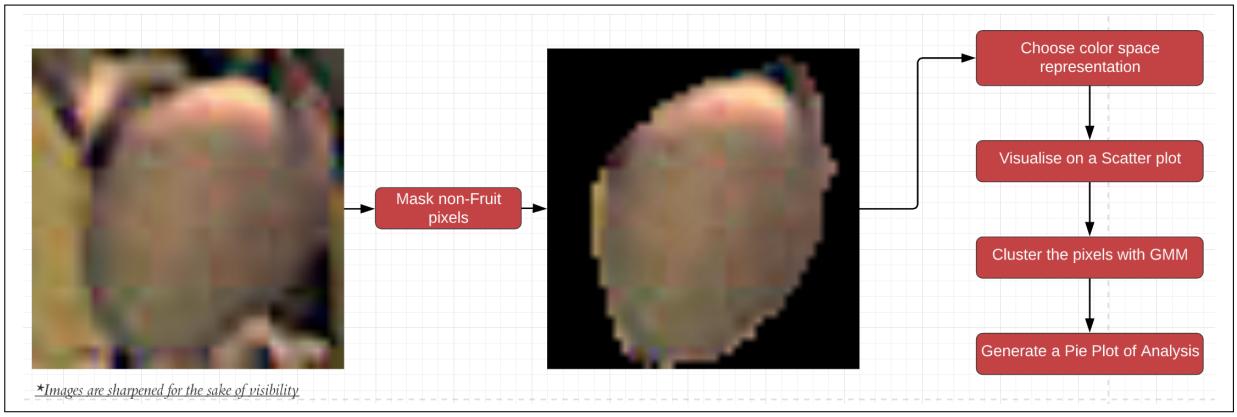


Figure 3.14: A flow chart of tasks involved in for answering RQ2

Images after masking are plotted on a scatter plot to see the distribution of pixels in RGB color space. Representations in color spaces like L*a*b*, YCrCb and HSV are reviewed to find the one that best represents the fruits at different levels of ripeness. Then images are clustered using the Gaussian Mixture Models.

To implement the GMM, we use *Python*3.0 [63] and *Scikit − learn* library [47] which is an open-source machine learning software for python to implement clustering, regression, classification algorithms etc. For reading images and masking them, we use *OpenCV* [8] which is also an open-source library with state-of-the-art computer vision and machine learning algorithms. For creating dataframes from ripeness data, we use

Pandas [44], a fast and powerful data analysis tool for python developers . *Matplotlib* [24], a python visualisation tool is used for plotting the scatter plots and color histograms. For numerical computing with image arrays we use *Numpy*, an efficient and fast python library.

The following are steps followed to while clustering the masked images.

- Import the essential Python libraries.
- Read the images in RGB color space using *imread* function of opencv.
- Reshape the image array with *reshape* function of numpy.
- Decide on the number of clusters to use.
- Instantiate the *GaussianMixture* class from scikit-learn library with the number of clusters.
- Pass the reshaped data to *fit_predict* function to estimate the model parameters and predict the labels for each data point.
- Then calculate the centres of the clusters with *scipy.stats.multivariate_norma* function.
- Predict the probability of a data point belonging to a class by applying *predict_proba* function on reshaped data.
- Create a scatter plot visualising the data with their corresponding labels.
- Now that every data point has a label of cluster associated by a probability, create a pandas dataframe grouping all the pixels by their labels. This makes more sense with the output shown in the Section 4.2.
- Create a *pandas* dataframe with all the pixels and their assocated cluster labels.
- Generate a new dataframe by grouping all the pixels by their cluster lables. Append the columns Mean R, Mean G and Mean B which are the mean values of red, green blue channels of all the pixels of a specific cluster. Also append a probabilities column with the list of probabilities obtained from *predict_proba* function.
- Eliminate the cluster containing Mean R, Mean G and Mean B as 0 as this is a cluster of mask-pixels(visually black and obvious from scatter plot). By achieving

this, we will be merely analysing the fruit pixels. We use these mean values to color sections of pie plot we draw later.

- Calculate the new probabilities by following formula

$$Pixelprobabilities(foragivencluster) = \frac{Pixelsinanygivencluster}{Totalpixelsinallclusters}$$

- Update the probabilities in the new dataframe and plot the pie chart using pandas library. This shows the probability of pixels in belonging to a particular colored cluster.

With these pieplots, we leave the decision of ripeness to the human perception and explore to automate this in our future scope with labelled training data.

Chapter 4

Experiments and Results

This chapter answers the two research questions by presenting the experimental results in two separate sections 4.1 and 4.2 and concludes with an analysis drawn from the individual sections.

4.1 RQ1

This section evaluates the performances of two object detectors discussed in the Chapter 3. Experiments are run entirely on cloud in the Google Colab environment. Yolov4, Yolov4 tiny are run for 6000 epochs. Here are the partitions of the data after augmentation, Train : Validation : Test = 3099 : 442 : 475. Evaluating these detectors in the light of the metrics mentioned in that chapter, this section also answers RQ1 proposed in Chapter 1.3.

4.1.1 Key Parameters -Comparision

From the figure 4.1, both the models are equally good in maintaining the accuracy of their predictions which is very clear from their precision scores that says, greater than 70% of their predictions are correct. However, recall scores indicate that more than 20% of the actual mangos stay undetected with v4-tiny model while it is only 14% with Yolo v4. The low resolution in the images and lot of illumination changes in the dataset required a very robust model and architecture like Yolov4.

The study of precision and recall can be further broken down by considering the parameters of confusion matrix viz. true positives, false positives and false negatives.

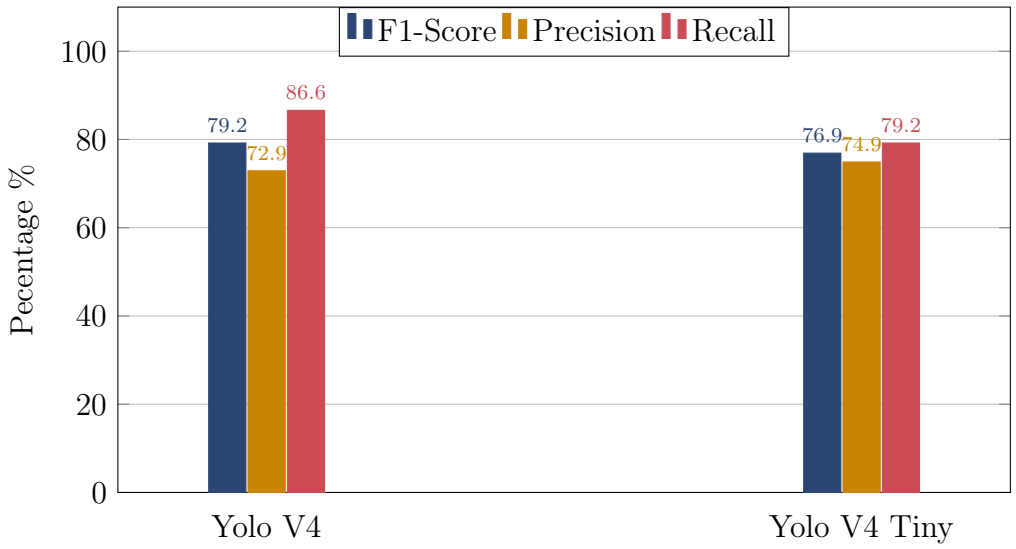


Figure 4.1: Performance comparision without CLAHE

Observing the chart below, we can infer that, out of the 1860 ground truth mangos, Yolov4 has got 7% more of its predictions correct than Yolo v4 tiny model.

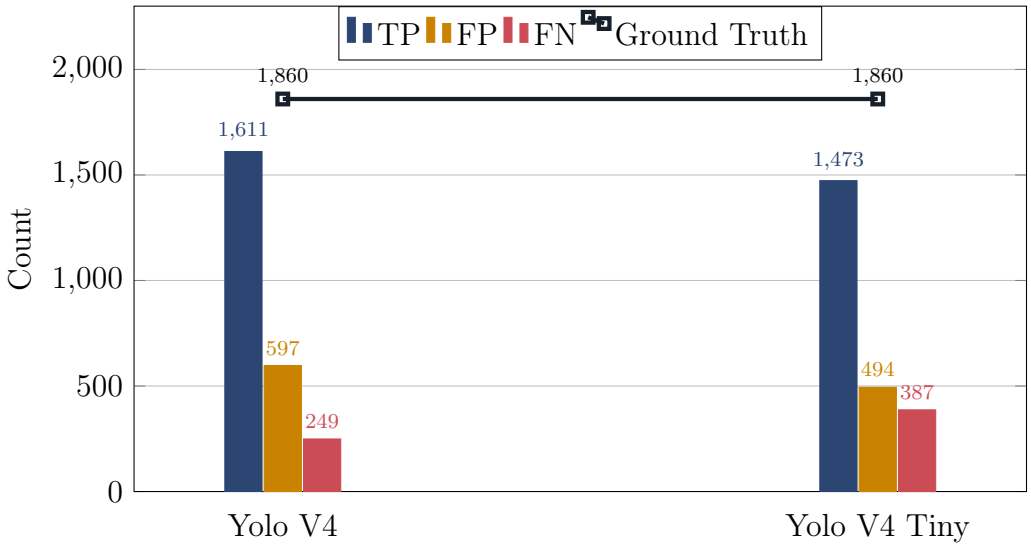


Figure 4.2: Total TP, FP, FN without CLAHE

However, in false positives, Yolov4 tiny performed well with 7% low predictions than Yolov4. In addition, with a high true positive rate, Yolov4 records a relatively low false negative rate. To generalise these quantitative results, a harmonic mean of precision and recall is calculated which is called as F1-score where Yolov4 outperforms the tiny model with a small margin of 2.22%

4.1.2 CLAHE on Testing and Training sets

Experiments to increase the accuracy of the models are conducted by applying contrast limited adaptive histogram equalisation and evaluating the performances with and without CLAHE on training and test sets. Best performances of the models are obtained while applying this affect on both the partitions.

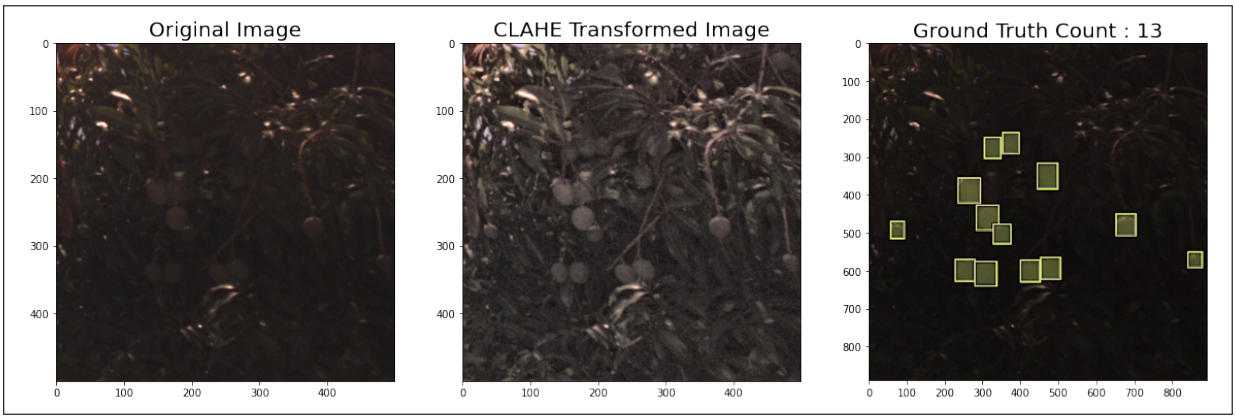


Figure 4.3: A badly illuminated image before and after CLAHE with 13 Mangos on it.

