Agricultural Produce Grading by Computer Vision Based on Genetic Programming

Panitnat Yimyam

A thesis submitted for the degree of
Doctor of Philosophy

School of Computer Science and Electronic Engineering
University of Essex

May, 2015

Abstract

An objective of computer vision is to imitate the ability of the human visual system. Computer vision has been put forward to produce a wide range of applications. Most vision software does not proceed alone; machine learning is usually involved in many vision systems. Some vision systems are developed to replace human working because they operate more reliably, precisely and speedily, and some tasks are dangerous for humans.

This thesis presents contributions to extend a vision system based on genetic programming to solve classification problems. Instances in the field of agricultural produce are employed to verify the system performance. A new method is proposed to determine the shape and appearance of reconstructed 3D objects. The reconstruction is based on using 2D images taken by a few cameras in arbitrary positions. Furthermore, new techniques are presented to extract properties of 3D objects; morphological, coloured and textural features.

New techniques are proposed to incorporate new features and new classes of samples into a GP classifier. For the former, the new feature is accommodated into an existing solution by mutation. For the latter, as generating a multi-class classifier is based on a binary decomposition approach, a binary classifier of the new class is produced and executed before the series of the original binary classifiers. Both cases are intended to be done with less computation than evolving a new classifier from scratch.

Acknowledgements

I would like to express my deep gratitude to Dr. Adrian F. Clark, my supervisor, for his insightful suggestions, valuable help and constant support during my PhD.

My grateful thanks are also extended to Prof. John Gan and Dr. John Woods for their useful critiques and advice in supervisory board meetings.

I am particularly grateful for funding by Royal Thai Government Scholarship and supporting by Burapha University Sakaeo Campus.

Special thanks for great advice should be given to Dr. Somkit Jaitrong, Dr. Sarita Pinmanee, Assoc. Prof. Panmanas Sirisomboon, Assist. Prof. Thanarat H. Chalidabhongse, Assist. Prof. Sunisa Rimcharoen and Sirinapha Phungpreeda.

I wish to thank Dr. Olly Oechsle who developed Jasmine leading to this thesis. I would like to offer my special thanks to my colleagues, Dr. Nadia Kanwal, Dr. Gazi Erkan Bostanci, Wichit Sombat, Suthira Plansangket, Santi Termprasertsakul, Warawit Phetruen and Felix Ngobigha, for fruitful suggestions and being my great friends. I would also like to thank the staff of the School of Computer Science and Electronic Engineering.

Finally, I most gratefully acknowledge my father, my aunt and my cousin for their encouragement and love as well as my late mother who is my inspiration and always in my heart.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Humans try to survive and prefer to live comfortably in the present environment. They attempt to generate and adapt many things to raise their lives better. Computer technology is a domain that is being developed to help humans to solve real-world problems. For example, operating systems are produced in various options and improved to support user requirements for processing digital data. Computer networking is steadily enhancing its performance to deal with wider connection and higher speed of communication. Computer applications are continuously being programmed to respond to various user domains.

Computer vision is associated with a wide range of tasks, such as medicine, industrial manufacturing and the entertainment industry. Some vision machines are developed to replace human working because they are able to perform more speedily, precisely and consistently than humans, and some circumstances are dangerous for humans. Vision applications are developed to comfort or entertain

humans, such as aids for the visually impaired, virtual navigation and visual games. Robotics is constantly put forward to mimic behaviors of living things, and computer vision is an important part of this subject. Smartphones and tablets are compact devices that are now intimately involved in humans' everyday lives, and they also contain various vision applications. Some vision systems are developed to respond to users' requirements in real time, such as sign translation, moving object tracking and surveillance monitoring.

An important piece of equipment for a vision system is a camera serving as the 'eye' of the machine. Some devices employ only one camera to recognize the scene in two dimensions (2D), whereas three dimensional (3D) object analysis or recognition involving the depths of objects needs multiple cameras. Most vision systems are able to make a decision automatically because they cooperate with an artificial intelligence technique to solve the problem efficiently. Machine learning is an approach frequently applied to computer vision tasks. Many machine learning techniques are available, and they have been put forward to result in better accuracy for making decisions. The fundamentals of computer vision and machine learning are described as follows.

1.1 Computer Vision

If the meaning of vision is to know the world by looking, then computer vision (also called machine vision) is a computational system with knowledge fed from input information for performing a vision task. Computer vision aims to duplicate the result of human vision by understanding images. Humans live in the 3D world, but images lose information due to the projection process from 3D to 2D properties. Therefore, it is desirable that a vision system should be capable of recovering

any 3D presentation of objects. 3D properties of an object can be recovered by employing two or more images captured around an object. The images can be binary, grey-scaled or colour taken by a single camera or from multiple viewpoints.

Computer vision involves techniques for geometric modelling and cognitive processing. It also encompasses techniques such as image processing and machine learning. Vision applications usually employ image processing algorithms in their early phases in order to enhance the input. Image processing modifies a digital image to generate a new image different in some respects from the original image. It can be considered as low-level processing to deal with images in detail or pixel by pixel. Image processing methods can be divided into three broad categories: image enhancement, image restoration and image compression [1]. Image enhancement involves a wide range of techniques to improve the visual quality of images responding to human viewers, such as contrast and brightness adjustments. Image restoration aims to remove or reduce degradations, *i.e.* blur, image motion or noise from photometric sources. In addition, image compression can reduce the size of images by eliminating redundant or irrelevant information.

Conversely, computer vision performs high-level processing— understanding the content of images in terms of semantics based on knowledge like humans involving geometric and material properties. Instances of geometric properties contain shape, size and positions of objects; examples of material properties include colour, texture and surface characteristics. Machine learning is incorporated into computer vision in order to make predictions or decisions from its experience. It is compared to the intelligence of systems, and it plays a key role in feature extraction and object recognition for vision systems.

In order to build a vision machine, several issues have to be concerned. For example, illumination conditions are always a critical factor to have an effect on

shadows. A scene needs to be setup properly to reduce the complexity of images. Even object positions have to be appropriately located to avoid object overlapping, for some types of processing. Some invented vision machines are able to manage several input types; in contrast, some systems are developed for a particular type of problems.

1.2 Machine Learning

Machine learning is the science of getting a computer to operate based on its own experience, and is a subset of artificial intelligence [2]. Machine learning is frequently applied as an important part of real-world applications, such as robotic control, industrial process control, speech recognition, optical character recognition and web searching.

An overview of the machine learning process starts with the identification of a learning domain, and it finishes with testing the result of the learning process. A learning domain is a set of features extracted from problem instances. Each instance is supposed to represent its features (inputs) corresponding to one class (output). Training instances are collected to use as a training set. When a training set is prepared, the learning environment of the system is identified. The learning process tries to generate a solution best able to predict the outputs of the training set from inputs. After the learning finishes, the obtained solution should be evaluated for the quality of the learning with an unseen dataset from the same problem domain. A high quality solution is supposed to cope with an unseen dataset well.

Machine learning contains two major approaches: unsupervised and supervised learning [3]. Their principles and learning methods are totally different, but they

aim to reach the same goal in order to classify input data. In some circumstances, evidence of sample types is inadequate, or the number of classes is unknown. Therefore, unsupervised learning is used to cope with the problem. It does not require the initial definition of datasets from the user; it aims to assign labels to categories after learning. The learning technique is based on finding the intrinsic structure, relations or relevance of data. Examples of unsupervised learning tasks consist of clustering and dimensionality reduction.

Conversely, supervised learning requires some external knowledge obtained from a user before implementation. The user is supposed to feed both input and output training data into the system. Broadly speaking, supervised classification works using either generative and discriminative approaches. Generative classifiers learn to model the joint probability of measured inputs and class labels whereas discriminative classifiers learn to model a direct mapping between inputs and class labels [4]. The size and quality of a training set have an effect on the quality of a generated classifier. If the training set is bigger and contains more effective training data, the classifier is likely to return better classification correctness. Nevertheless, in reality it is difficult to define how large a training size is appropriate and which data are proper for training. Sample supervised learning approaches are genetic programming, support vector machines and neural networks.

1.3 Problem Statement

Automatic systems have been put forward for a wide range of application areas. They also play a key role in various industries to use for manufacturing, quality control and produce grading, *etc.* High volume production is a major factor

(a) Passion fruit inspection

(b) Mango grading

Figure 1.1: Grading processes by humans

causing human working to be replaced by machines. Furthermore, safety and reliability derived from using machines are taken into account. Some work is hazardous for humans, and some manufactured parts need 100% accuracy.

The agricultural industry uses automatic machines to deal with grading tasks. Grading is important commercially because higher-grade produce is able to be sold at a higher price. Manual grading performed by humans is a traditional operation; however, it is time-consuming, labour-intensive and unreliable. Figure 1.1 shows passion fruit and mango grading by humans. The labourers have to inspect skin qualities and size using the same grading standard. In the photographs, they did not use a measurement tool because they were experienced enough; but if it was difficult to decide for some fruits, such as size, weighing would be used to solve the problem.

In Figure 1.1, it is noticeable that there is a huge number of fruits. Human operators may be negligent, subjective or inconsistent resulting in less accuracy. Consequently, computer vision is attractive for inspection. To date, the variations in size, colour, texture *etc.* of different types of produce means that the image analysis of each type of foodstuff must be devised independently, so a single system

that could simply be re-trained is very attractive.

Consequently, this thesis aims to develop a vision system able to solve classification problems with emphasis on agricultural inspection. A category of samples chosen to verify the system performance is agricultural produce because grading machines are now in high demand in the agricultural industry. This research modifies a vision system involving genetic programming, Jasmine [5], in order to enhance its performance to have high potential to cope with any possible classification tasks.

1.4 Contributions

The major contributions proposed in this thesis are as follows:

1. A vision system based on genetic programming was extended to work more effectively and automatically to support a wide range of classification problems (Chapter 3).

2. A VCFA (Voxel Colouring based on a Few cameras in Arbitrary positions) technique was devised to determine the shape and appearance of a 3D object reconstructed from 2D images taken by a small number of cameras in arbitrary locations (Chapter 4).

3. Techniques were presented to extract characteristics of reconstructed 3D objects including size, shape, colour and texture features (Chapter 4).

4. The performance of using features extracted from reconstructed 3D objects was compared with that of employing solely 2D features. The results generated by the GP system were compared with those produced by k-nearest neighbours, neural networks and support vector machines (Chapter 5).

5. A new way was proposed to extend an already-trained classifier with new features using mutation. If the mutated solution performed well, it was able to reduce computational cost compared with evolving a new solution from scratch (Chapter 6).

6. A technique was developed to accommodate a new class of samples into an existing classifier without having to evolve a new solution to all classes from scratch (Chapter 6).

1.5 Publications

There have been two published conference papers generated from parts of the thesis contributions:

1. Yimyam, P. and Clark, A. F. "Agricultural produce grading by computer vision using Genetic Programming." In Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on, pp. 458-463. IEEE, 2012.

   This paper demonstrated the possibility of using a computer vision system based on genetic programming, Jasmine, to solve a range of tasks in the domain of inspection of agricultural produce. The original Jasmine was modified to extend additional colour and texture operations. The experimental results proved that the added features were able to enhance the classification performance of the system. The GP performance was also compared with the performance of neural network and support vector machine classifiers. We found that the GP classifiers were superior to the others for some tasks.

2. Yimyam, P. and Clark, A. F. "Adding new features and classes to classifiers evolved using genetic programming." In Electronics, Computer and Com-

putation (ICECCO), 2013 International Conference on, pp. 224-227. IEEE, 2013.

This paper proposed two techniques to accommodate new features or new classes of samples to the already-trained classifier. Both techniques aimed to reduce the computation cost compared with evolving a new solution from scratch. The first proposed method used mutation to incorporate additional operators that introduce the new features into the existing solution. We found that the mutation process was able to deliver mutated programs that performed well enough without requiring normal GP processing for most experiments. As the GP framework generated a classifier based on a binary decomposition approach, the other proposed idea presented a technique to prepend a binary classifier distinguishing a new class from all existing classes to the sequence of existing binary classifiers. Almost all experiments showed that the proposed classifiers achieved more accuracy than performing by the *ab initio* classifier.