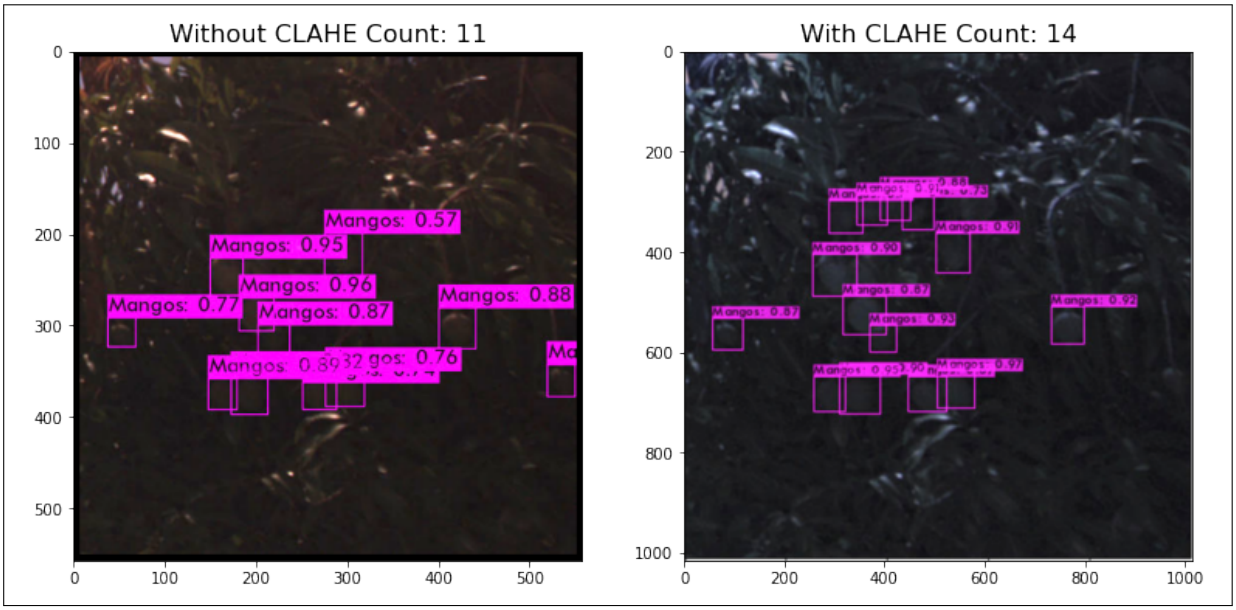


Figure 4.4: Performance of Yolov4 on the same image

Figures 4.3 4.4 4.5 show a sample of a badly illuminated image and the detections of the models on it before and after CLAHE transformation. With the ground truth count of 13 mangos in the image, Yolov4 made 14 detections on a transformed image, before

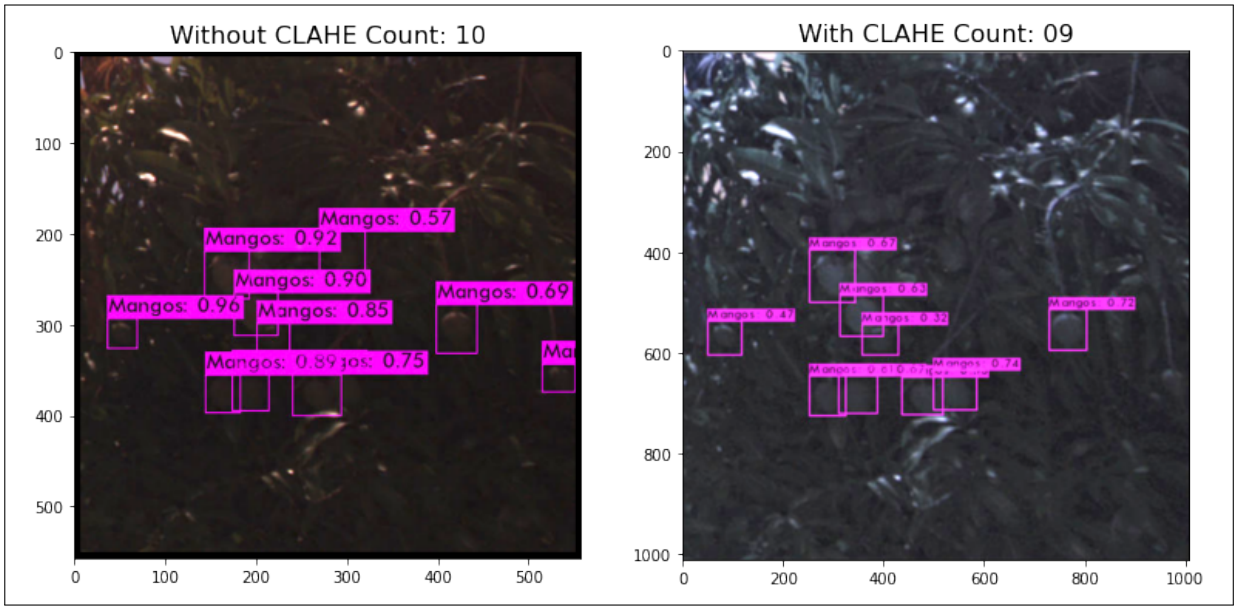


Figure 4.5: Performance of Yolov4 tiny on the same image

which it was only 11. This is not the case with Yolov4 tiny which actually underestimated the count after the transformation.

Figure 4.6 reports the percentage improvement in performance of the models after applying CLAHE on both training and test partitions.

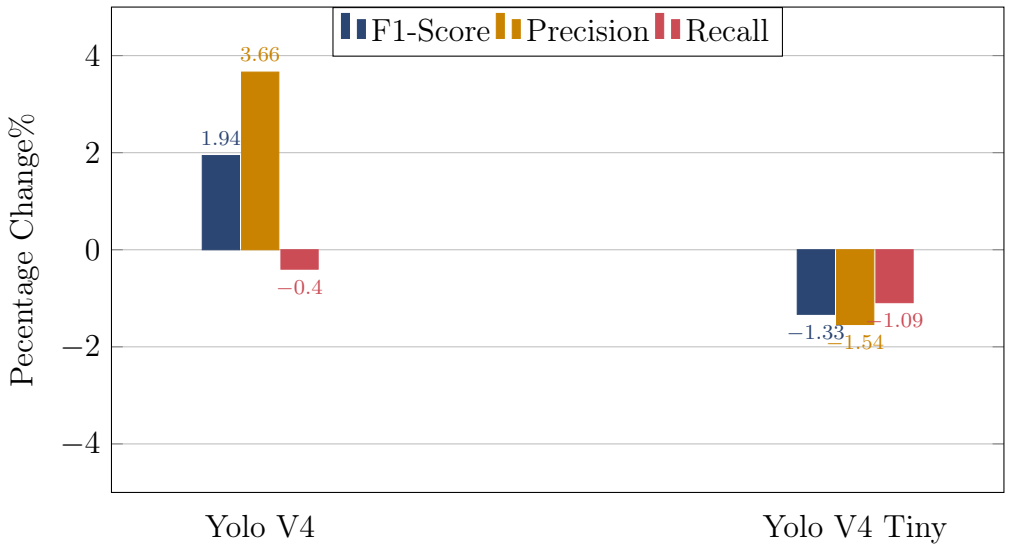


Figure 4.6: Performance comparision with CLAHE

While a clear improvement in the metrics for Yolov4 can be observed from the percentage change Figure 4.6, it is not the case with the Yolov4 tiny, where applying

histogram equalization diminished the key metrics by about 2%. The Figure 4.7 below breaks down these results in an intuitive way.

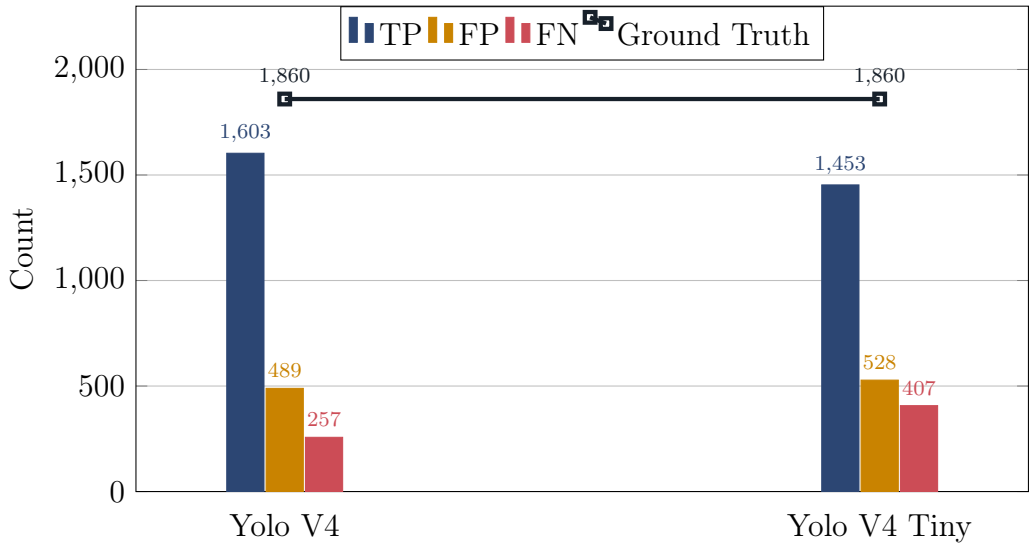


Figure 4.7: Total TP, FP, FN with CLAHE

A slight decrease in the true positive count accompanied by 6.8% rise in false positives and 5.16% rise in the false negatives have caused the decrease in precision, recall and therefore F1-score.

4.1.3 Precision-Recall Curves on Testset -Comparision

These curves are drawn by plotting precision and recall values at multiple confidence thresholds. From the Figure 4.8, area under the curve is relatively more for Yolov4 making it a better performing model than Yolov4 tiny.

4.1.4 Mean Average Precision -Comparision

Here is the comparision of mean Average Precision for all the models before and after transformation. While v4 tiny reports a very small decline in the precision and recall values, it's mAP escaltes from 72.59% to 73.17% by a value of 0.58%. Also Yolov4 reports a small increase in mAP by about 0.19% from 82.19% to 82.38%.

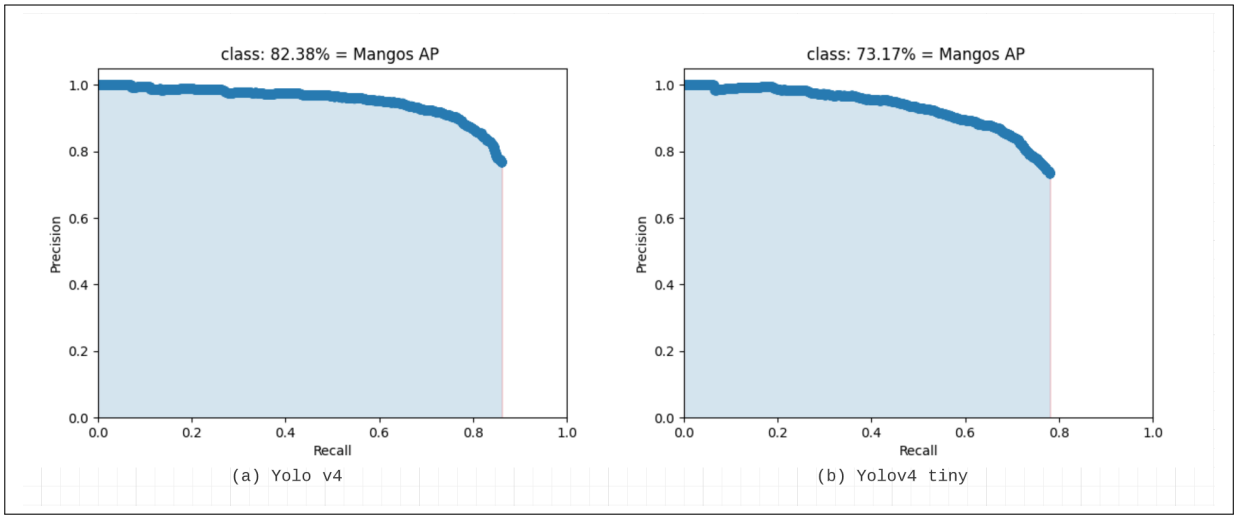


Figure 4.8: Precision-Recall curves comparision

Comparision chart of mAP with and without CLAHE

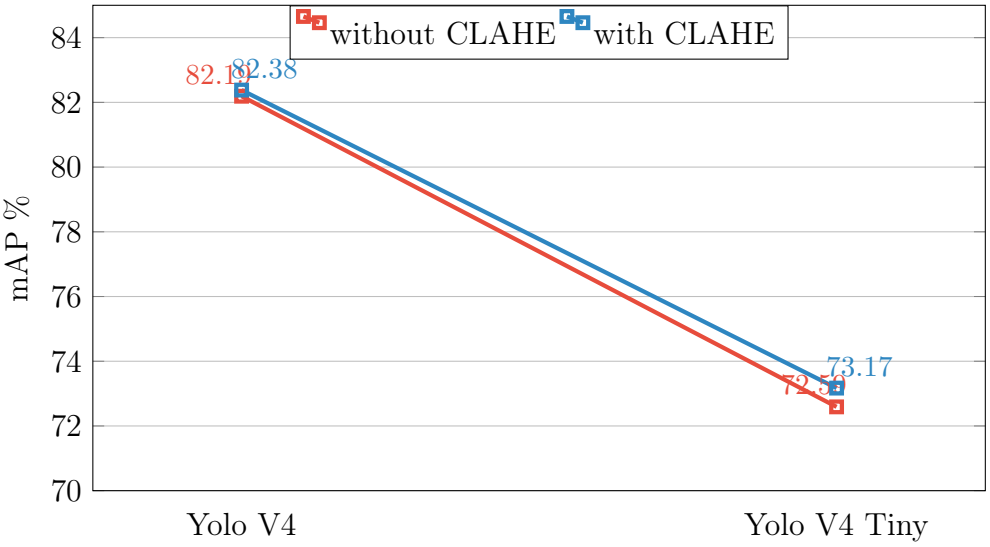


Figure 4.9: Comparision chart of mAP with and without CLAHE

4.1.5 Yield Estimation Results -Comparision

Here are the yield estimation results in Figure 4.10 that are obtained from the test set containing 475 orchard images with a total of 1860 mangos. These results are calculated by summing up the true positives and false positives obtained through each trained model at a confidence threshold of 0.5.

Although the Figure 4.10 puts forward the tiny model as a better estimator of the yield as it estimated better than Yolo v4 with a margin of 6% close to the ground truth,

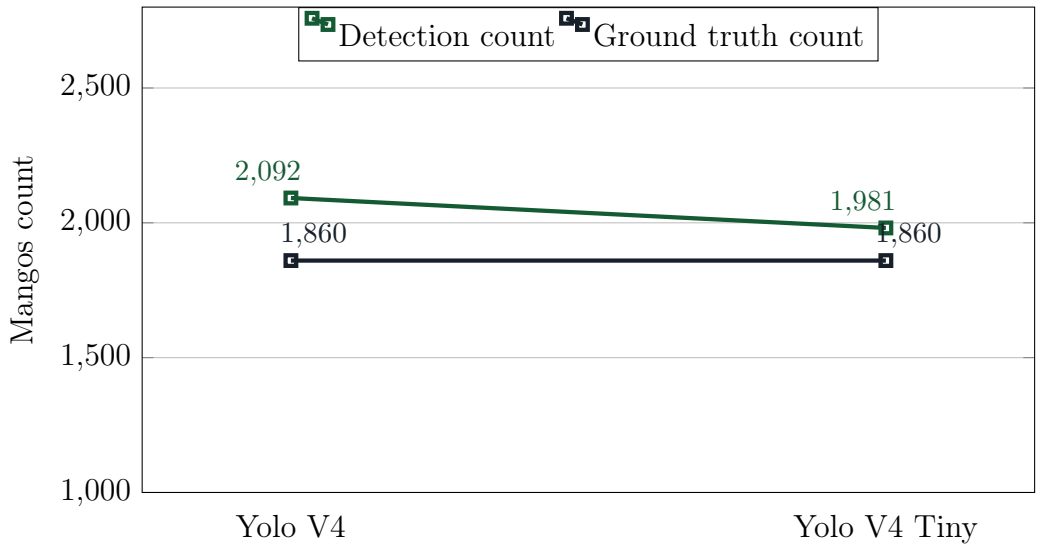


Figure 4.10: Comparision chart of yield estimation results with CLAHE

the other-side of analysis shows that 28% of this count corresponds to the false positives detected. While it is only 26% of false positive detection on overall count by Yolo v4. This implication shows that the Yolov4 is the best performer in yield estimation task.

We have also calculated the the mean detection time for both the models on the test dataset. Figure 4.11 potrays these results.

4.1.6 Mean Detection times -Comparision

It is obvious from the figure that the mean detection time for Yolov4 is high by 5 milliseconds. This can be attributed to the architectural differences in Yolov4 and Yolov4 tiny presented in the Section 3.3

4.1.7 Log Average Miss Rate -Comparision

Here is the log average miss rate for these models before and after transformation. From the Figure 4.12, we can see that average miss rate for v4 tiny is high for it's relatively high false negative count.

4.1.8 Average Loss vs Number of iterations chart

Figure 4.13 curve shows the performances of both the models interms of decreasing Loss with increase in iterations. The mAP calculations in this chart are for validation data.

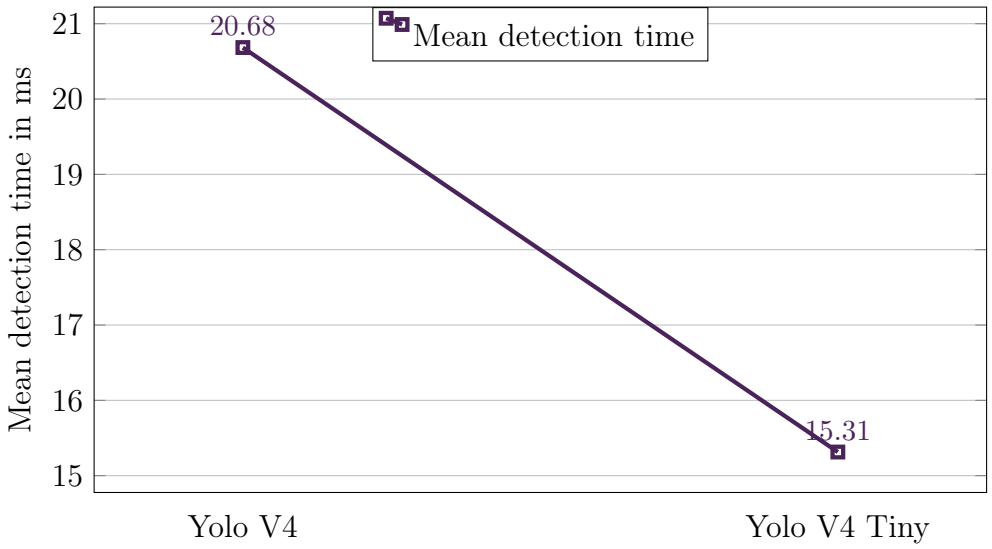


Figure 4.11: Comparision chart of mean detection times on the testset.

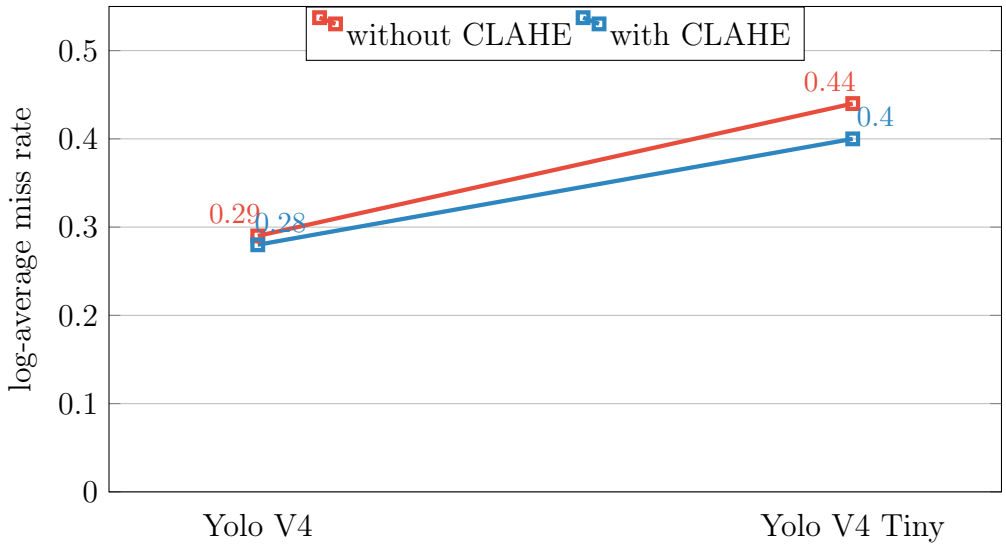
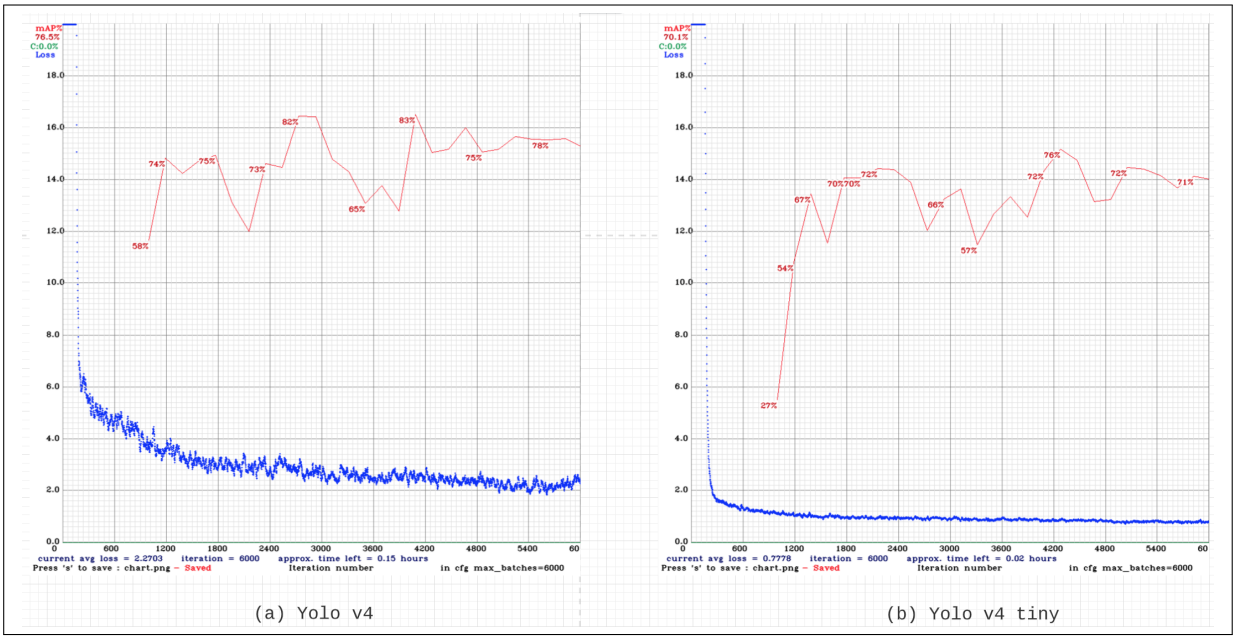


Figure 4.12: Comparision chart of Log Average Miss Rate with and without CLAHE

4.2 RQ2

For the analysis of ripeness, we clustered the mangos of three different maturity stages: (1) Raw (2) Half ripe (3) Fully ripe. Visually green mango is considered to be in Raw stage, half-ripe mangos are with pixel colors from green, orange to yellow, and a fully ripe mango has a huge accumulation of red pixels. These decisions are made based on inherent human perceptions and may change from human to human.

The remainder of this section discusses analysis on these mango stages. We have



(a) Yolo v4 (b) Yolo v4 tiny

Figure 4.13: Average Loss vs Iteration number chart

experimented our methods on 10 such mangos. For generality purposes, we present the analysis on 3 mangos in different stages. The flow of this section is according to the flowchart presented in the Section 3.9 assuming the detections of the model are saved to *result_img* folder as required in the Section 3.9.1.

4.2.1 Masking

This section presents the images that are hand-masked. Going further, these masked images are used for clustering. Figure 4.14 shows the masked fruits from original detections.

4.2.2 Color Space Representation

Scatter plots of *Matplotlib* libraray are used to visualise the color distribution of pixels. Here are the color distribution of a ripe mango in RGB, L*a*b*, YCrCb and HSV color spaces. The following Figure 4.15 visualizes the pixel distributions in different color spaces.

Here we can see that in RGB and L*a*b* color space, pixel colors are more localised and visually separable. This thesis limits it's analysis to RGB color space representation for all the mango images in use.

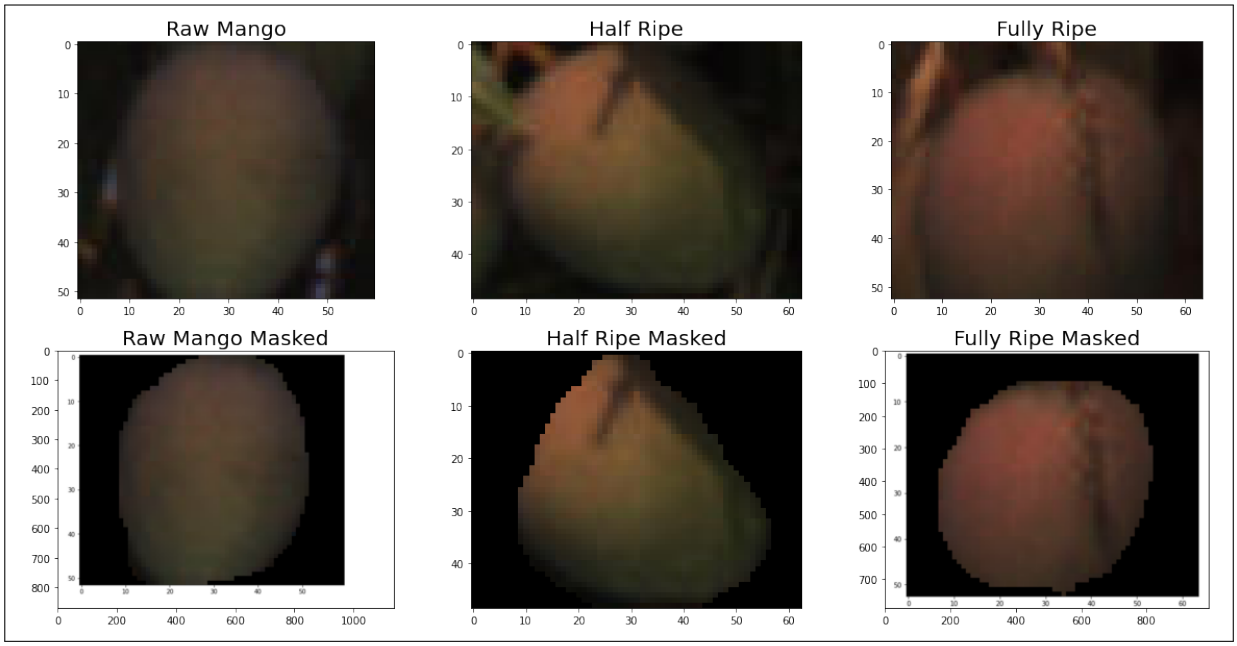


Figure 4.14: 3 Mangos at different stages masked

4.2.3 Clustering Results

Now that we stick to RGB representation, we arbitrarily choose to pick 6 dominant colors from all the 3 masked images by instantiating *GaussianMixture* class with 6 clusters. By doing so we can visually identify which colors actually are close to the human perception of ripeness. As GMM clustering outputs the probability of pixel belonging to each cluster, this analysis is going to be more intuitive. The following subsections show the analysis of mangos in use.