1.6 Thesis Structure

The thesis structure is summarized in terms of chapter relationships as shown in Figure 1.2.

Chapter 2 presents the relevant fundamentals of computer vision and reviews vision systems used for agricultural produce grading. This chapter also describes the basis of genetic programming and reviews vision applications using genetic programming, and techniques developed to improve conventional genetic programming.

Chapter 3 introduces a vision system based on genetic programming called

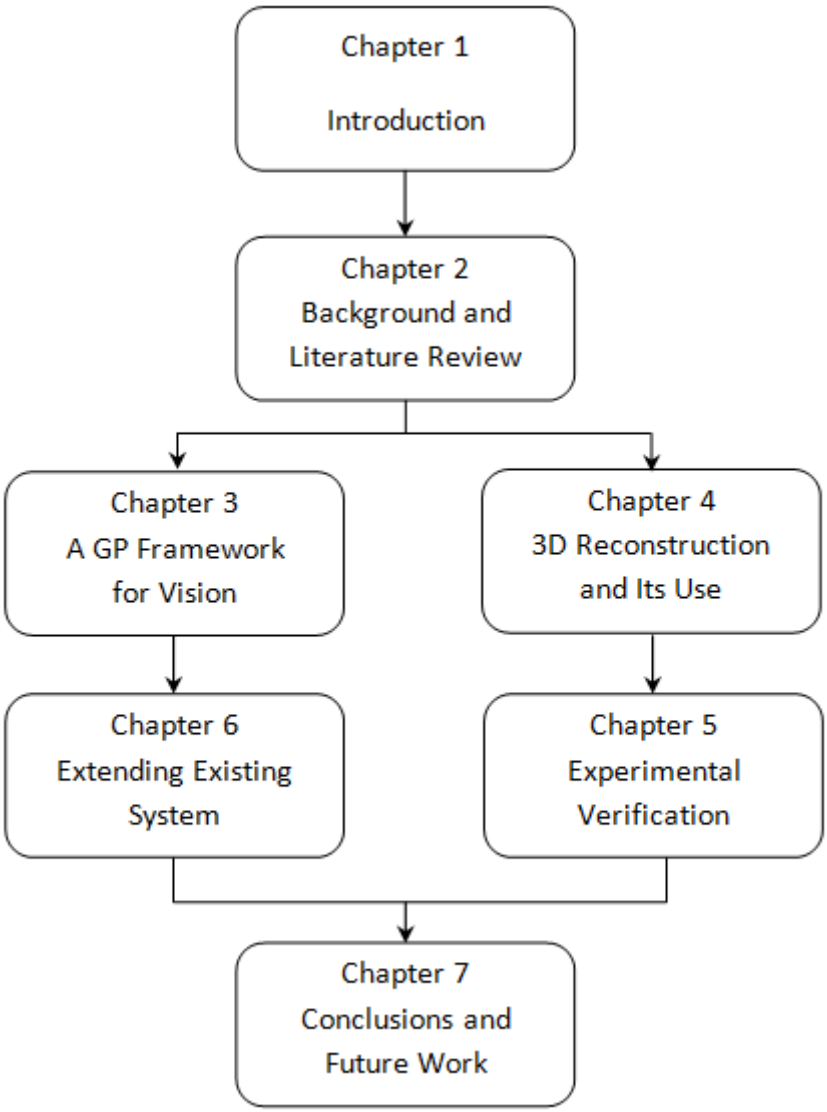


Figure 1.2: Relationships of thesis chapters

Jasmine, which was the starting point for the research described in this thesis. Its framework and shortcomings are expounded. A number of improvements are described, and the performance of the extended system is verified by experiments on general objects and agricultural produce. Its performance is compared with that of the original Jasmine using McNemar's test.

Chapter 4 presents 3D reconstruction fundamentals and reviews previous techniques. The remaining chapter presents proposed methods to determine the shape and appearance of 3D objects reconstructed from 2D images taken by a small number of cameras in arbitrary positions and to extract physical properties: morphological, coloured and textural features.

Chapter 5 verifies the performance of using 3D features extracted from the proposed techniques. Its achievement is compared with employing 2D features. McNemar's test is used to compare the performance differences, and Bonferroni correction is employed to adjust the critical value. Classification results produced by the GP framework are compared with those generated by other classification approaches: k-nearest neighbour, neural network and support vector machine.

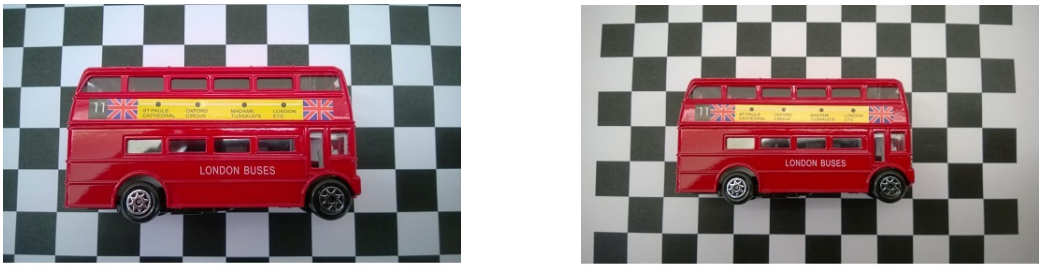
Chapter 6 presents two approaches involving genetic programming to accommodate additional features and a new class of samples into an already-trained classifier. In both concepts, extending the ability of the engine can be done with less computation than evolving a new solution from scratch. The performances of the proposed classifiers are compared with those of the original Jasmine using McNemar's test.

Chapter 7 draws conclusions of the contributions and makes suggestions for further work.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

The system that forms the focus of this thesis uses genetic programming (GP) to solve a range of classification tasks in the field of agricultural produce grading. Related theories and a survey of previous work are presented in this chapter, which is divided into four parts. The first part introduces the computer vision fundamentals associated with this thesis. The second part shows existing techniques based on vision to inspect fruit and vegetables. The third section presents the principles of GP. The final part gives examples of various tasks using GP and ideas to improve the performance of GP classifiers.



(a) Object near the viewpoint (b) Object far from the viewpoint

Figure 2.1: Demonstration on a size effect when an object lies near and far from a camera

2.1 Computer Vision Fundamentals

Computer vision aims to mimic the capabilities of the human visual system. However, computers do not have an amazing brain system like humans, and it is accepted that the richness of human foreknowledge beats computer systems. A major disadvantage of machine vision compared with a human observer, is the lack of wide and flexible knowledge of the real world.

For example, position and orientation changes generally have an effect on the appearance of 3D shapes. For instance, an object is perceived at a different size if it lies at a different distance from the camera. In other words, an object looks bigger when it is closer to a point of view; in contrast, it seems smaller if it is farther from a viewpoint. Figure 2.1 demonstrates the apparent size of a model bus laid near and far from a camera. Actually, the models in the pictures are the same object, so their size should be returned equally. Thus, in order to estimate the size of an object for this issue, it needs to use another object of known size as a reference. For this example, the checkerboard acted as background that can be used to calibrate the size of objects.

For an example of a problem on orientation changing, humans' brain systems

are able to recognize things that they have seen at different view angles. For instance, humans can recognize an aircraft seen in any direction. For 2D analysis by computer vision, an aircraft can result in many different 2D projections. Therefore, various aircraft projections have to be employed as an input set to train the system, and it does not guarantee that the system is able to recognize a new view of it. The difficulty will increase if recognition has to cope with overlapping objects.

An example showing how humans' knowledge can beat computers is assuming that you see an image of the Titanic. Certainly, you have never seen the real one before, but you may know that it was a big ship, and its sinking was a famous tragedy. However, if you have never heard about it before, you probably can predict where an object of interest is in the picture, or how a ship appears. The image may show that the ship is at a dock or in the open sea, and it is not supposed to be located on land. Humans are able to predict or recognize which one should be a ship, the sea or a dock. Conversely, much information involving the picture has to be fed to a computer system for learning. For example, the sea can be detected as a wide blue area, and the ship is probably recognized as an object with many edges. A computer has to be trained with this knowledge probably for every different task.

This research extends a vision system, Jasmine [5] using basic vision approaches, but they are practical and effective enough to deal with vision tasks. Computer vision principles relevant to this thesis and mostly used for agricultural produce grading (shown in the next section) are presented as follows.

2.1.1 Segmentation

Image segmentation or object isolation is one of the most important processes to decompose the important features of an image into regions. If a segmented object shape is close to that of the real object, it is advantageous for subsequent image analysis. Segmentation can be divided into two concepts, complete segmentation and partial segmentation [6]. Complete segmentation aims to return a set of disconnected regions that correspond to objects of interest. A segmentation algorithm is more likely to achieve complete segmentation if the characteristics of object regions are uniform and homogeneous. Conversely, partial segmentation focuses on discarding regions that are not directly related to the objects of interest, and background subtraction can be considered as this type of approach.

Background subtraction aims to distinguish objects of interest from the scene or to separate the image into regions that correspond to structural units, such as foreground and background. In other words, background subtraction removes the background from the image in order to simplify it without making an impact on elements of interest. For image processing, digital image subtraction operates on pixel values, pixel by pixel. Considering the difference of colour properties is the most commonly employed approach to deal with this issue. Furthermore, a background uniform colour is often set differently to that of the objects of interest in order to reduce the complexity of analysis.

Grey level thresholding is an established, simple method to subtract background from images. It uses global knowledge, and it is computationally fast and cheap. It converts a grey scale image into binary partitions including foreground (objects of interest) and background. The foreground is clearly separated from the background if the intensity has adequate contrast. Some images of a dataset

are used to generate a threshold. A range of grey scales is defined to indicate foreground and background. When a threshold is produced, it is implemented for the whole dataset. There are two ways to segment the background: pixels with a value within the range as being part of the foreground are kept while the other pixels are discarded; or pixels that correspond to the background range are removed while the other pixels are maintained.

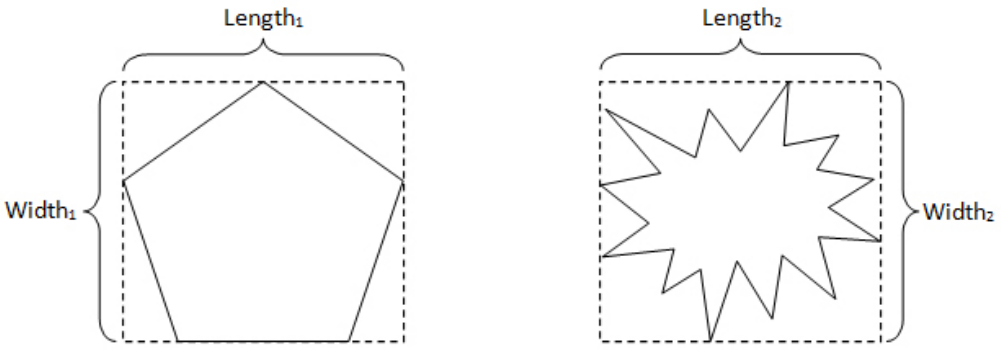
2.1.2 Morphological feature analysis

This aims to extract region-based shape descriptors regarding object shape and outline but excluding pixel properties. Morphological features can be used efficiently for object recognition. For example, shape and size properties play a key role in agricultural produce grading, such as variety identification and size sorting. Sample operations employed to extract morphological features are represented as the following. The sample functions are simple and flexible enough in order to apply to a wide range of objects.

Bounding area

The bounding area of an object can be used to represent the object size. The area depends on the boundary of the object, and it is scoped by an enclosing rectangle called a bounding box. For this research, a bounding box is defined as the top line at the minimum value of X, the bottom line at the maximum value of X, the left line at the minimum value of Y and the right line at the maximum value of Y. It is one of the possible rectangles that enclose the object completely.

A bounding box cannot imply whether object shape is simple or complex. If object shape is simple, the perimeter of the object encloses the area. Simple and



(a) Simple shape (b) Complex shape

Figure 2.2: Bounding boxes of simple and complex shapes

complex shapes sampled in Figure 2.2 result in the same value of bounding area as $Width_1 = Width_2$ and $Length_1 = Length_2$.