4.2.3.1 Raw Mango

Figure 4.16 illustrates the GMM clustering of a Raw mango on a scatter plot with cluster centers represented as dark red circles. Table 4.1 shows the pixel belongingness probabilities to each of the 6 clusters where we observed that GMM clustering outputs the every cell of *Pixelbelongingprobability* column by using the following equation:

$$Pixel - belonging - probability = \frac{\#Pixels}{\sum_{Cluster_0}^{Cluster_n} \#Pixels}$$

Out of the six clusters in the Figure 4.16 and Table 4.1, only 5 clusters belong to the fruit pixels and the other is of the masked(black) pixels which is misleading as we want to analyse only fruit pixels. So we tried eliminating the cluster with mask pixels(that

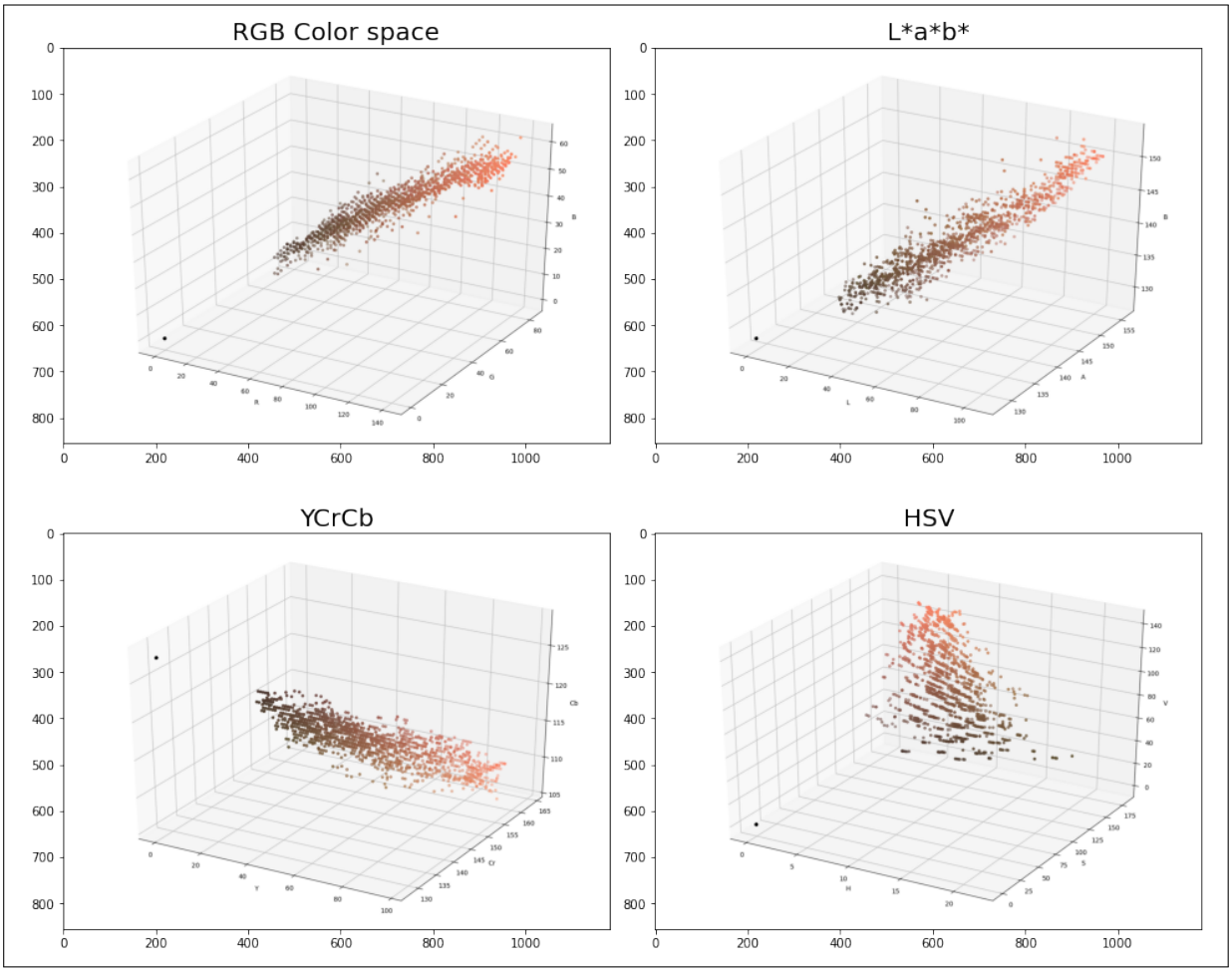


Figure 4.15: Pixels distribution of a Fully ripe mango in 4 color spaces.

Table 4.1: Table showing the pixel probabilities before dropping Mask pixels cluster

Cluster Labels	#Pixels	Mean R	Mean G	Mean B	Pixel belonging probability
Cluster 0	431	86	66	49	0.134001
Cluster 1	1240	0	0	0	0.397436
Cluster 2	361	46	42	35	0.116039
Cluster 3	504	59	52	40	0.160091
Cluster 4	76	30	27	23	0.028318
Cluster 5	508	73	60	44	0.164100

are visually black with Mean R = Mean G = Mean B = 0) and update the associated cluster probabilities by following the same formula used earlier. Thus we would have

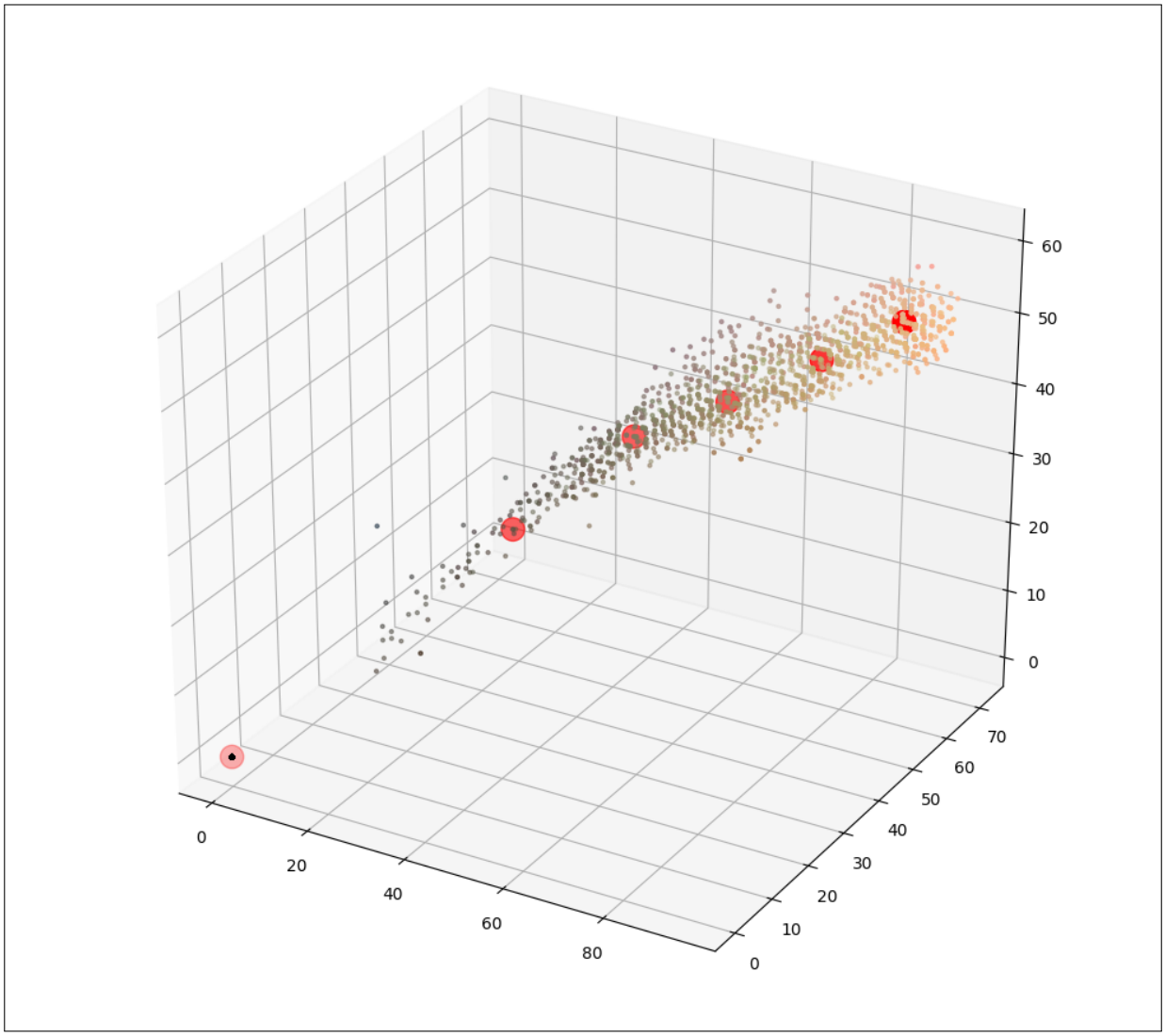


Figure 4.16: GMM clusters representation on a scatter plot for a Raw mango

only 5 clusters instead of 6 making more intuitive to the grower to assess the maturity stage of a fruit. Table 4.2 quantifies these practices.

Table 4.2: Table showing the updated pixel probabilities

Cluster Labels	#Pixels	Mean R	Mean G	Mean B	Outdated Probabilities	Updated Probabilities
Cluster 0	504	59	52	40	0.160091	0.268085
Cluster 2	431	86	66	49	0.134001	0.229255
Cluster 3	508	73	60	44	0.164100	0.270213
Cluster 4	361	46	42	35	0.116039	0.192021
Cluster 5	76	30	27	23	0.028318	0.040426

Finally the visual representation of the dominant clusters before and after dropping the non-fruit pixels cluster is shown in the Figure 4.17. The sub-figure on right represents the distribution of clusters colors and their associated pixel probabilities after dropping the mask-pixels and that of the left one before dropping. The right sub-figure is more intuitive that one can learn from it and say that about 23% of pixels of this fruit are close to ripeness.

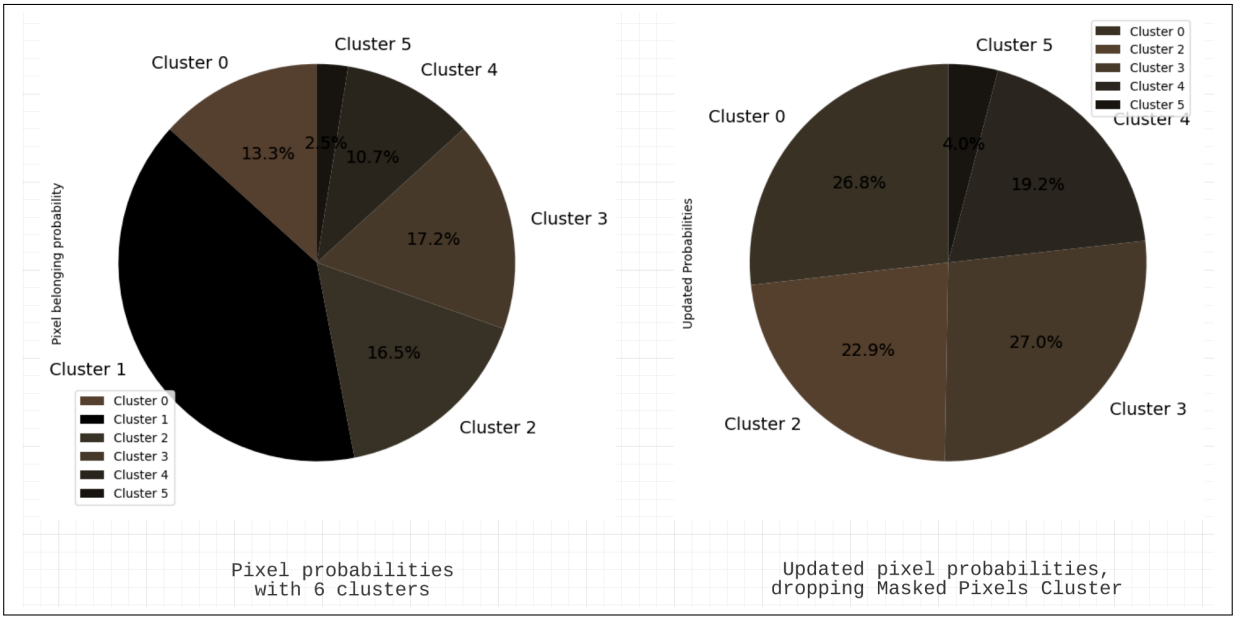


Figure 4.17: Pie chart of fruit color and probabilities distribution for a Raw mango

4.2.3.2 Half ripe Mango

In this section we present the analysis on Half a ripe mango. Figure 4.18 shows the pixel distributions in a scatter plot whereas Figure 4.19 shows the probability and color distribution of pixels.