Number of pixels

The number of object pixels is the simplest and most natural characteristic representing an object region. This function counts the number of pixels inside and including the boundary of the object. The shapes in Figure 2.2 return the same bounding area value but different numbers of pixels; the simple shape has more pixels than the other.

Aspect ratio

Aspect ratio measurement can be simply calculated by the ratio of the width to the length of a bounding box: (2.1).

$$Aspect\ ratio = \frac{Width}{Length} \tag{2.1}$$

where $Width$ is the width of the bounding box, and $Length$ is the length of the bounding box.

Rectangularity

The rectangularity feature proves how well an object fills its bounding box. This is calculated as the ratio of the number of object pixels to the bounding box area, as shown in (2.2). R is in the range (0,1]. If R returns 1, it means that the object is a perfect rectangular shape; in contrast, if it is close to 0, the object shape is thin or curvy. This operation is invariant to scale.

$$Rectangularity = \frac{A_p}{A_b} \tag{2.2}$$

where A_p is the number of pixels of the object area and A_b is the bounding rectangle area.

Centroid

The centroid or the centre of mass is the balance position of an object. Assuming that each pixel of the object has a unit weight, the centroid is the point where the object can completely balance, or there is equal mass left, right, above and below. The centroid (x_c, y_c) involves the value of half the total pixels denoted by HT. The position of x_c is searched along the X lines, and x_c is the pixel at position HT. Similarly, y_c is searched along the Y lines, and the pixel located at position HT represents y_c.

Roundness

The roundness presents how closely the object fits a circle. The function returns the standard deviation of the distance between perimeter points and the centroid of the object. A higher output results from more irregular shape.

Balance

The balance property is employed to compare the distance between the centroid of the object and the centroid of the bounding box. The formulae of balance in X and Y axes are shown in (2.3) and (2.4) respectively.

$$Balance_X = \frac{x_c - x_b}{Width} \tag{2.3}$$

$$Balance_Y = \frac{y_c - y_b}{Length} \tag{2.4}$$

where x_c and y_c are coordinates of the centroid of the object, x_b and y_b are coordinates of the centroid of the bounding box, $Width$ is the width of the bounding box, and $Length$ is the length of the bounding box.

Symmetry

The symmetry is a measurement of how well the shape is reflected; it can be estimated in both horizontal and vertical axes. Figure 2.3(a) illustrates a symmetry evaluation in the horizontal axis and Figure 2.3(b) in the vertical axis.

(a) Horizontal axis (b) Verical axis

Figure 2.3: Shape symmetry analysis in horizontal and vertical axes

Number of edges

This operation involves a process of skeletonisation on the image. It returns the number of skeleton edges. This descriptor is scale and rotation invariant.

Number of joints

The process of skeletonisation is also involved in this function. The number of points where edges converge are counted and returned as output of the function. This descriptor is scale and rotation invariant.

Maximum and average depth

This function operates during the process of skeletonisation. It measures the depth of each pixel from the nearest edge. A value of the maximum depth is compared, and a value of average depth is calculated. This descriptor is invariant to rotation.

2.1.3 Colour analysis

Colour is a primitive physical property of any object. For the human visual system, light reflection from an object reaches a human's eyes. Signals travel to the optic

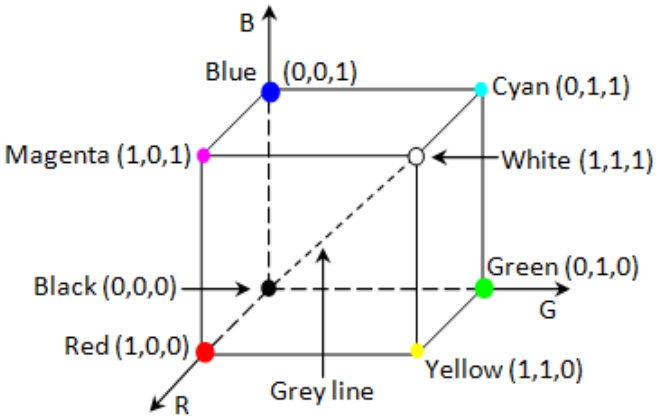


Figure 2.4: Representation of the RGB cube

nerve, and they are interpreted by the brain as colour. Various colour spaces are used in different tasks, and there are methods to convert from one colour model to another.

Colour Spaces

Different colour spaces are used in many types of work; for example, RGB is well known and generally employed in digital image processing applications. CMYK is a major colour model used primarily in the printing industry. HSI is normally implemented in computer graphic and art work. CIE L*a*b* is employed in many industries aside from printing and photography. The details of the colour spaces are discussed in the following paragraphs.

RGB The RGB colour space generates colours by mixing different amounts of red (R), green (G) and blue (B) light. Each signal ranges from 0 to 255 levels, and it is typically digitized to 8 bits. Thus, this takes 3 bytes for storage of an RGB pixel. Figure 2.4 illustrates the RGB colour cube spanned by the three major axes of the primary colours.

Three primaries consisting of red, green and blue and the secondary colours comprising of cyan, magenta and yellow are vertices of the the RGB 'cube'. The origin (0,0,0) is at the vertex of the cube presenting as black, and the major axes include black–red, black–green and black–blue. Colour is identified by coordinates along the axes, and it can be referred to as (R,G,B). From the RGB cube, coordinates can be specified from 0 to 1 on each axis; meanwhile, 0 to 255 levels of each component are usually preferred in computer technology. For instance, cyan combines maximum proportions of green and blue without red, so it is represented as (0,255,255). White light is full brightness of red, green and blue together; therefore, it is (255,255,255). If amounts of the three components are specified equally, it results in a shade of grey, on the grey line shown in Figure 2.4. In addition, the entire spectrum of colours can be generated by mixing the primary colours in different proportions.

The RGB components can yield various colour features. A basic one is normalized RGB, which is often used, and is sometimes able to remove distortions caused by lights and shadows. Normalizing the RGB values is obtained as:

$$r = \frac{R}{R+G+B} \tag{2.5}$$

$$g = \frac{G}{R+G+B} \tag{2.6}$$

$$b = \frac{B}{R+G+B} \tag{2.7}$$

Gevers and Smeulders [7] proposed colour models consisting of $c_1c_2c_3$ and $l_1l_2l_3$.

The models are invariant to a change in viewing direction and illumination. $l_1l_2l_3$ is also invariant to highlights, where:

$c_1c_2c_3$:

$$c_1 = \arctan\left(\frac{R}{\max(G,B)}\right) \tag{2.8}$$
$$c_2 = \arctan\left(\frac{G}{\max(R,B)}\right) \tag{2.9}$$
$$c_3 = \arctan\left(\frac{B}{\max(R,G)}\right) \tag{2.10}$$

$l_1l_2l_3$:

$$l_1 = \frac{(R-G)^2}{(R-G)^2+(R-B)^2+(G-B)^2} \tag{2.11}$$
$$l_2 = \frac{(R-B)^2}{(R-G)^2+(R-B)^2+(G-B)^2} \tag{2.12}$$
$$l_3 = \frac{(G-B)^2}{(R-G)^2+(R-B)^2+(G-B)^2} \tag{2.13}$$

CMYK As mentioned above, the printing industry commonly uses CMYK (Cyan Magenta Yellow Key). It is represented as a subtractive colour model, that is mixing based on reflective colours. The outline of the CMYK model as shown in Figure 2.5 is presented similarly to the RGB cube. The subtractive colours or secondary colours include cyan, magenta and yellow used to refer mainly for this colour space. K is black, and it can be generated from C, M and Y components; however, a mix of the CMY inks does not give a good black (black ink is also cheaper). White is shown at the origin, and the volume inside the model with mixing the subtractive colours in different proportions creates all colours of the

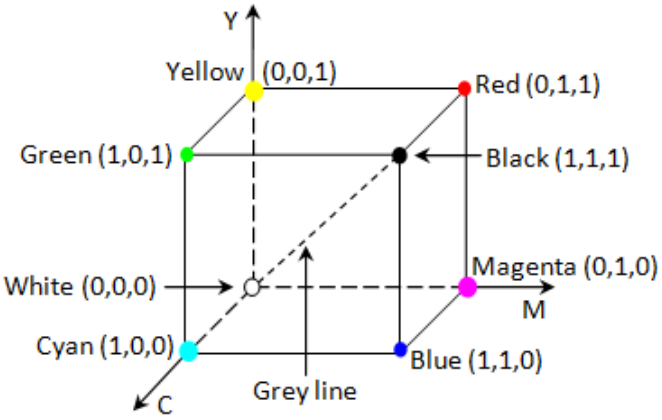


Figure 2.5: Representation of the CMYK cube

spectrum.

HSI Another colour format used to explain an idea of colour is HSI. It comprises of hue (H), saturation (S) and intensity (I). Each component is demonstrated differently from the RGB colour space. Figure 2.6 shows a geometrical representation of the HSI model.

Hue is a dominant factor to present light-reflecting properties of materials. The 'colour circle' expresses hue as an angular position between 0° and 360° that controls the colour spectrum from red through the yellows, greens, blues and violets respectively. For instance, red is at 0°, green at 120°, and blue at 240°. One can say that hue is useful for image processing in that it presents colour in a way that corresponds to physical characteristics of objects and human visual system response.

Saturation represents the purity of colour. Its value is represented by the radius of the point from the origin to the edge of the colour circle. The degree of saturation is measured in a 0–1 scale. It is probably considered as the amount of white mixed with the colour. A colour lying around the edge of the colour circle

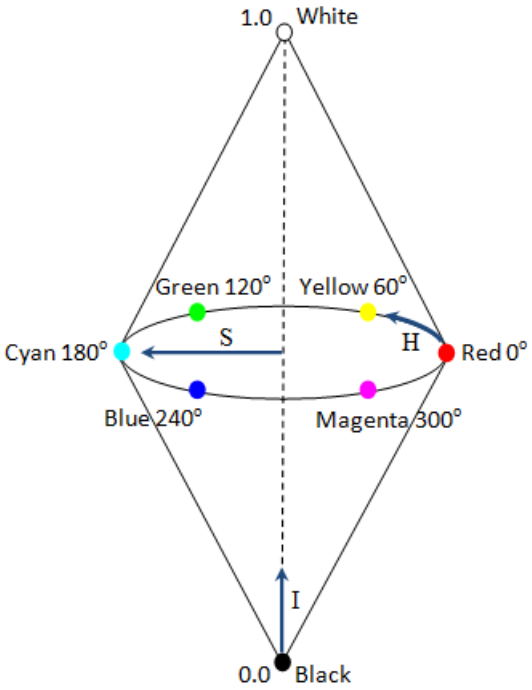


Figure 2.6: Geometrical representation of the HSI model

is 100% saturated, and the purity becomes paler when the distance is closer to the centre until there is no saturation (grey) at the centre.

Intensity represents the overall brightness or dullness. Illumination and viewing angle have an effect on the intensity of an object. The intensity is also presented on a 0–1 scale. The shades of intensity fall along the axis that starts at the bottom point represented as black, and increase upwards to the top, shown as white. These vertical shades produce grey levels without saturation.

Other colour spaces defined similarly to the HSI space include HSL (hue, saturation and lightness), HSB (hue, saturation and brightness) and HSV (hue, saturation and value). HSI is widely employed in vision applications. An advantage of using it is that the H and S components are invariant to changes of light, reflection and viewing direction. Hue also reputedly has less impact from highlights [7].

In terms of defect inspection of agricultural produce, hue and saturation can be employed effectively to analyse colour skin defects — hue presenting the defective colour and saturation presenting the severe degree of defects [8].

CIE L*a*b* The CIE L*a*b* colour space is widely used to provide colour specifications regarding household paint, texture dyes and plastic colour. It is also used in laboratory colour measurement devices. The colour model of CIE L*a*b* is represented in Figure 2.7.

The central vertical axis represents lightness (L*), ranging from 0 (black) to 100 (white). The other horizontal axes crossing each other in the middle of the vertical axis represent a* and b*. The a* axis indicates the green–red axis, and the b* axis represents the colour falling along the blue–yellow axis. In theory, values of a* and b* are unlimited, but in practice they are generally referred in 256 levels from -128 to 127. a* and b* are chromaticity axes, based on the fact that a colour cannot be both green and red, or blue and yellow, because they oppose each other. The centre of each axis is zero, describing neutral grey.