4.2.3.3 Ripe Mango

This section presents the same analysis on a Ripe mango. Figure 4.20 shows the scatter plot and Figure 4.21 shows the pie plot of probabilty distribution with updated probabilities.

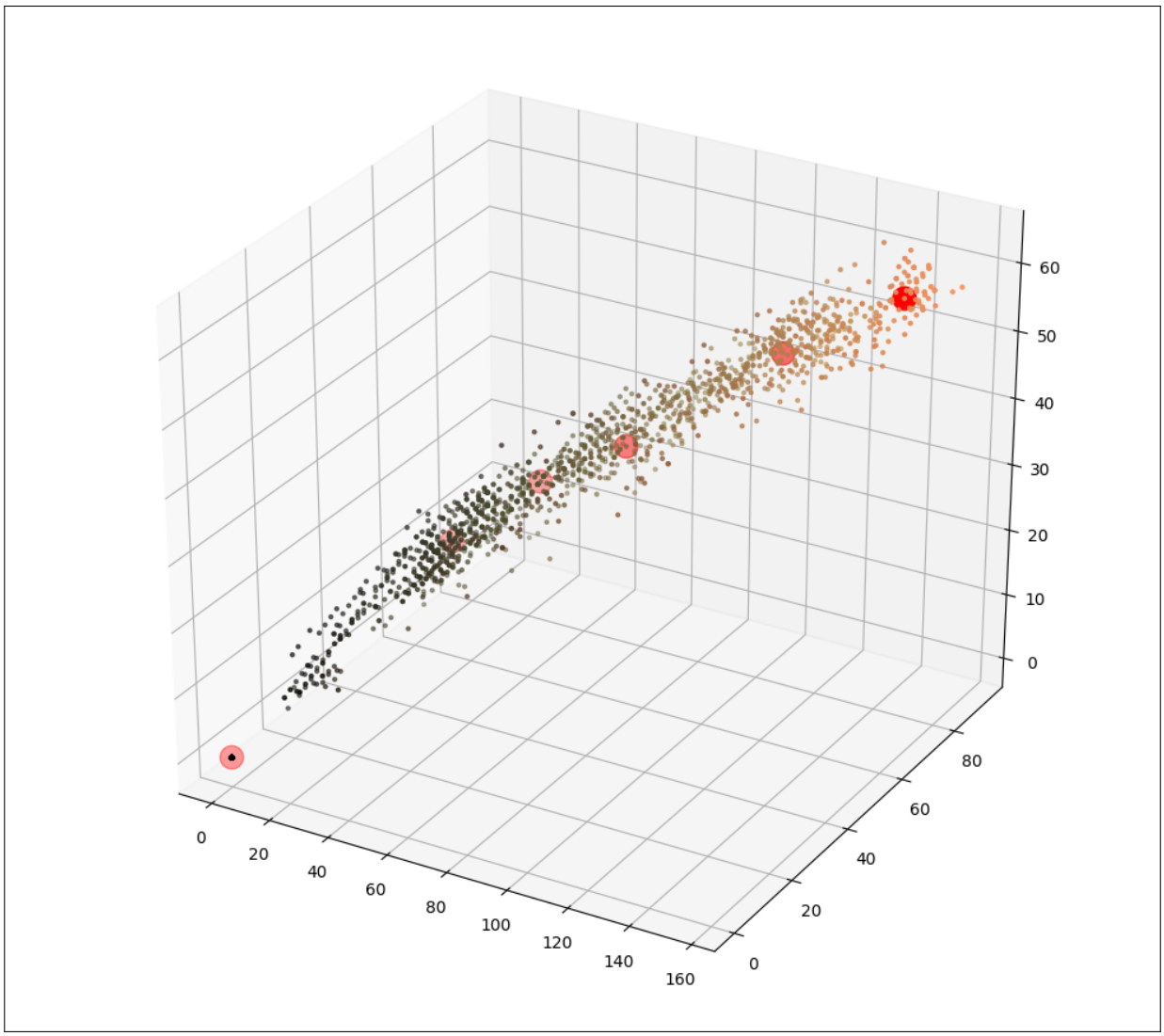


Figure 4.18: GMM clusters representation on a scatter plot for a Half ripe mango

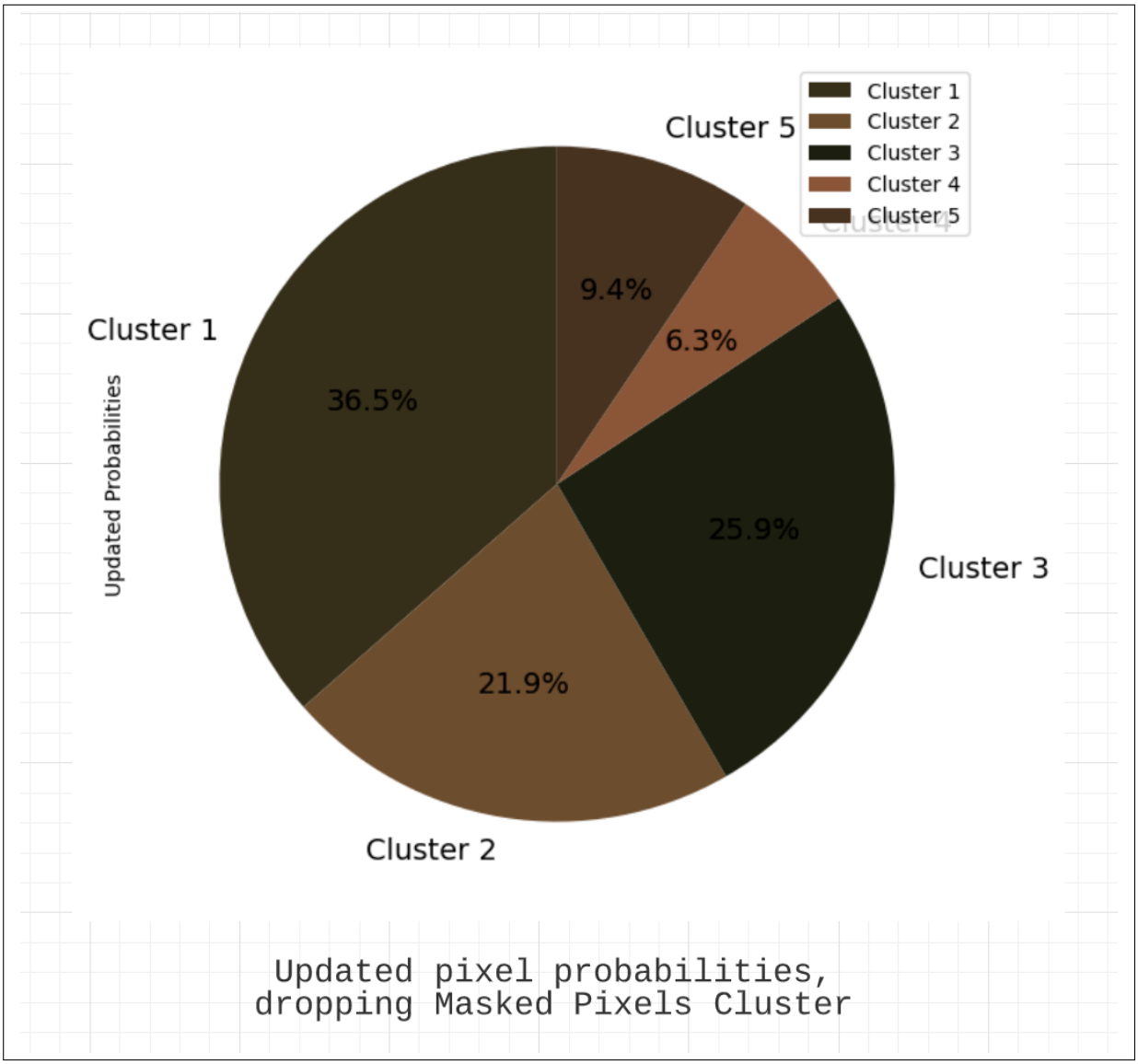


Figure 4.19: Pie chart of fruit color and probabilities distribution for a Half ripe mango

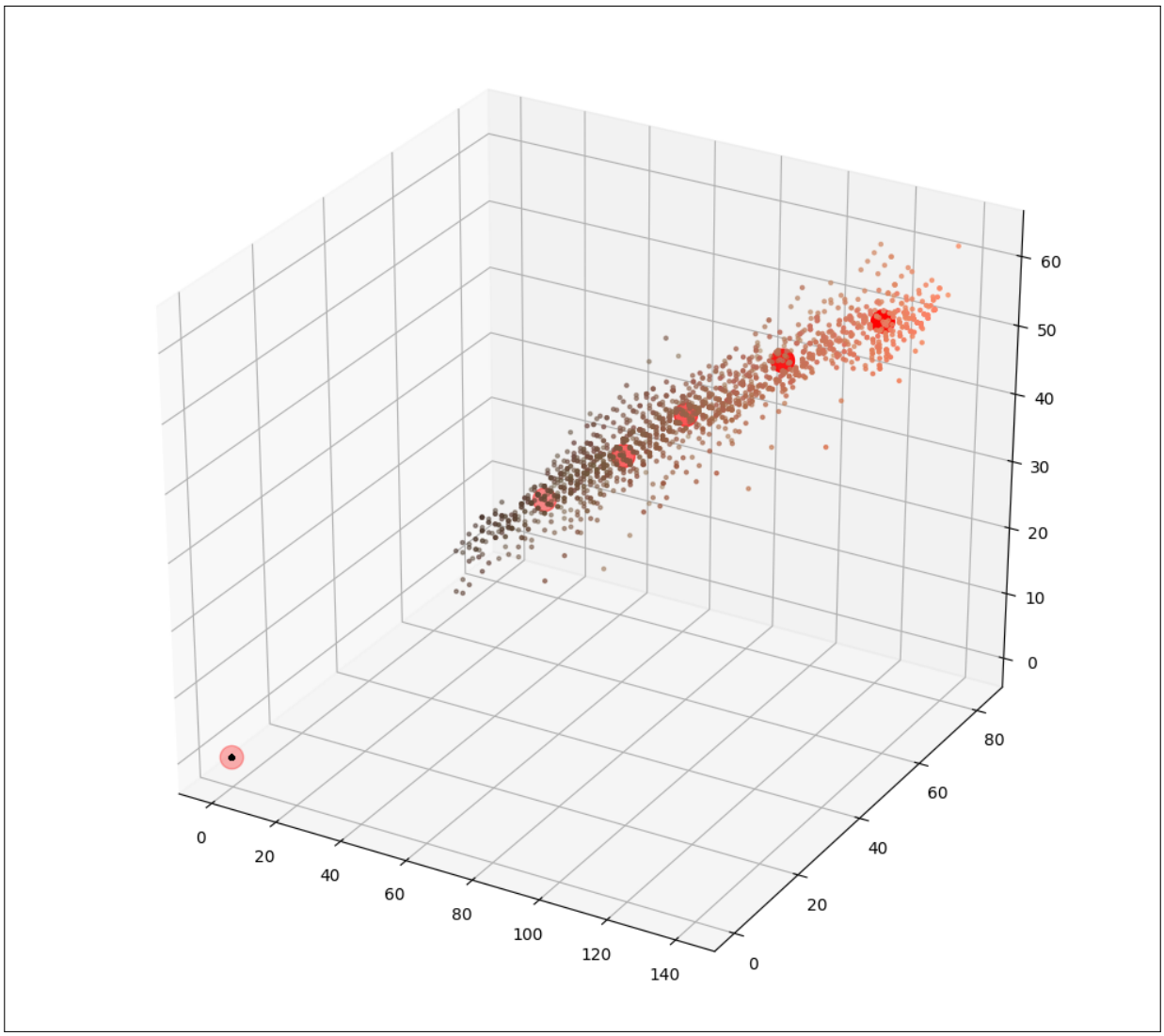


Figure 4.20: GMM clusters representation on a scatter plot for a Ripe mango

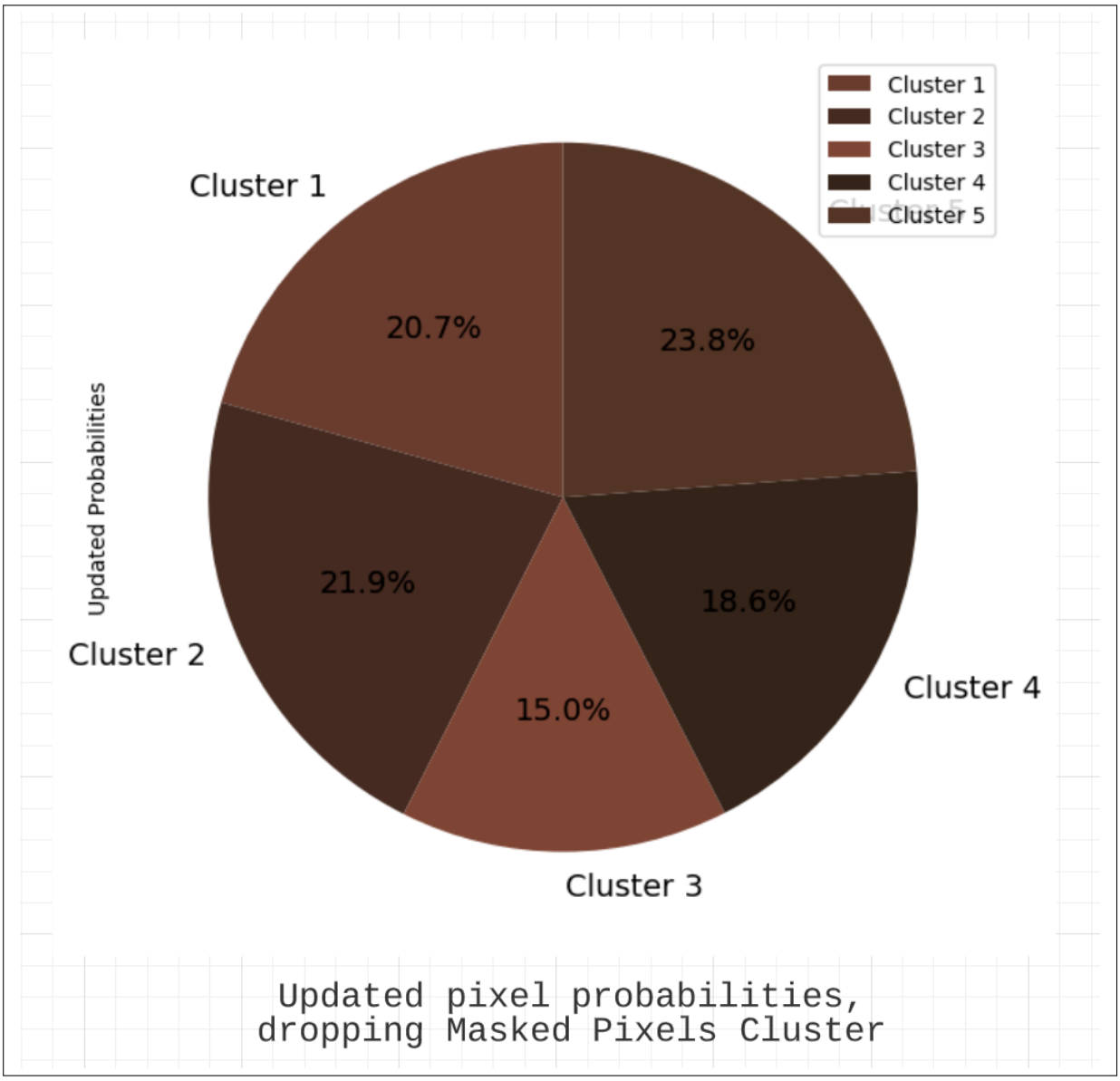


Figure 4.21: Pie chart of fruit color and probabilities distribution for a Ripe mango

4.3 Discussion

4.3.1 Answering RQ1

How can we develop a robust and accurate object detector that suits to highly dynamic agricultural settings for fruit detection and yield estimation tasks?

From the experimental results of RQ1 in Section 4.1, we can conlcude that Yolov4 is the best performing model for fruit detection in orchards. Yolov4 tiny suffers from high false positives and false negatives and may require more training data which is not ideal for our scope.

While there is not a pronounced difference noticed in the performance of Yolov4 with CLAHE, Yolov4 tiny has a negative effect on F1-score, Precision and Recall. With CLAHE, Yolov4 has considerably dropped false positive count escalating the precision by 3.66%. Finally, it performed better with F1 score, mAP and Area under Precision-Recall curve maintaining good balance between precision and recall.

When it comes to Yield estimation, Yolov4 is again the better performer with more true positive results. Thus with Yolov4, we can atleast be more confident on the predictions. Yolov4 tiny has the shorter inference time and training time compared to Yolov4. In the context of orchard fruit detection, accuracy matters over the speed of detection thus concluding that Yolov4 is the better performer.

4.3.2 Answering RQ2

How can we assess the probability of ripeness for the detected mangos in RQ1?

We have developed a clustering analysis on mangos at different stages of maturity and presented the probabilities of each pixel belonging to a specific un-labelled maturity stage. With the non-availability of labelled training examples for ripeness, we limited our analysis to only unsupervised learning algorithms. On the other hand, poor quality of image detections (or badly illuminated image data) could also affect the decisions on ripeness. This should be addressed with techniques mentioned in the Section 3.8.

Although we could cluster and classify pixel colors, our system couldn't precisely output the stage of maturity of the given fruit. At the moment, this decision is limited to human perception and is going to be incorporated in our future project's scope by gathering labels for ripeness and building a robust classifier.

Chapter 5

Conclusion

The purpose of this thesis is to develop an object detection, counting and ripeness analysis framework for mango orchards by interpreting the raw digital images. This chapter starts with a synopsis of the various chapters of this thesis followed by the recommendations on field deployment of models drawing upon the lessons learned from the literature and then finishes with a discussion on the future implications of this project.

5.1 Summary

Starting with a strong motivation behind the thesis, Chapter 1 clearly defined the aims and objectives that demand the need to address the two research questions to achieve them completely. Then it discussed the key contributions in the light of the inevitable limitations due to various reasons.

Sticking to the computer vision based object detection approaches, the Chapter 2 presents a coherent literature spanning the choice of sensors, feature engineering and feature learning techniques, machine learning frameworks for orchards. Various established methodologies for image based fruit detection in orchards for different fruits and conditions are studied and reported. Despite the advances in computer vision within the object detection, adoption of these techniques in the orchards for vision based tasks is still limited. This chapter addresses this gap by exploiting state-of-the-art object detection algorithms and customising it to suit the dynamic conditions in orchards thus motivating the further chapters.

Chapter 3 devises a research design keeping in mind the key gaps to fill in the literature. In the way, it proposes the research methods and justifies them with a reflection from the literature. Sources of data and collection methods are presented, augmentation and

pre-processing techniques are introduced to address the problems of over-fitting and low contrast respectively. It then details every step that revolved around these methods from setting up the environment, installing the sources to running the models. Following this, relevant metrics for evaluating object detection models are briefly introduced. It then sets out the steps to be followed for testing the models.

Chapter 4 investigates for a better performing model by presenting a comparative study on the methods chosen. It follows the procedures and metrics set in the Chapter 3 to evaluate the models. It concludes with a comprehensive discussion on the better performing models to answer the research questions in the Section 1.3.

This thesis developed frameworks for fruit detection, counting and ripeness analysis that are crucial for growers to efficiently plan the in-farm operations like yield estimation and proper time to harvest etc. The proposed research methods are the first of its kind in the orchard fruit detection tasks.

5.2 Recommendations for Field Deployment

This section discusses various practical aspects to be considered before deploying the developed frameworks on a field robot at orchards and puts forward recommendations drawn from the literature and the experiments conducted within this thesis.

5.2.1 Sensor Data

For dynamic and unstructured outdoor orchards specifically the Mango orchards, designing adaptable and robust frameworks for fruit detection is a challenging task. Quality of data plays a key role in the performance of the models regardless of the model architectures.

Details like resolution of captured image, depth of the scene, fruit size, light conditions and camera exposure time have to be carefully considered while capturing images for fruit detection [3]. A few of these conditions are tested with the best performing model which can be seen through the Figure 5.1 where the detector is only able to detect fruit of a size range from among the fruits from near and at a depth.

To curb the occlusion problems, multi-view approach can be used to reveal fruits obstructed by the parts of tree [61]. In multi-view approach, the authors captured the right, centre and left view of the tree for better detection of fruits. For example, consider a tree in the Figure 5.2 that has the fruits occluded while capturing through the camera centre view and got them disclosed while viewing it from the right. Hence multi-view

Figure 5.1: Figure showing the significance of depth of scene and size of the fruit to be well detected

approach helps mitigate the false negatives count but comes with an additional complexity to avoid multi-view registration of a fruit (counting a same fruit multiple times).

5.2.2 Timing Data Acquisition

For all orchards, timing the data acquisition plays an important role in controlling the variations due to external illumination. When sun is in the field of view of the camera,

Figure 5.2: An example of multi-view approach in mango orchards, referenced in [61]

imaging artefacts such as lens flares and over-exposure occur on the bright days [3]. Figure 5.3 depicts this fact with detections on the same image that are captured at different elevations of the Sun. This is not desirable as the detection count varies with the time of data acquisition. Hence, techniques that are robust to these changes and allow us to gather data at any time of the day need to be explored.

5.3 Future Work

The components of this thesis are designed to address machine vision under the challenging environmental conditions imposed by orchards. Nonetheless, some of the limitations in this work could be addressed in future work leading to potential improvements in both fruit detection and ripeness analysis tasks for developing data driven approaches for decision making in farms.

The proposed frameworks in this thesis bridge the gap that existed in the machine

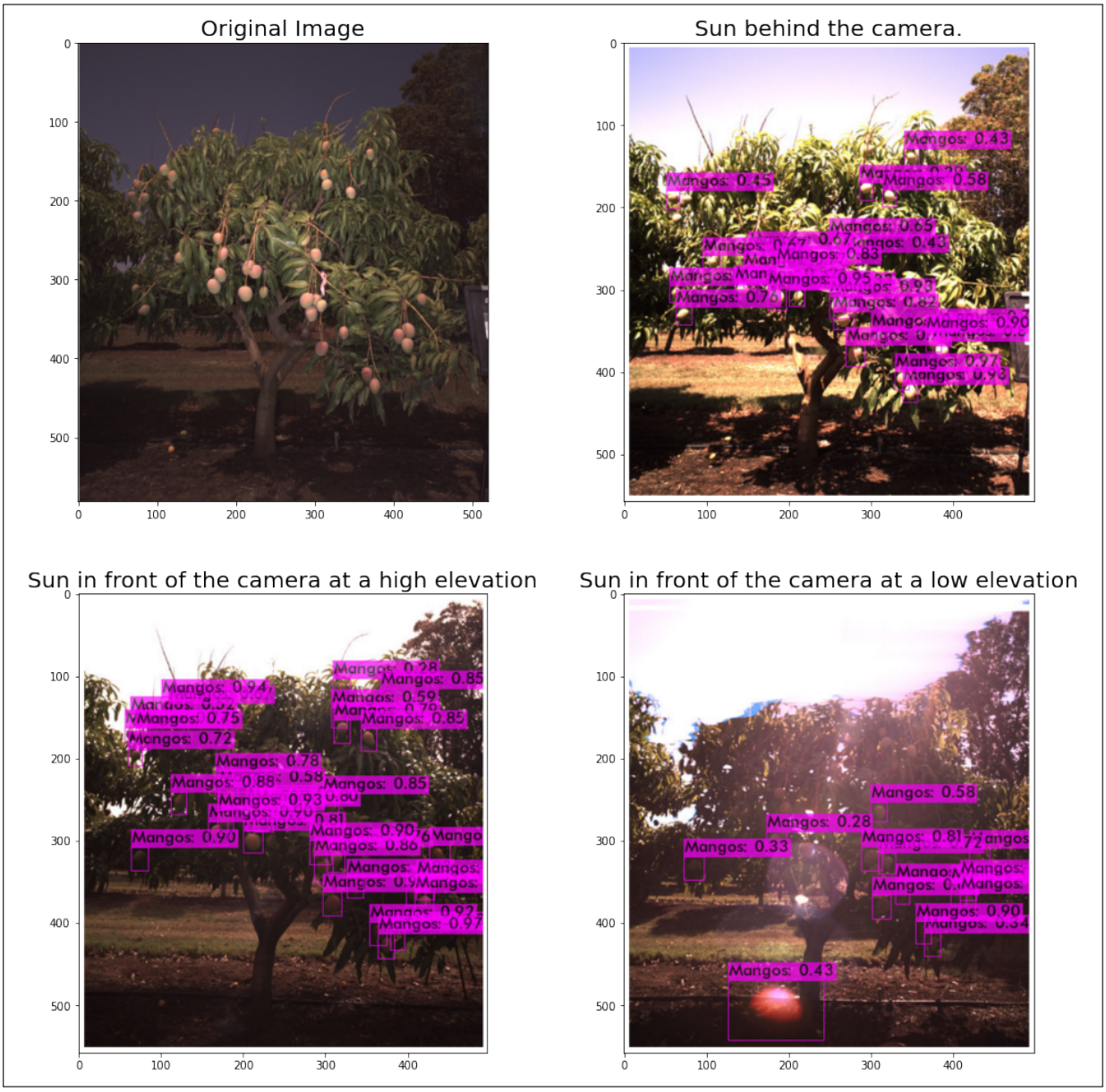


Figure 5.3: A variation in the detection count observed on same image captured at different times.

vision literature in orchards by delivering significant performance gains. In spite of our successful efforts, this performance didn't match the state-of-the-art detection accuracy in terms of both mAP and F1-Scores thus keeping us motivated to drive further. We believe that the improvements in the performance could be achieved by improving the quality of the data captured [3], targeting at robust data acquisition approaches and may be exploring 3D point cloud data or fusing data from hyper-spectral and thermal sensors [10].