Colour conversion

For image processing, RGB components can be derived directly from images. Furthermore, they are capable of conversion to other colour spaces [9, 10], such as CMYK and HSI as shown in the following.

RGB-to-CMYK conversion As mentioned above, a digital image is displayed using the RGB space. When it is printed, it requires a process to convert RGB

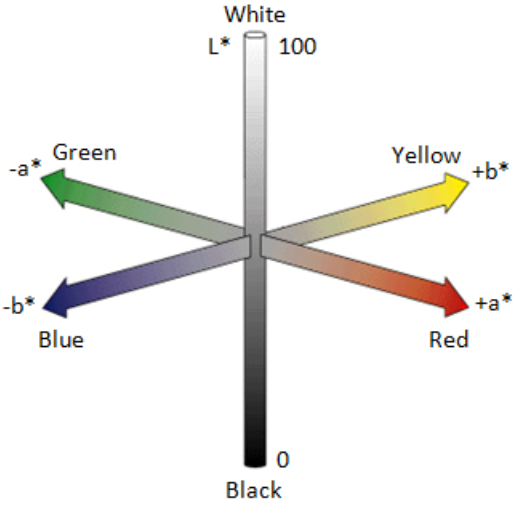


Figure 2.7: Representation of the L*a*b* model

to CMYK. Formulae for RGB to CMYK conversion are

$$R' = \frac{R}{255}, \qquad G' = \frac{G}{255}, \qquad B' = \frac{B}{255}$$

$$K = 1 - \max(R', G', B') \tag{2.14}$$

$$C = \frac{1 - R' - K}{1 - K} \tag{2.15}$$

$$M = \frac{1 - G' - K}{1 - K} \tag{2.16}$$

$$Y = \frac{1 - B' - K}{1 - K} \tag{2.17}$$

CMYK-to-RGB conversion In order to convert CMYK to RGB components, it can be managed by the following formulae:

$$R = 255 \times (1 - C) \times (1 - K) \tag{2.18}$$

$$G = 255 \times (1 - M) \times (1 - K) \tag{2.19}$$

$$B = 255 \times (1 - Y) \times (1 - K) \tag{2.20}$$

RGB-to-HSI conversion As mentioned above, hue is able to express object colour directly. Therefore, not only RGB space is employed in the research, but HSI is also used to represent object features. The following formulae can be used to convert RGB to HSI straightforwardly information lost.

$$I = \frac{R + G + B}{3} \tag{2.21}$$

If I $>$ 0,

$$S = 1 - \frac{3\,min(R, G, B)}{R + G + B} \tag{2.22}$$

If I $=$ 0,

$$S = 0 \tag{2.23}$$

If G $\leqslant$ B,

$$H = \cos^{-1}\left[\frac{\frac{1}{2}[(R - G) + (R - B)]}{\sqrt{(R - G)^2 + (R - B)(G - B)}}\right] \tag{2.24}$$

If B $>$ G,

$$H = 360 - \cos^{-1}\left[\frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right] \tag{2.25}$$

HSI-to-RGB conversion The converting from HSI to RGB is considered separately in three cases. The formulae are used depending on the hue value.
For $0° \leqslant$ H $< 120°$,

$$R = \frac{I}{\sqrt{3}}\left[1+\frac{S\cos(H)}{\cos(60°-H)}\right] \tag{2.26}$$
$$G = \sqrt{3}I - R - B \tag{2.27}$$
$$B = \frac{I}{\sqrt{3}}(1-S) \tag{2.28}$$

while for $120° \leqslant$ H $< 240°$,

$$R = \frac{I}{\sqrt{3}}(1-S) \tag{2.29}$$
$$G = \frac{I}{\sqrt{3}}\left[1+\frac{S\cos(H-120°)}{\cos(180°-H)}\right] \tag{2.30}$$
$$B = \sqrt{3}I - R - G \tag{2.31}$$

and for $240° \leqslant \mathrm{H} < 360°$,

$$R = \sqrt{3}I - G - B \tag{2.32}$$

$$G = \frac{I}{\sqrt{3}}(1 - S) \tag{2.33}$$

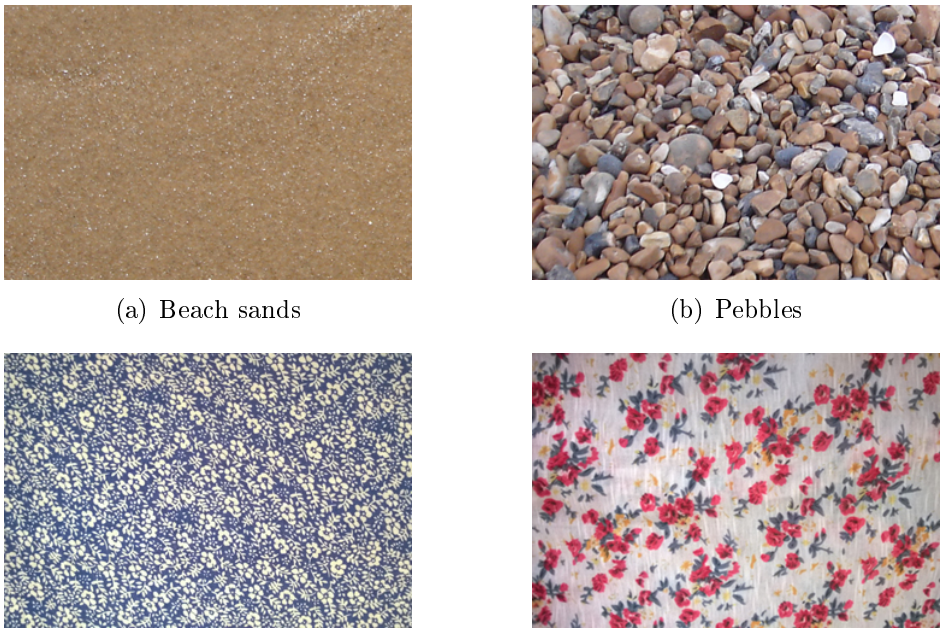
$$B = \frac{I}{\sqrt{3}}\left[1 + \frac{S\cos(H - 240°)}{\cos(300° - H)}\right] \tag{2.34}$$

2.1.4 Texture analysis

Texture is a physical property that represents surface characteristics of objects, and it is the repetition of a basic pattern. These characteristics can be described in many ways by computer vision that would not be recognized by human eyes. Texture is generally described as fineness, smoothness and coarseness, *etc.* Textural characteristics are usually examined in terms of spatial frequency content, such as spatial size and spatial changes of distribution according to distance. For example, fine textures are formed in smaller primitives and higher distribution; in contrast, coarse textures are formed in larger primitives and lower distribution in the same area. Photographs of different textures are presented in Figure 2.8. It can be seen that the textures of (a) and (c) are finer than those of (b) and (d).

Noticeably, a distinct property of a texture is pattern repetition. From Figure 2.8, photographs of (a) and (b) are natural textures whereas (c) and (d) are human-made textures. Most natural textures are not designed, so variation in intensity, size and shape may occur in the repetitions of the basic structure. Human-made textures are generally designed in repetitive patterns; therefore, they are probably more consistent and can be predicted.

Texture description can be analysed by statistical and syntactic approaches [6].



(a) Beach sands (b) Pebbles

(c) Portion of a shirt (d) Portion of a dress

Figure 2.8: Examples of different textures

Statistical methods are suitable to manage textures generated from statistical patterns. The statistical pattern is the repetition of the basic pattern that is irregular, and it is not able to be deterministic. Most natural textures are in this category. Example methods discriminating textures based on an analogy with grey level statistics include spatial frequency analysis, co-occurrence matrices and run lengths.

Spatial frequency evaluation is a general term for a large group of texture analysis methods including approaches that aim to estimate relations of texture pattern distributions. Correlation coefficients are calculated to retrieve texture features. Various principles can be applied to describe spatial frequencies, such as the Fourier transform [11] and the wavelet transform [12]. The co-occurrence method [13] analyses pairs of grey levels with defined distances at particular directions. The run length procedure [14] evaluates the travelling distance of a pixel

value moving in a specific direction until meeting a change of the pixel value. The co-occurrence and run length approaches are described in detail in section 3.4.2. A basic statistic, such as the standard deviation, may describe the brightness variance compared with the mean brightness. Low standard deviation value indicates a smooth texture; in contrast, high output value points out a coarse texture [15].

Syntactic methods are rarely used for general image analysis. They are able to compute textures generated from deterministic patterns or structural textures, such as human-made textures. These textures have identical structures designed in the regular repetition of patterns. The existing methods are capable of dealing with shape chain, graph and hierarchical textures.

2.2 Assessment of Agricultural Produce by Computer Vision

Grading of produce by humans has the disadvantage of subjectivity, inconsistency and inaccuracy, so the agricultural industry desires an automated grading machine to replace the human operation. Consequently, vision machines have been developed to play a key role in this field. Computer vision has been widely employed in terms of inspection, classification and grading of fruit and vegetables [16–18]. Systems can extract and analyse physical properties of produce, such as size, abnormal shape, and defects from a single view or multiple views. In this section, the literature regarding computer vision to manage agricultural produce grading is reviewed.

2.2.1 Maturity and quality grading

Colour is a significant factor in determining the maturity or quality of agricultural produce [19–26]. Lino *et al* [27] analysed the ripening process of tomatoes based on RGB and grey-scale. They found that during ripening, the red value rapidly increased while the others gradually rose to a turning point. After that the red line continued to go up slowly whereas the others decreased.

Gejima *et al* [28] evaluated the maturity of tomatoes based on colour properties. The correlation of RGB and L*a*b* colour components and the maturity was investigated. They found that the G channel had the highest correlation coefficient with the maturity for the RGB model, and a* returned more significant relation to the maturity than the others for L*a*b*. The value of a* increased according to the maturity. Overall, using a* achieved more accuracy than employing the G component. They defined maturity index based on a*, and its average result of maturity identification reached 96% accuracy. From the a* histogram, they found that the reflection of the sample skin had an impact on analysis. Skin reflection occurs in many types of agricultural produce, and it probably causes a main problem for colour analysis.

In addition to colour, shape properties were extracted to inspect the quality of agricultural produce [29–42]. For instance, Fang *et al* [43] recognized physiological diseases of tomatoes. A roundness feature was calculated to identify cavernous tomatoes, and variations of diameter were evaluated to detect deformed tomatoes. The sample was perpendicularly divided along the fruit axis into twelve equal sections, and all partitions were extracted to form input parameters of variations. As roundness was evaluated from perimeter and area values, they compared the accuracy of their estimated measurement with actual values. The results showed

that the correlation coefficient of the perimeter measurement was 0.9990, and 0.9995 for the area measurement. Artificial neural networks optimized by a genetic algorithm were employed for deformed tomato discrimination, and it was able to achieve perfect classification for the unseen testing data.

Colour characteristics also play a key role in inspecting the maturity and quality of apples [44–47]. For example, Xiaobo *et al* [48] presented an apple grading system. A sample was laid on a roller instrument, and it was rotated around the stem-calyx axis. Pictures were taken separated by 90^o, so four images of an object were employed. 17 colour feature parameters (FP) including mean, variance and normalization in RGB space, and eight intervals of hue, were extracted. A method called organization feature parameter (OFP) was proposed. The technique was based on formulae expression trees using a genetic algorithm. The combination of OFP and a step decision tree [48] were employed to classify the samples. The FP was fed as input to a back-propagation artificial neural network (BP-ANN) and support vector machine (SVM). The classification performance using FP was compared with performance by the proposed technique. The experimental results showed that the achievement of the proposed method was superior to operating by BP-ANN but inferior to the SVM.

As the intensity of stem and calyx areas is similar to that of skin defects, it causes misclassification for defective apple discrimination. Consequently, researchers have tried to develop techniques that distinguish stem and calyx regions from defective areas. Shape and texture characteristics were extracted to incorporate with colour properties to solve the difficulty [49–51].

For instance, Unay and Gosselin [52] recognized stem and calyx on Jonagold apples. The images can be divided into four groups; good skin view, stem view, calyx view and defect view, with various sizes and types of defects. Each image

was composed of four filter images. Seven colour, one texture and three shape properties were extracted from each filter image. Five classification approaches were used to verify this task. The classifiers included linear discriminant classifier, k-nearest neighbor, fuzzy k-nearest neighbor, AdaBoost and support vector machine. They found that the support vector machine was able to give the most accuracy. It could achieve perfect classification for calyx recognition and hit 99% accuracy for stem detection.

Many researchers tried to distinguish stem-end and calyx regions from defective areas of apples; in contrast, Xu *et al* [53] developed a vision machine to detect apple defects. The presented technique inspected defective apples without analysis on that issue. They employed three cameras set above the object position to take three pictures while a object was rolling and passing through the trigger. In total, nine images were obtained (three images per camera). They proposed an idea that stem-end and calyx areas could not be seen in the same image. Therefore, if there were two or more doubtful blobs (stem-end, calyx and defective regions) in an object image, the object had a defect.

They did experiments to compare the accuracy performed by using nine images (from three cameras), six images (from two cameras) and three images (from one camera). They found that using nine images they were able to cover the whole surface of an object. 5–10% of the object surface would be lost using six images and 15–20% lost using three images. The performance of the system achieved 94.5%, 83.7% and 63.3% from employing nine, six and three images respectively. The system accuracy obviously decreased when the system processes fewer images. This task is apparently able to indicate that using more object viewpoints is capable of representing the appearance of an object better, and it probably results in more accuracy for inspection.

Colour, texture and shape properties were used for grading in various types of produce [54, 55]. Kondo *et al* [56] examined relationships between physical properties and the quality of Iyokan oranges based on sugar content. Features extracted from a vision technique consisted of R–G colour component ratio, texture calculated on the green band and height–width radio. Sugar content was measured using a laboratory instrument. They found that the extracted features had low correlation with the sugar content. The measured properties were also employed to predict sugar content by neural networks. The experimental results showed that the correlation coefficient between measured and predicted sugar content reached 0.84. Although the extracted physical properties of the samples were not directly related to the sugar content, the combination of them could be used to predict the sugar content using a classification system.

The RGB colour space is the basis of colour for image processing. It represents an image in three separated components (red, green, blue) whereas hue in the HSI colour model can represent chromaticity by itself. Xu and Zhao [57] proved this issue with a type of agricultural produce. They analysed the hue of strawberries with different maturity levels in different light intensities. The experiments were demonstrated in two parts. The first part was to compare the hue of samples in the same maturity group under four light intensities. The experimental results showed that the hue histograms of the sample groups differed minimally, indicating that the variation in the brightness of light does not significantly affect hue values.

The second experiment was to evaluate the hues of samples in three maturity groups under the same light intensity. Each light intensity was employed for hue measurement on all maturity groups. They found that hue histograms of the different maturity categories were formed with two regions, but they apparently differed from each other for all light conditions. That means strawberries with

different maturity levels exhibit different hue distributions. Important hue values were chosen to generate input features to be employed for maturity classification by using neural networks. The network structure was optimized by a genetic algorithm. The average result reached 91.7% accuracy. The HSI colour space is widely applied to derive colour and texture features for analysis in various types of agricultural produce [23, 25, 44, 47, 54, 58–60]. Its effectiveness has been proved in theory and practice. Accordingly, both the HSI model and a basic colour space (RGB) are utilised in this thesis.

Chalidabhongse *et al* [61] developed a vision system to measure 2D and 3D shape properties of mangoes and used them for sorting. 2D features consisted of length, width, thickness and projected area. Volume and surface area were extracted from 3D voxels reconstructed from four images of an object. The accuracy of the vision-based measurement was compared with measuring by laboratory instruments. The former yielded slightly more variation than the latter, but the former was more efficient in terms of speed and convenience. The extracted features were employed for size classification using backpropagation neural networks, and the system performance was compared with decisions by experienced farmers. The system achieved 96.47% accuracy, while the farmers managed 87.66% average accuracy. This experiment indicates that a vision system is able to give more accuracy than humans. The 3D reconstruction techniques of this research were applied to the system described in this thesis.

2.2.2 Cultivar classification

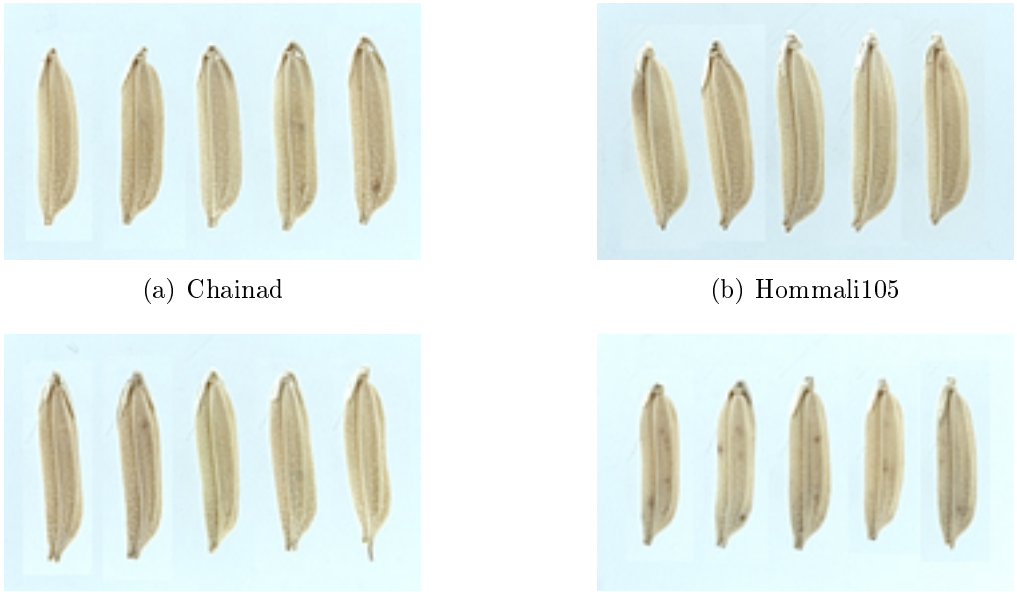
Most fruit and vegetables are available in more than one cultivar. Each variety generally has its own properties and qualities. Some types can be easy to identify,

(a) Golden delicious (b) Granny smith (c) pink lady

Figure 2.9: Different types of apples

but some types are difficult to discriminate from each other. For example, apples in cultivars golden delicious, granny smith and pink lady are shown in Figure 2.9. It can be seen that it is easy to discriminate the pink lady fruit from the others because of colour. The golden delicious fruit can be distinguished from granny smith by size because golden delicious apples are usually bigger than granny smith. In contrast, the physical properties of some varieties of rice are moderately similar to each other. Four types of rice are sampled in Figure 2.10. Therefore, rice variety identification is a challenging issue for computer vision. Various vision systems have been developed to deal with cultivar classification on agricultural produce [11,62,63].

Techniques have been proposed to classify types of grains [64–74]. For example, Paliwal *et al* [65] identified five types of cereal grains including barley, oats, rye and two varieties of wheat. They also classified them in five categories of dockage constituents: broken wheat kernels, chaff, buckwheat, wheat spikelets (one to three wheat kernels inside husk) and canola (rapeseed with low erucic acid content in the oil and low glucosinolate content in the meal). Experiments were done to compare the performance of employing different feature models. 51 shape, 123 colour and 56 texture features were extracted. The shape characteristics consisted of area, perimeter, major axis length, minor axis length, maximum



(a) Chainad (b) Hommali105

(c) Patumthani (d) Supanburi

Figure 2.10: Different types of rice

radius, minimum radius, mean radius, invariant shape moments and Fourier descriptors. The colour properties contained mean, variance, ranges of RGB and HSI components and RGB histogram features. Texture features were calculated from the co-occurrence matrix and grey level run-length matrix in grey and RGB bands. Various functions can be applied to extract texture features based on the co-occurrence matrix and grey level run-length matrix. The texture extraction functions used in [75] are adopted for this research. Neural networks were employed for classification. The extracted features were ranked, and the best 60 and best 30 features were grouped in different models performing as input sets: a combination of top-60 features of all features (combined-60 model) and top-30 features of all features (combined-30 model), a combination of top-20 features in each feature group (top-60 model) and top-10 features in each feature group (top-30 model).

The experimental results showed that employing the combined-60 model was superior to using the top-60 model; similarly, the performance of the combined-30 model out-performed operating by the top-30 model. The achievement comparisons of using all features, the combined-60 and combined-30 models were not significantly different; however, employing more features took a longer time for training. As a huge number of features were employed in this research, better features should be selected to generate a classifier. Therefore, the idea of the combined feature model is adopted for experiments in this thesis.

In the same year, Paliwal *et al* [66] compared the classification performance of a backpropagation neural network and a non-parametric statistical classifier, experiments being done with the same types of cereal grains and the same feature operators. They found that the neural network achieved better results than the other classifier. Mehrez *et al* [73] classified hard wheat, tender wheat and barley using a statistical pattern recognition method and a fuzzy logic approach. Morphological features and RGB colour properties were extracted for input parameters. They found that the statistical pattern recognition method was able to achieve the highest accuracy for barley identification but performed poorly for classifying hard wheat and tender wheat. A combination of length and angle parameters was applied as input features for fuzzy logic. Its achievement was dramatically better for wheat discrimination, but it did not perform well for barley recognition. Therefore, they proposed a combination of the two methods (hybrid method) that could sharply improve the accuracy for all sample groups.

Douik and Abdellaoui [71] continued this thread of research. More input features were added, such as wavelet features and further morphological features extracted from Freeman code, and Fourier transform methods. Artificial neural networks (ANN) were employed for classification, and their performance was

compared with the previous proposed classifiers. For ANN, input feature groups including morphological, colour and wavelet were employed for classification separately. The experimental results showed that an ANN claimed 100% accuracy for tender wheat recognition using the wavelet features and yielded the most accuracy for barley identification using the morphological features. The hybrid method, using length and angle properties, could return the highest accuracy to classify hard wheat.

A neural network approach is most widely used for classification problems and its performance is effective and appropriate for various tasks. Not only neural networks are efficiently used in the field of agricultural produce [35, 36, 48, 62, 76–78], other classifiers, such as support vector machine [48, 52, 79] and k-nearest neighbour [52, 62, 77] are also reliable and often applied. They are examined for classification tasks in this thesis and their achievement is compared with the genetic programming classifiers generated by this research as shown in chapter 4.

Rice is a major food for Asian countries. Milling of rough rice is a process to remove rice husks. A whitening process then removes the brownish outer bran layer, and finally polishing is performed. The degree of milling depends on the removal of bran layer that relates to the whiteness. The milling processes cause an increase in the number of broken seeds. The quality of milled rice is associated with physical appearance, such as shape, whiteness and cleanliness [80]. Different cultivars have different qualities, but some types are similar to each other in appearance: people who are not involved in the agricultural field may not be able to identify them. Certainly, prices are related to the quality. Traditionally, rice seeds are randomly selected to be examined manually before they are transferred to the market. Nowadays, various systems are being developed to classify varieties and qualities of rice [80, 81].

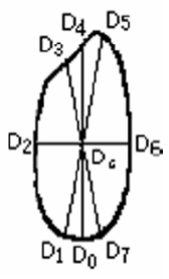


Figure 2.11: Detected points used to extract shape features [83]

Guzman and Peralta [82] classified properties of 52 cultivars of rice collected from different areas. Only 13 morphological features were extracted and used as an input set for artificial neural networks. Some types of the samples had roughly similar physical properties, so they were categorized in terms of size and shape. Neural networks were built with different structures in each experiment. The classifiers achieved 98.76% average accuracy for size identification and 96.67% for shape discrimination.