Furthermore, domain knowledge in orchards could help us capture the common attributes between different orchards of the same/different fruit types and develop more generalisable detection systems that are adoptable across the orchards.

The outputs of the detection system are rectangular bounding boxes on input image where individual boxes have both fruit and non-fruit pixels. For ripeness analysis, we take each of the detections from image, hand-mask the background pixels and then analysed the fruit pixels for the probability of ripeness. This procedure gets tedious when we are trying to count ripeness probability on the whole test set of images. In future, we prioritise to automate these tasks by developing routines that backup all the detections from test dataset and iteratively filter the background based on the shape and texture of fruit pixels making use of the corresponding literature in the Section 2.1.2. Achieving this should help us the accurately answering RQ2 in the Section 1.3. Furthermore, as an extension of RQ2 we would like to broaden the idea of assessing ripeness to assessing the stage of maturity of a fruit that could help a wide range of growers with a variety of market needs.

Appendix A

Appendix

A.1 CLAHE

Below is the script for applying CLAHE on an image.

```
def histogram_equalization(img):
    lab = cv2.cvtColor(img, cv2.COLOR_RGB2LAB)
    # -----Splitting the LAB image to different channels
    # L for lightness and a and b for the color opponents green to red
   and blue to yellow.
    l, a, b = cv2.split(lab)
    # -----Applying CLAHE to L-channel
    # clipLimit  : Threshold for contrast limiting.
    # tileGridSize : Image will be divided into grids of size
   tileGridSize to apply histogram equalization
    clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8, 8))
    cl = clahe.apply(l)
    # -----Merge the CLAHE enhanced L-channel with the a and b channel
   -----------
    cl_limited = cv2.merge((cl, a, b))
    # -----Converting image from LAB Color model to RGB model
   --------------------
    transformed_image = cv2.cvtColor(cl_limited, cv2.COLOR_LAB2RGB)

    return transformed_image
```

A.2 Ripeness Analysis

```
#!/usr/bin/env python
# coding: utf-8

from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist
import matplotlib.pyplot as plt
import argparse
import numpy as np
import cv2
import matplotlib.gridspec as gridspec
import glob
import os
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import pyplot
from PIL import Image
from matplotlib import cm
from matplotlib import colors
from sklearn.mixture import GaussianMixture
from matplotlib.patches import Ellipse

import scipy.stats
from sklearn import mixture

import pandas as pd


def plot(image):
    plt.figure(figsize=(8, 8))
    plt.imshow(image, interpolation='nearest')


# In[10]:


def RGB(red,green,blue): return '#%02x%02x%02x' % (red,green,blue)


# In[11]:


def colorPixels(img):
    img = cv2.imread(img)
```

```
43      img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
44      #plt.imshow(img)
45      pixel_colors = img.reshape((np.shape(img)[0]*np.shape(img)[1], 3))
46      norm = colors.Normalize(vmin=-1.,vmax=1.)
47      norm.autoscale(pixel_colors)
48      pixel_colors = norm(pixel_colors).tolist()
49
50      return pixel_colors
51
52
53  # In[12]:
54
55
56  def GMM_Cluster_Prob(imgData, clusters):
57
58      gmm_ = GaussianMixture(n_components = clusters)
59      labels = gmm_.fit_predict(imgData)
60
61      centers = np.empty(shape=(gmm_.n_components, imgData.shape[1]))
62
63      for i in range(gmm_.n_components):
64          density = scipy.stats.multivariate_normal(cov=gmm_.covariances_
        [i], mean=gmm_.means_[i]).logpdf(imgData)
65          centers[i, :] = imgData[np.argmax(density)]
66
67      #print("Posterior probability of data belongingness to each
        Gaussian")
68      probs = gmm_.predict_proba(imgData)
69      probs = probs.round(3)
70
71      return labels, centers, probs
72
73
74  # In[156]:
75
76
77  def plotColorHist(image_newdf):
78      clusters = image_newdf["Cluster Labels"].tolist()
79      count    = image_newdf["#Pixels"].tolist()
80      colors   = [ RGB(r, g, b) for r, g, b in zip(image_newdf["Mean R"],
        image_newdf["Mean G"], image_newdf["Mean B"]) ]
81
```

```
82      #print("The plot below shows the dominance of clusters interms of
        their number and color.", colors)
83
84      plt.figure(figsize=(16,8))
85      ax1 = plt.subplot(121, aspect='equal')
86      image_newdf.plot(kind='pie', y = 'Updated Probabilities', ax=ax1,
        autopct='%1.1f%%', colors = colors,
87        startangle=90, shadow=False, labels=image_newdf['Cluster Labels'],
        legend = True, fontsize=14)
88
89
90      #ax1 = plt.subplot(122, aspect='equal')
91      '''image_newdf.plot(kind='pie', y = 'Pixel belonging probability',
        ax=ax1, autopct='%1.1f%%', colors = colors,
92        startangle=90, shadow=False, labels=image_newdf['Cluster Labels'],
        legend = True, fontsize=14)'''
93
94      #Updated Probabilities...are
95      '''width = 0.90
96      fig, ax = plt.subplots()
97      a = ax.bar(clusters, count, width, color = colors) # plot a vals
98
99      plt.show() '''
100     return colors
101
102
103 # In[40]:
104
105
106 def GenerateProb(prob):
107     meanProb = prob.mean(axis = 0)
108     #print("Mean probability of pixel belongingness to each gaussian
        cluster.")
109     #print(meanProb)
110     return meanProb
111
112
113 # In[157]:
114
115
116 def createDataFrame(image_df, probabilities):
117     image_newdf = pd.DataFrame()
```

```
118     sortedLabels = sorted(image_df['Cluster Labels'].unique())
119     #print("A new Dataframe with Mean R, G, B values and Pixel count
    grouped by their cluster labels.")
120     for label in sortedLabels:
121         d = image_df[image_df['Cluster Labels'] == label]
122         count = d[0].count()
123         temp = pd.DataFrame(
124             {
125                 'Cluster Labels': "Cluster " + str(label),
126                 '#Pixels': count,
127                 'Mean R' : int(d[0].mean()),
128                 'Mean G' : int(d[1].mean()),
129                 'Mean B' : int(d[2].mean()),
130                 'Pixel belonging probability' : probabilities[label]
131             }, index=[0]
132         )
133         image_newdf = pd.concat([image_newdf, temp], ignore_index=True)
134
135
136     mask_cluster_index = image_newdf[((image_newdf['Mean R'] == 0) &(
    image_newdf['Mean G'] == 0) & (image_newdf['Mean B'] == 0))].index.
    values
137
138     print(mask_cluster_index[0])
139     print("Original DF")
140     print(image_newdf)
141     new_image_newdf = image_newdf.drop(mask_cluster_index[0])
142
143     total = new_image_newdf['#Pixels'].sum()
144     print(total)
145     print("Dropped DF")
146     print(new_image_newdf)
147
148     pixels_list = new_image_newdf['#Pixels'].tolist()
149     print(pixels_list)
150
151     update_prob = [ t/total for t in pixels_list ]
152     print(update_prob)
153
154     new_image_newdf['Updated Probabilities'] = update_prob
155
156
```

```
157     print("New DF")
158     print(new_image_newdf)
159 
160     #print("mask_cluster_index is : ", mask_cluster_index)
161 
162     return new_image_newdf
163 
164 
165 # In[42]:
166 
167 
168 def scatterPlot(centers, reshapedImage, colors):
169     fig2 = plt.figure(figsize = (12, 50))
170     axis = fig2.add_subplot(4, 1, 1, projection="3d")
171     axis.scatter(reshapedImage[:, 0], reshapedImage[:, 1],
    reshapedImage[:, 2], facecolors= colors, marker=".")
172     axis.scatter(centers[:, 0], centers[:, 1], centers[:, 2], marker="o
    ", facecolor ='RED',  s = 200)
173 
174     plt.show()
175 
176 
177 # # Half-Ripe Mango Clustering Analysis
178 
179 # In[161]:
180 
181 
182 from matplotlib import colors
183 
184 imagePath = "halfRipe.png"
185 clusters = 6
186 
187 mng_img = cv2.imread(imagePath)
188 mng_img = cv2.cvtColor(mng_img, cv2.COLOR_BGR2RGB)  # COnvert to RGB
    COlor Space.
189 
190 plot(mng_img)
191 
192 mng_img_re = mng_img.reshape((mng_img.shape[0] * mng_img.shape[1], 3))
     # Reshape the 3-channel image to cluster
193 
194 labels, centers, probs = GMM_Cluster_Prob(mng_img_re, clusters)
```

```
195
196 image_df = pd.DataFrame(mng_img_re)
197 # Add the Cluster Labels column to Reshaped Image dataframe
198 image_df['Cluster Labels']= labels
199 # Picks the color pof pixels for scatter plot
200 colors_ = colorPixels(imagePath)
201
202 scatterPlot(centers, mng_img_re, colors_)
203 # Generates the mean probability that a pixel belongs to a cluster
204 probability_list = GenerateProb(probs)
205 # Creates a New Dataframe with #Pixels corresponding to each cluster
    and their mean R, G, B values.
206 updated_image_df = createDataFrame(image_df, probability_list)
207 # Plots the n(= clusters) dominant colors in the image.
208 plotColorHist(updated_image_df)
209
210
211
212 # # Ripe Mango Clustering Analysis
213
214 # In[160]:
215
216
217 from matplotlib import colors
218
219 imagePath = "fullyRipe.png"
220 clusters = 6
221
222 mng_img = cv2.imread(imagePath)
223 mng_img = cv2.cvtColor(mng_img, cv2.COLOR_BGR2RGB)  # COnvert to RGB
    COlor Space.
224
225 plot(mng_img)
226
227 mng_img_re = mng_img.reshape((mng_img.shape[0] * mng_img.shape[1], 3))
     # Reshape the 3-channel image to cluster
228
229 labels, centers, probs = GMM_Cluster_Prob(mng_img_re, clusters)
230
231 image_df = pd.DataFrame(mng_img_re)
232 # Add the Cluster Labels column to Reshaped Image dataframe
233 image_df['Cluster Labels']= labels
```

```
# Picks the color pof pixels for scatter plot
colors_ = colorPixels(imagePath)

scatterPlot(centers, mng_img_re, colors_)

# Generates the mean probability that a pixel belongs to a cluster
probability_list = GenerateProb(probs)
# Creates a New Dataframe with #Pixels corresponding to each cluster
    and their mean R, G, B values.
updated_image_df = createDataFrame(image_df, probability_list)
# Plots the n(= clusters) dominant colors in the image.
plotColorHist(updated_image_df)


# # Raw Mango Clustering Analysis

# In[159]:


from matplotlib import colors

imagePath = "Raw.png"
clusters = 6

mng_img = cv2.imread(imagePath)
mng_img = cv2.cvtColor(mng_img, cv2.COLOR_BGR2RGB)  # COnvert to RGB
    COlor Space.

plot(mng_img)

mng_img_re = mng_img.reshape((mng_img.shape[0] * mng_img.shape[1], 3))
     # Reshape the 3-channel image to cluster

labels, centers, probs = GMM_Cluster_Prob(mng_img_re, clusters)

image_df = pd.DataFrame(mng_img_re)
# Add the Cluster Labels column to Reshaped Image dataframe
image_df['Cluster Labels']= labels
# Picks the color pof pixels for scatter plot
colors_ = colorPixels(imagePath)

scatterPlot(centers, mng_img_re, colors_)
```

```
273
274 # Generates the mean probability that a pixel belongs to a cluster
275 probability_list = GenerateProb(probs)
276 # Creates a New Dataframe with #Pixels corresponding to each cluster
    and their mean R, G, B values.
277 updated_image_df = createDataFrame(image_df, probability_list)
278 # Plots the n(= clusters) dominant colors in the image.
279 plotColorHist(updated_image_df)
280
281 # In[158]:
282
283
284 from matplotlib import colors
285
286 imagePath = "Raw.png"
287 clusters = 6
288
289 mng_img = cv2.imread(imagePath)
290 mng_img = cv2.cvtColor(mng_img, cv2.COLOR_BGR2RGB)  # COnvert to RGB
    COlor Space.
291
292 #plot(mng_img)
293
294 mng_img_re = mng_img.reshape((mng_img.shape[0] * mng_img.shape[1], 3))
     # Reshape the 3-channel image to cluster
295
296 labels, centers, probs = GMM_Cluster_Prob(mng_img_re, clusters)
297
298 image_df = pd.DataFrame(mng_img_re)
299 # Add the Cluster Labels column to Reshaped Image dataframe
300 image_df['Cluster Labels']= labels
301 # Picks the color pof pixels for scatter plot
302 colors_ = colorPixels(imagePath)
303
304 #scatterPlot(centers, mng_img_re, colors_)
305
306 # Generates the mean probability that a pixel belongs to a cluster
307 probability_list = GenerateProb(probs)
308
309 # Creates a New Dataframe with #Pixels corresponding to each cluster
    and their mean R, G, B values.
310 updated_image_df = createDataFrame(image_df, probability_list)
```

```
311 # Plots the n(= clusters) dominant colors in the image.
312 #mask_cluster_index = image_newdf[((image_newdf['Mean R'] == 0) &(
        image_newdf['Mean G'] == 0) & (image_newdf['Mean B'] == 0))].index
313
314 print(updated_image_df.to_latex(index=False))
315
316 # plot chart
317 cluster_colors = plotColorHist(updated_image_df)
318 #print(cluster_colors)
```

Bibliography

[1] C. Akin, M. Kirci, E. O. Gunes, and Y. Cakir, *Detection of the pomegranate fruits on tree using image processing*, in 2012 First International Conference on Agro- Geoinformatics (Agro-Geoinformatics), 2012, pp. 1–4.