Gao *et al* [83] proposed an idea to extract the shape properties of rice. They claimed that Fourier descriptors were often used for shape extraction, but a huge number of calculations were performed. Therefore, a simple technique was presented: a centroid point was calculated, and eight points on the seed boundary were selected according to the defined angles. The detected centroid and eight points of a rice seed are shown in Figure 2.11. They measured the lengths D_0–D_7 to generate shape parameters. The performance of the proposed shape extraction was not proved. The measurement is simple and able to extract rice features effectively; however, it is probably not flexible enough to apply to a wide range of agricultural produce.

Not only morphological features have been employed for rice variety classification; RGB and HSI components have also been employed. For example, [59,68,84]

extracted morphological and colour properties and fed them as input to neural networks. The classification results were able to reach over 80% accuracy.

Paulus and Schrevens [85] described the shape characteristics of apples based on Fourier expansion. Shape was able to be identified using two principal components: the first one was the ratio of height to width. It was divided into five intervals; lower values indicated more elongated shapes. The other component measured the conicity, and it has three intervals. Lower values represented more triangular shapes whereas higher values described more rectangular. Similarly, Currie *et al* [86] characterized apple shape. Each sample was cut along the stem-calyx axis, and the outline was extracted to gain Fourier descriptors. They found that respectively aspect ratio, conicity and squareness properties effectively performed for this task.

Shape representation approaches can be divided into two groups, region-based and contour-based [87] representations. For the region based method, shape descriptors are derived from operations on pixels in an object's shape. This technique operates on the entire object shape rather than the object boundary. Object shape can be represented by octree, skeleton or morphology decomposition approaches. Examples of region-based shape descriptors are area, height, width, eccentricity, centroid, rectangularity, elongatedness, compactness, *etc.*

For the contour-based principle, shape boundary information is calculated. Fourier descriptors and wavelet descriptors are examples in this group. Fourier transformation on shape signatures is frequently applied for shape analysis. Shape signatures are any functions describing boundaries or areas, and they are performed by Fourier series that can generate coefficients forming Fourier descriptors of shape. These descriptors can represent the shape in terms of either a lower or higher frequency domain. Higher frequency descriptors contain finer informa-

tion of the shape than lower frequency descriptors [87]. Wavelets are functions employed to represent data or other functions. Wavelet transforms are used to operate the decomposition of data in different frequency components. Wavelet representation of a sequence of points forming a shape boundary delivers wavelet descriptors [88]. Both descriptors are normalized to be invariant under scale, translation and rotation.

Osowski and Nghia [88] recognized object shape by using wavelet and Fourier descriptors. They found that the wavelet functions had better abilities for shape localization in both time and frequency domains. Zhang and Lu [87] claimed that using wavelet descriptors needed more intensive computation, and they were more appropriate for model-based object recognition than shape information retrieval.

This research takes computation cost for object extraction into account, so simple but effective shape properties will be used. The shape extraction operations perform based on object region and as discussed in section 2.1.2. They are practical and suitable for general shape analysis.

2.3 Genetic Programming Fundamentals

Genetic programming (GP) is a part of machine learning. Computer programs are generally coded to respond to every feasible request from users. Computer programs command computers, and genetic programing produces computer programs. GP aims to generate computer programs able to solve the problem without having to specify exactly how to do it [89]. Operators are formed randomly into a tree called 'computer program' or 'individual'. An initial population of programs is produced, and a cost function is defined to measure how well a computer program solves the problem. The better-performing individuals continue to the next

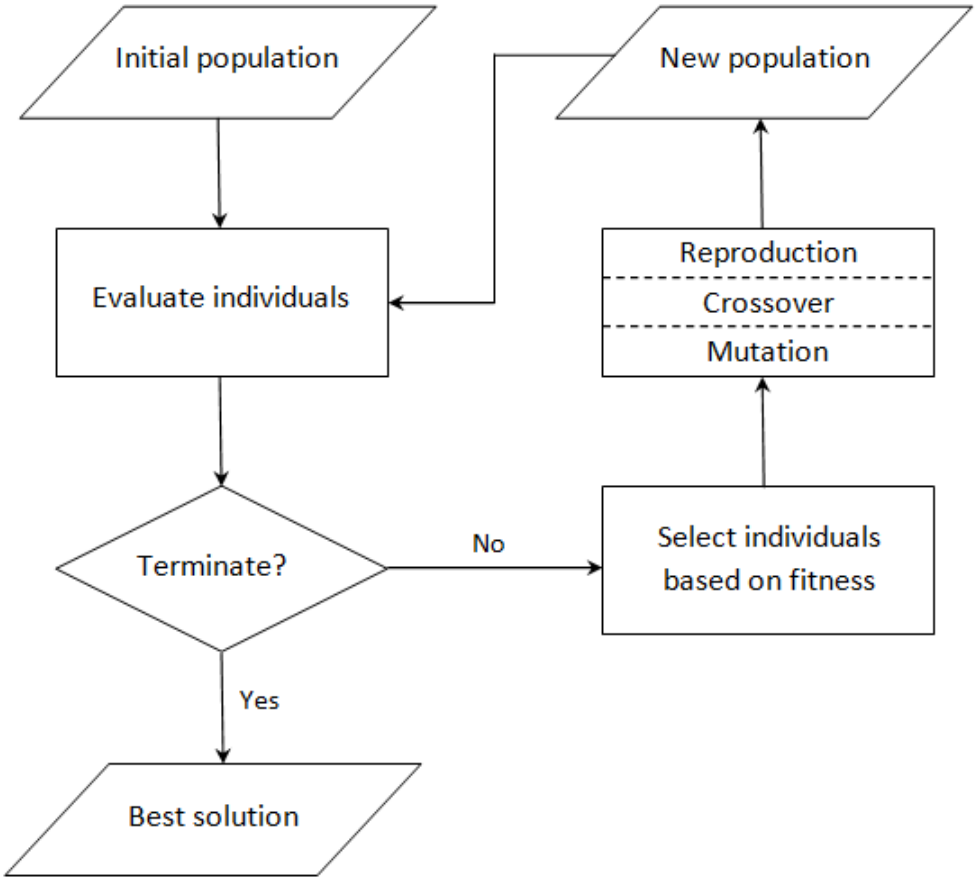


Figure 2.12: Overview of genetic programming processes

generation or take part in a creation of a new population. These individuals are then evaluated using the same cost function, and so on. The GP evolution runs iteratively until it gains a solution with good fitness, or a limit on the number of generations is reached.

Genetic programming generates a solution automatically without requiring the user to instruct how to do it or identify the solution structure in advance. It does not guarantee that an obtained solution is the global optimum, but it is certainly the best solution of the evolution locally. Furthermore, many GP runs on the same

training dataset may return different solutions. Genetic programming processes are illustrated in Figure 2.12, and genetic programming background is described in the following paragraphs.

2.3.1 Population

Genetic programming is a descendant or an extension of a machine learning technique — Genetic Algorithms (GA). General GP theories are derived from GA theories; however, a main difference between them is the structure of individuals. GP individuals are computer programs expressed as syntax trees with arbitrary size, while any kind of fixed-length vectors are formed for genetic algorithms [90].

The first step of a GP run is to initialize a population. A GP population contains computer programs consisting of root, internal and leaf nodes. Normally, functions are randomly selected for root and internal nodes whereas terminals (explained in the next section) are randomly assigned to leaves. A maximum size is defined to control the size of the tree regarding depth or numbers of nodes. In terms of the depth, the root node of a tree is presumed to be at depth 0, and the depth of the deepest leaf is the depth of the tree. From a sample individual shown in Figure 2.13, the depth of the tree is three.

There are two basic methods used to construct trees that result in different tree shape. The first approach is a full method. It randomly chooses nodes from a function set to build a tree until the tree reaches the maximum depth, then it randomly selects a node from a terminal set instead. Consequently, all terminals are at the same depth in every branch, and each branch reaches the maximum depth.

The other basic tree builder is named grow; in contrast to the full method, it

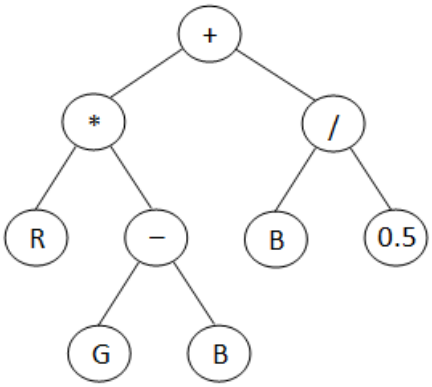
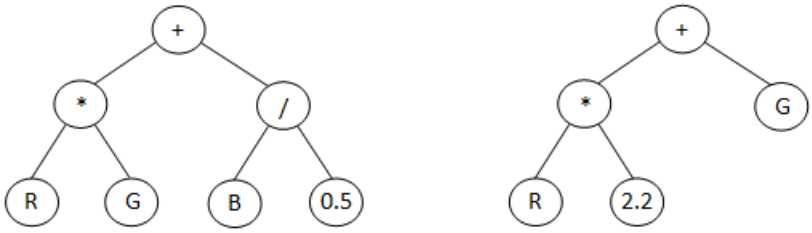


Figure 2.13: Sample tree

creates trees of unbalanced shape because not all branches reach the maximum depth. Every node is randomly selected from function and terminal sets (except the root node, which is a function node). When a branch obtains a terminal node, the branch terminates building before reaching the maximum depth. Sample full and grow trees are presented in Figure 2.14.

Furthermore, Koza [91] proposed *ramped half-and-half* to combine the full and grow methods so as to create computer programs in a wide range of size and shape. In other words, the approach supports half building by full and half building by grow for an initial population that is used in the research.

Only individuals in the initial population are formed ordinarily by functions and terminals selected randomly. For a new generation, a new population is generated based on the previous population using genetic operations: elitism, crossover and mutation. For elitism, some better individuals are directly put through to the next generation. Some individual programs are probabilistically selected to generate offspring by crossover and mutation operators (the genetic operations will be described in detail later). Creating a new population is repeated for subsequent generations until the run reaches a termination criterion. Note that the size



(a) Full tree (b) Grow tree

Figure 2.14: Sample trees generated by full and grow builders

of the population remains the same in each generation. As highly fit computer programs are more likely to be selected in order to produce offspring than inferior programs, so the inferior programs will be removed from the population after a few generations. Consequently, the average fitness of a population rises over the generations.

2.3.2 Terminal set

As mentioned in section 2.3.1, terminal nodes are leaves of a tree because they lies at the end of every branch of a tree. The terminal set may comprise of variables, constants and zero-argument functions, and the details of each are described as follows.

Variables They are external inputs retrieved from input data features. For example, for the program in Figure 2.13, R, G and B are variables whose values result from colour extraction in red, green and blue respectively.

Constants They are a set of real numbers. A constant is randomly chosen from a pre-specified range of real numbers to put in a computer program. For instance, from the tree in Figure 2.14, random constants put in the trees are 0.5 and 2.2.

Table 2.1: Sample functions usually used in genetic programming

Function type	Samples
Arithmetic	+, -, *, %
Mathematical	sqrt, max, log, abs
Boolean	AND, OR, NOT, XOR
Condition	IF-THEN-ELSE, SWITCH
Loop	FOR, WHILE, DO-WHILE

Zero-argument functions They are functions that take no argument but return a value, and they are also known as parameterless functions. A well-known function of this type is the random number function, rand(). Although `rand()` may return different values each time, Jasmine has a mechanism to set the seed before a run to make the experiment repeatable.

2.3.3 Function set

The function set can consist of all sorts of functions used in computer programs, such as arithmetic, mathematical and boolean functions. Commonly-used functions are shown in Table 2.1. Functions and terminals are referred to as nodes. They together create possible computer programs for evolution. They should be provided reasonably and sufficiently to form a solution to the given problem. For instance, a GP engine with a terminal set consisting of only a constant and a function set comprising of only the subtraction operator will probably not generate a solution to most problems. The sequence of functions and terminals assigned as a solution of a run is not predefined by the user, but it results from the evolutionary process during the run.

2.3.4 Fitness function

A fitness function is employed to evaluate the quality of individuals, where 'quality' means how well an individual has learnt to predict outputs from inputs. The fitness of computer programs has an impact on their survival. Highly fit computer programs have more opportunity to take part in producing a population in the next generation, whilst computer programs with lower fitness have a high chance to be removed from the population.

The fitness function used depends on the problem. For instance, some fitness functions are determined to calculate the accuracy of computer program outputs; in contrast, some functions evaluate the amount of error between the returned outputs and the desired outputs instead. The fitness function employed throughout this research is presented in (2.35); it is used to measure the error of computer programs. If a fitness value of a computer program is estimated and its error is equal to zero, the computer program is effective enough to be returned as a solution for the evolution.

$$error = \frac{FP + FN}{N} \tag{2.35}$$

where FP is the number of false positives (explained in section 3.5.4), FN is the number of false negatives (explained in section 3.5.4), and N is the number of training samples.

2.3.5 Genetic programming parameters

GP parameters play a key role in controlling evolution. Control parameters include population size, the maximum size of programs, the probabilities of performing the genetic operations and other details of the run. The control parameters used

Table 2.2: Sample genetic programming parameters set in the program

Parameter	Setting
Population size	500
Numbers of generations	50
Maximum size of tree	100
Minimum depth of tree	1
Maximum depth of tree	6
Crossover rate	70%
Mutation rate	20%
Elite count	5

in the GP engine of this research are presented in Table 2.2.

2.3.6 Termination criteria

A run of a GP engine commonly terminates by either obtaining a satisfactory solution or reaching a defined maximum number of generations. In other words, a GP engine runs iteratively until it gains the best solution. However, in the worst case if no program is fit enough, a maximum number of generations is employed to terminate the evolution. As shown in Table 2.2, 50 is assigned to be a maximum number of generations throughout this thesis.

2.3.7 Selection

In order to generate computer programs for a new generation by using GP operators, individuals of the previous population are mostly involved to be chosen to take part in producing a new population. Individual selection is based on fitness because better individuals are probably more suitable for breeding than inferior

individuals. Tournament selection [92] is the most widely used technique for choosing individuals. The quantity of individuals defined as tournament size (t) are randomly selected from the previous population. Their fitnesses are compared, and the one of the highest fitness is selected as the winner of the tournament and assigned to create a new offspring. For the thesis system, the tournament size is set as $t = 2$. Individuals are allowed to be selected for breeding more than once.

Tournament selection is simple to implement, and it can process in parallel [93]. It also does not need a centralized fitness comparison among all individuals in every generation [91]. However, it is probabilistic [89]. The best individual in the population is not selected all the time, and the worst individual is not always ignored from the selection process.

2.3.8 Genetic operations

A population of programs is iteratively produced for every new generation of the evolution. Three strategies, crossover, mutation and reproduction, are employed to generate individuals from the previous population. Note that these operations proceed based on copies without disrupting the original individuals. Thus, they are able to be selected to create offspring multiple times by selection.

Crossover

The main operator of genetic programming is crossover. Sub-tree crossover is generally used the most, and one-point crossover is employed in the thesis. Two parents are probabilistically chosen from the population on the basis of their fitness (this research uses the tournament selection), and a crossover point of them is randomly selected. Swapping the parents' sub-trees creates an offspring

program (for one-offspring crossover) or two offspring programs (for two-offspring crossover). The two-offspring version of crossover is used throughout this research. Tree-based crossover is shown in Figure 2.15. Each offspring certainly contains some characteristics from each of its parents. The crossover probability rate is normally declared with a highest probability than the other genetic operators.

Mutation

With crossover, offspring are inherited from their parents, chosen from the previous population. In mutation, a child is generated from a parent altered with a new sub-tree. Sub-tree mutation is the most commonly employed form for mutation. One parent is selected from the population based on fitness (this research uses the tournament selection), and a mutation point is randomly chosen. Following that, a child is produced by replacing a parent's sub-tree with a randomly created sub-tree. An instance of the mutation operation is described graphically in Figure 2.16. Mutation is applied to most GP systems with a much lower probability than crossover.

Reproduction

The reproduction operator is straightforward, and this strategy is also known as elitism. The best n individuals are copied to put through the new population of the next generation without alteration. This is able to reserve highly fit programs for the next generation. The probability of reproduction is commonly defined by a smaller number compared with the other genetic operators.

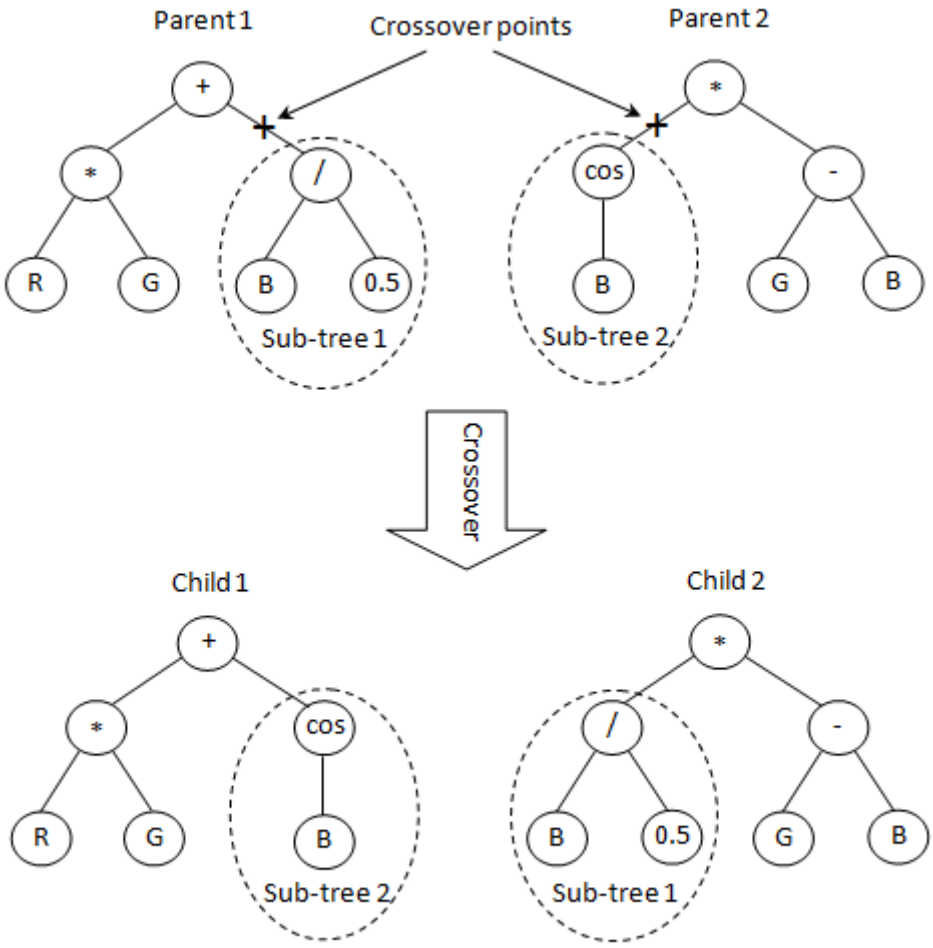


Figure 2.15: Example of sub-tree crossover

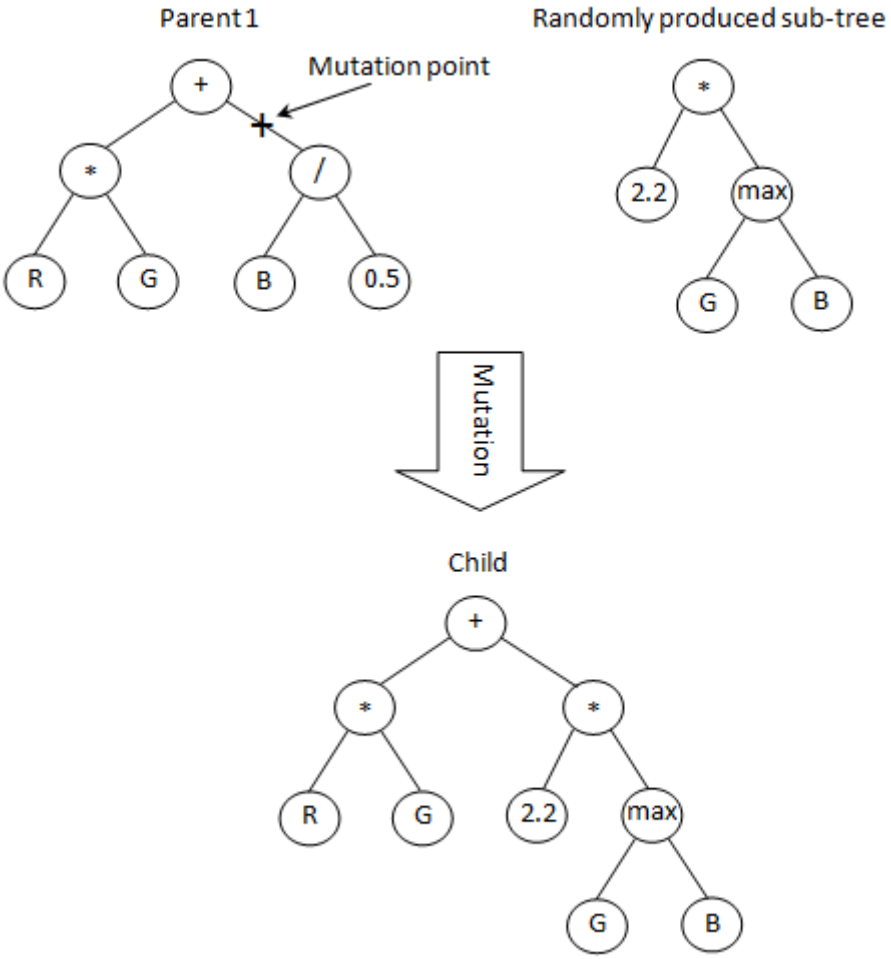


Figure 2.16: Example of sub-tree mutation

2.3.9 Bloat

Typically, GP evolution starts from the initial population of randomly generated programs that are usually small and not fit enough to solve the problem. It is necessary for them to grow in order to contain effective elements to be able to reach the fitness. GP trees can grow considerably without size limitation, unlike GA individuals with a fixed length. When genetic operators and selection perform together to produce better solutions, code growth normally happens. As small programs are probably unfit, they are ignored by selection. Larger programs with better fitness are likely to be chosen as parents; consequently, the mean program size increases.

When program growth becomes excessive without improving fitness, this situation is termed *bloat*. Part of a tree may be a redundant code segment (sometimes described as introns, by analogy with DNA) that does not contribute to the fitness or change the return value. As the increase in program size is not always accompanied by a corresponding improvement in fitness, various bloat control methods have been proposed [94]. For instance, lexicographic parsimony pressure adapts tournament selection so as to select the smaller contestant in the case of individuals having the same fitness. This technique and tree depth checking are example methods employed in Jasmine in order to prevent code bloat.

2.4 Computer vision using genetic programming

An advantage of GP is flexibility, hence GP has been applied to a wide range of tasks [95]. In the vision domain, GP has been applied to tasks, such as [96] (colour, hand gestures and benchmark classification), [97–101] (medical diagnosis), [102–104] (edge detection), [105,106] (texture analysis) and [5,107,108] (character

recognition). GP has also been used for feature selection and construction [97, 109–114]. Some researchers adapted conventional GP procedures to obtain better solutions or to perform more efficiently [115–120]. Many researchers incorporated other approaches with GP in order to improve results [99, 110, 111, 121–123].

Most vision applications involve a classification process, and GP may be chosen to perform this. The first use of genetic programming on a recognizably computer vision problem was to recognize military targets in infra-red images [124]. A GP-evolved solution was employed as the classification stage in an existing multi-stage automatic recognition system, taking as input some twenty features and emitting a target/non-target classification, using arithmetic combinations of these features and a conditional test. A 2,000-example training set was used, and performance was measured on a 7,000-example test set. It was found that the GP solution outperformed a multi-layer perceptron, higher true positive and lower false positive rates being achieved with less computation. A GP solution was also evolved to work with simpler features, just the means and standard deviations of rectangular image regions, and the latter was found to be more effective than the 20-feature solution. Many subsequent attempts to use genetic programming to construct vision systems have followed this same route, providing both image processing and classification operators to the GP 'engine' and running it until a solution is found.