[2] P. Annamalai, W. Lee, and T. Burks, *Color vision system for estimating citrus yield in real-time*, in ASAE Annual International Meeting., 2004.

[3] S. Bargoti, *Fruit Detection and Tree Segmentation for Yield Mapping in Orchards*, doctor of philosophy ph.d., 2017-02-03.

[4] S. Bargoti and J. Underwood, *Deep fruit detection in orchards*, in 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2017, pp. 3626–3633.

[5] E. Barnea and O. Ben-Shahar, *Depth based fruit detection from viewer-based pose*, in Proceedings of the AgEng'14 Conference (Zurich, Switzerland). Paper, volume 137., 2014.

[6] M. Bergerman, J. Billingsley, J. Reid, and E. van Henten, *Robotics in Agriculture and Forestry*, Springer International Publishing, Cham, 2016, pp. 1463–1492.

[7] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, *Yolov4: Optimal speed and accuracy of object detection*, arXiv preprint arXiv:2004.10934, (2020).

[8] G. Bradski, *The OpenCV Library*, Dr. Dobb's Journal of Software Tools, (2000).

[9] J. Brezmes, M. L. L. Fructuoso, E. Llobet, X. Vilanova, I. Recasens, J. Orts, G. Saiz, and X. Correig, *Evaluation of an electronic nose to assess fruit ripeness*, IEEE Sensors Journal, 5 (2005), pp. 97–108.

[10] D. Bulanon, T. Burks, and V. Alchanatis, *Image fusion of visible and thermal images for fruit detection*, Biosystems Engineering, 103 (2009), pp. 12 – 22.

[11] J. Cartucho, R. Ventura, and M. Veloso, *Robust object recognition through symbiotic deep learning in mobile robots*, in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 2336–2341.

[12] S. Chaivivatrakul and M. N. Dailey, *Texture-based fruit detection*, Precision Agriculture, 15 (2014), pp. 662–683.

[13] H. Cheng, X. Jiang, Y. Sun, and J. Wang, *Color image segmentation: advances and prospects*, Pattern Recognition, 34 (2001), pp. 2259 – 2281.

[14] M. Chhabra, A. Gupta, P. Mehrotra, and S. Reel, *Automated Detection of Fully and Partially Riped Mango by Machine Vision*, vol. 131, 01 2012, pp. 153–164.

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *ImageNet: A Large-Scale Hierarchical Image Database*, in CVPR09, 2009.

[16] P. Dollar, C. Wojek, B. Schiele, and P. Perona, *Pedestrian detection: An evaluation of the state of the art*, IEEE transactions on pattern analysis and machine intelligence, 34 (2011), pp. 743–761.

[17] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, *The pascal visual object classes (voc) challenge*, International journal of computer vision, 88 (2010), pp. 303–338.

[18] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, *Object detection with discriminatively trained part-based models*, IEEE transactions on pattern analysis and machine intelligence, 32 (2009), pp. 1627–1645.

[19] D. Font, T. Pallejà, M. Tresanchez, M. Teixidó, D. Martinez, J. Moreno, and J. Palacín, *Counting red grapes in vineyards by detecting specular spherical reflection peaks in rgb images obtained at night with artificial illumination*, Computers and Electronics in Agriculture, 108 (2014), pp. 105 – 111.

[20] A. Gongal, S. Amatya, M. Karkee, Q. Zhang, and K. Lewis, *Sensors and systems for fruit detection and localization: A review*, Computers and Electronics in Agriculture, 116 (2015), pp. 8–19.

[21] C. Guo, F. Liu, W. Kong, Y. He, B. Lou, et al., *Hyperspectral imaging analysis for ripeness evaluation of strawberry with support vector machine*, Journal of Food Engineering, 179 (2016), pp. 11–18.

[22] M. Hannan, T. Burks, and D. Bulanon, *A machine vision algorithm combining adaptive segmentation and shape analysis for orange fruit detection*, Agricultural Engineering International : The CIGR e-journal, 0 (2010).

[23] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[24] J. D. Hunter, *Matplotlib: A 2d graphics environment*, Computing in Science & Engineering, 9 (2007), pp. 90–95.

[25] A. Jimenez, R. Ceres, and J. L. Pons, *A survey of computer vision methods for locating fruit on trees*, Transactions of the ASAE, 43 (2000), p. 1911.

[26] K. Kapach, E. Barnea, R. Mairon, Y. Edan, and O. Ben-Shahar, *Computer vision for fruit harvesting robots - state of the art and challenges ahead*, Int. J. Comput. Vis. Robotics, 3 (2012), pp. 4–34.

[27] E. E. Kelman and R. Linker, *Vision-based localisation of mature apples in tree images using convexity*, Biosystems Engineering, 118 (2014), pp. 174 – 185.

[28] A. Koirala, K. B. Walsh, Z. Wang, and N. Anderson, *Deep learning for mango (mangifera indica) panicle stage classification*, Agronomy, 10 (2020), p. 143.

[29] T. Kurbiel, *How to efficiently implement Area Under Precision-Recall Curve (PR-AUC)*, MAY 05, 2020.
https://towardsdatascience.com/how-to-efficiently-implement-area-under-precision-re

[30] F. Kurtulmus, W. S. Lee, and A. Vardar, *Immature peach detection in colour images acquired in natural illumination conditions using statistical classifiers and neural network*, Precision agriculture, 15 (2014), pp. 57–79.

[31] H. Lalel, Z. Singh, S. Tan, and M. Agustí, *Maturity stage at harvest affects fruit ripening, quality and biosynthesis of aroma volatile compounds in 'kensington pride'mango*, The Journal of Horticultural Science and Biotechnology, 78 (2003), pp. 225–233.

[32] Y. LeCun, Y. Bengio, et al., *Convolutional networks for images, speech, and time series*, The handbook of brain theory and neural networks, 3361 (1995), p. 1995.

[33] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, *Handwritten digit recognition with a back-propagation network*, in Advances in neural information processing systems, 1990, pp. 396–404.

[34] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, *Microsoft coco: Common objects in context*, in European conference on computer vision, Springer, 2014, pp. 740–755.

[35] 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, 122 (2016), pp. 118–123.

[36] P. Mahto, P. Garg, P. Seth, and J. Panda, *Refining yolov4 for vehicle detection*, International Journal of Advanced Research in Engineering and Technology (IJARET), 11 (2020).

[37] 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), 2016, pp. 2506–2512.

[38] O. Mirbod, L. Yoder, and S. Nuske, *Automated measurement of berry size in images*, IFAC-PapersOnLine, 49 (2016), pp. 79–84.

[39] B. Namdari Gharaghani, H. Maghsoudi, and M. Mohammadi, *Ripeness detection of orange fruit using experimental and finite element modal analysis*, Scientia Horticulturae, 261 (2020), p. 108958.

[40] S. Nuske, S. Achar, T. Bates, S. Narasimhan, and S. Singh, *Yield estimation in vineyards by visual grape detection*, in 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, pp. 2352–2358.

[41] S. Nuske, K. Wilshusen, S. Achar, L. Yoder, S. Narasimhan, and S. Singh, *Automated visual yield estimation in vineyards*, Journal of Field Robotics, 31 (2014), pp. 837–860.

[42] M. S. Padda, C. V. do Amarante, R. M. Garcia, D. C. Slaughter, and E. J. Mitcham, *Methods to analyze physico-chemical changes during mango ripening: A multivariate approach*, Postharvest Biology and Technology, 62 (2011), pp. 267–274.

[43] S. J. Pan and Q. Yang, *A survey on transfer learning*, IEEE Transactions on Knowledge and Data Engineering, 22 (2010), pp. 1345–1359.

[44] T. pandas development team, *pandas-dev/pandas: Pandas*, Feb. 2020.

[45] A. Payne and K. Walsh, *Machine vision in estimation of fruit crop yield*, Plant Image Analysis: Fundamentals and Applications; CRC Press: Boca Raton, FL, USA, (2014), pp. 329–374.

[46] 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, 100 (2014), pp. 160 – 167.

[47] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research, 12 (2011), pp. 2825–2830.

[48] S. M. Pizer, E. P. Amburn, J. D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B. ter Haar Romeny, J. B. Zimmerman, and K. Zuiderveld, *Adaptive histogram equalization and its variations*, Computer vision, graphics, and image processing, 39 (1987), pp. 355–368.

[49] S. K. J. S. S. R. D. J. V. K. Pooja Ravindra Kamble, Rudrakshi Sanjeev Marathe, *Development of an effective system to identify fruit ripening stage for apple, banana and mango*, International Journal of Advanced Science and Technology, 29 (2020), pp. 2766–2772.

[50] 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), 2016, pp. 5171–5176.

[51] ——, *Texture-based fruit detection via images using the smooth patterns on the fruit*, in 2016 IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 5171–5176.

[52] J. Redmon, *Darknet: Open source neural networks in c.* `http://pjreddie.com/darknet/`, 2013–2016.

[53] M. Regunathan and W. S. Lee, *Citrus fruit identification and size determination using machine vision and ultrasonic sensors*, in 2005 ASAE Annual Meeting, American Society of Agricultural and Biological Engineers, 2005, p. 1.

[54] A. Roy, S. Banerjee, D. Roy, and A. Mukhopadhyay, *Statistical video tracking of pomegranate fruits*, in 2011 Third National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics, IEEE, 2011, pp. 227–230.

[55] I. Sa, C. McCool, C. Lehnert, and T. Perez, *On visual detection of highly-occluded objects for harvesting automation in horticulture*, in ICRA 2015 : IEEE International Conference on Robotics and Automation, 2015.

[56] R. Serraj and P. Pingali, *Agriculture Food Systems to 2050: Global Trends, Challenges and Opportunities*, 01 2019.

[57] A. W. Setiawan, T. R. Mengko, O. S. Santoso, and A. B. Suksmono, *Color retinal image enhancement using clahe*, in International Conference on ICT for Smart Society, 2013, pp. 1–3.

[58] J. Solawetz, *Breaking Down YOLOv4*, JUN 04, 2020. `https://blog.roboflow.com/a-thorough-breakdown-of-yolov4/`.

[59] Y. Song, C. Glasbey, G. Horgan, G. Polder, J. Dieleman, and G. Van der Heijden, *Automatic fruit recognition and counting from multiple images*, Biosystems Engineering, 118 (2014), pp. 203–215.

[60] D. Stajnko and Z. Cmelik, *Modelling of apple fruit growth by application of image analysis*, Agriculturae Conspectus Scientificus (ACS), 70 (2005).

[61] M. Stein, *Improving image based fruitcount estimates using multiple view-points*, 2016.

[62] M. Tan, R. Pang, and Q. V. Le, *Efficientdet: Scalable and efficient object detection*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 10781–10790.

[63] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*, CreateSpace, Scotts Valley, CA, 2009.

[64] J. P. Wachs, H. I. Stern, T. Burks, and V. Alchanatis, *Low and high-level visual feature-based apple detection from multi-modal images*, Precision Agriculture, 11 (2010), pp. 717–735.

[65] C.-Y. Wang, H.-Y. Mark Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, *Cspnet: A new backbone that can enhance learning capability of cnn*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 390–391.

[66] E. Westerberg, *AI-based Age Estimation using X-ray Hand Images: A comparison of Object Detection and Deep Learning models*, PhD thesis, 2020.
Presentationen gjordes online via Zoom.

[67] G. Yadav, S. Maheshwari, and A. Agarwal, *Contrast limited adaptive histogram equalization based enhancement for real time video system*, in 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2014, pp. 2392–2397.