There has been research into using genetic programming specifically for classification. Song *et al* [125] performed texture classification experiments, comparing identification accuracy between static range selection (SRS) and dynamic range selection (DRS). The experiments can be divided into two groups, for binary and multi-class classification problems. They expected that SRS was more suitable for binary datasets, but they found that all accuracy results generated by SRS were

poorer than those of DRS for binary classification, and the classification ability of DRS exceeded that of SRS for multi-class datasets. The DRS classifiers could produce fit solutions at a faster rate than the SRS classifiers. For the experiments involving DRS, some textures needed a multi-threshold (more than one segment for class labels) that was difficult to manage by SRS.

Loveard and Ciesielski [126] presented five classification approaches based on genetic programming: binary decomposition, SRS, DRS, class enumeration and evidence accumulation. They performed experiments with datasets from the UCI Machine Learning repository [127], comparing classifiers' ability in terms of accuracy and training time on binary and multi-class problems. The experiment showed that the performance of DRS was superior to operating by the others for binary classification. The binary decomposition approach achieved higher accuracy than the others for multi-class datasets. Its achievement was slightly greater than performing by DRS, but the standard deviation of errors produced by DRS was more stable than that generated by the binary decomposition classifiers.

Class enumeration took less time for training in all experiments, but when more time was allowed, class enumeration could not improve better than DRS. This task indicated that DRS effectively performed both binary and multi-class classification problems, and the binary decomposition method also operated well for multi-class datasets. Details of the SRS, DRS and binary decomposition approaches are presented in section 3.1, and the DRS and binary decomposition were applied to Jasmine [5], which is the GP framework that formed the starting point for this research.

Smart and Zhang [128] proposed two dynamic range selection methods: centred dynamic range selection (CDRS) and slotted dynamic range selection (SDRS). For CDRS, class boundaries were randomly initialized. After that, genetic pro-

grams were generated and evaluated the fitness. A formula was proposed to calculate the centre of each class. New class boundaries were redefined by using the middle point of every two class centres. Finally, the training data were classified based on the new class ranges and the fitness estimated again.

The SDRS technique performed similarly to the basic DRS as described in section 3.1, but they used 100 slots in the range of $[-25, 25]$ with a step of 0.5. The classification results generated by the proposed techniques were compared with those produced by SRS. The experiments were performed on multi-class datasets comprising of circular shape, square shape and coin images with noisy backgrounds. The experimental results indicated that SRS was suitable for a small number of object classes whereas CDRS and SDRS performed more effectively when the samples were more complex, and the number of classes was bigger.

A year later, Zhang and Smart [129] modified CDRS and SDRS and changed their names to be centred dynamic class boundary determination (CDCBD) and slotted dynamic class boundary determination (SDCBD) respectively. CDCBD was adapted in the size of class range, and the number of slots were extended for SDCBD. They found that the achievement of SDCBD out-performed operating by CDCBD for the difficult datasets.

Attempts have been made to improve conventional genetic programming to produce novel and more effective classifiers. For example, Muni *et al* [130] proposed techniques to improve conventional GP. For instance, they modified the mutation operation. Originally, a mutation point was randomly selected; if the point belonged to a function node, it would be replaced by a new random function with the same number of arguments. If it was a terminal node, a new terminal was randomly chosen to replace it. The presented technique accepted new mutated programs if they were better than the original ones. After mutation, a mutated

program was evaluated with half the training data. If its resulting fitness was equal to the original program, the remaining training samples were employed. If its fitness was better than the original, it would be kept; otherwise, it was discarded. Although this technique probably results in a new population containing better fitness on average, it needs more computation time compared with the conventional operation.

Zhang and Nandi [131] proposed a bundled-GP scheme, which was developed from an independent-GPs scheme. A concept of the independent-GPs was similar to the binary decomposition approach. A binary GP was generated for each class from all other classes; following that, all obtained binary GPs were combined to generate a final solution to give a decision output. Effective features related to each class were selected to produce a GP of the class. As the final solution contained all features from the binary GPs, so this technique was adapted to the bundled-GPs scheme in order to reduce the number of feature nodes in the final multi-class solution. Since binary GPs knew selected features of each other, they attempted to choose the same features. Then, a binary GP of each class were bundled together to generate individuals for a population. This technique resulted in fewer features formed for a final solution. From the results, the independent-GPs could achieve the perfect accuracy. Although the bundled-GPs technique gave slightly poorer classification accuracy and took a longer time than the independent-GPs, the solutions generated from the former contained fewer features than those produced by the latter.

Recently, Al-Madi and Ludwig [121] proposed GP-K and GP-D to adapt the original GP classification process so as to generate better accuracy. GP-K employed k-means clustering to transform GP outputs to class labels. GP-D applied a supervised discretization approach to identify class boundaries based on calcu-

lating the information gain. The GP-D technique is similar to the basic DRS in terms of defining class boundaries in a linear representation. The experiments were done on binary and multi-class datasets, and the performances of GP-K and GP-D were compared with performing by conventional GP and other classification methods, such as k-nearest neighbours, neural networks and support vector machines. The results of GP-D were superior to the other classifiers for three out of four binary datasets. For multi-class datasets, the accuracy of GP-D was higher than operating by GP and GP-K but moderately poorer than the other classification approaches.

CHAPTER 3

A GP FRAMEWORK FOR VISION

The starting point for this research is Jasmine [5], a system that uses GP to evolve complete computer vision systems from components. As will be demonstrated in this chapter, Jasmine described in [5] is not able to solve many problems concerning agricultural produce well, largely because it lacks operators able to classify on the basis of colour and texture.

This chapter describes the principles underlying the original Jasmine and identifies its major shortcomings. Enhancements that overcome these shortcomings are then described and a series of experimental examples are presented that provide evidence that the enhancements improve Jasmine's ability to classify agricultural produce.

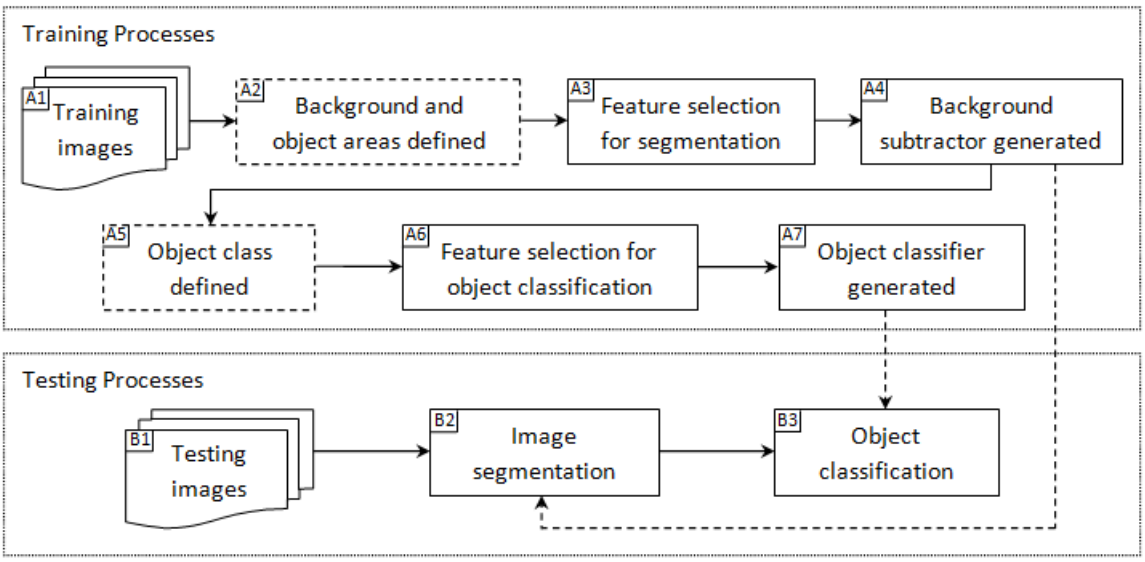


Figure 3.1: Processes of the Jasmine framework

3.1 Jasmine, a learning vision system

Jasmine (Java, A Segmented IMage Notation Environment) uses GP to build vision systems from components. Those evolved vision systems typically have 3 steps: segmentation of interesting features from the background, extraction of those features, and classification of them. Jasmine provides a graphical user interface that allows the user to capture training data for these stages, and to evolve the stages independently. Having trained a system, Jasmine then allows it to be applied to individual test (i.e., unseen in training) images. Figure 3.1 illustrates the separation of training and testing and the stages of the evolved vision systems. Each stage is identified by a letter and number (e.g., A4 is the background subtractor generated by GP), which are used in the descriptions later in this chapter. Those stages with dashed borders mainly involve interaction with the user.

Figure 3.2: Background and object pixels marking

A1, a training image dataset is put into the system. The program is able to support image and video files. For this research, only images are employed as an input set.

A2, each image needs to have both background and object (or objects) of interest identified. This is a manual process. If objects and background of input images are consistent, and there is a large number of images, this does not need to be done on all images in order to reduce the time cost. Figure 3.2 shows sample marking of background and object areas of a melanoma image. Black marked regions are presented as selected background areas, and a white marked region is for chosen object areas. Noticeably, not the whole regions of foreground and background have to be marked.

A3, feature extraction functions provided in the system are evaluated and selected to generate a segmenter. As we do not know in advance that which features are effective, many extraction functions can be built in the system. However, the effect of a huge number of operations is increasing the computing time requirements, and the accuracy does not necessarily increase.

Since various extraction functions on RGB, HSI, binary modes and texture are provided, feature operator selection is used to reduce the number of unnecessary operators. All features are evaluated for their effectiveness for background subtraction, then the better extraction functions are employed to produce a segmenter. Three feature selection approaches: linear discriminant analysis (LDA),

information gain (IG) and program classification map (PCM) can be a choice for the function evaluation.

Step A4, is to generate a background-subtractor, and this task is based on genetic programming. Four GP representations are offered to produce a segmenter. They include two dynamic range selection methods comprising of original dynamic range selection (DRS) and improved dynamic range selection (DRS2C) developed by Oechsle [5] and two threshold theories consisting of entropy thresholding and variance thresholding.

As a computer program returns a numeric value as program output, it will be interpreted as a class label. Static range selection (SRS) and DRS are GP classification strategies to transform program output into a class label, and they performed differently. Class boundaries of SRS are predefined before evolution begins, and they do not change during evolution. For binary classification problems, the zero point is set as the class boundary. Negative program output is classified as class 1; in contrast, class 2 is labeled to positive output.

DRS is an alternative method to SRS. Class segments of DRS can be represented by slots that are dynamically determined during the evolutionary process; therefore, the class boundaries can be different in each run. In practice, the range of [-250,250] is used for the segmentation. Numeric output values are rounded to the nearest integer. Output values lower than -250 are considered as -250, and values higher than 250 are considered as 250. A class is probably labeled in multiple segments. A slot may contain multiple class labels, and this can be dealt with a proposed technique in DRS2C [5], which will be introduced later. A slot with no class assigned is defined by a class of the nearest neighbor slot [126]. DRS2C manages this issue differently, and it will be described as follows.

DRS2C was devised by Oechsle [5]. This technique enhances the basic DRS in

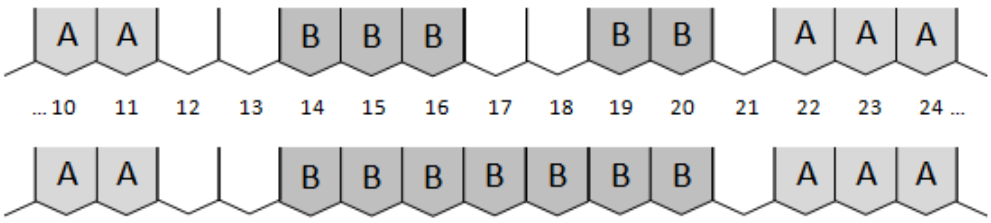


Figure 3.3: Segment of slots illustrated on the DRS2C idea

terms of dealing with unassigned slots, scaling the appropriate number of slots and estimating class confidence. To begin with the unassigned slot case, slots based on the DRS concept are managed to be defined as a class; however, some slots without training data have no class set. Therefore, the DRS2C technique will deal with those slots. If the nearest slots in the left and right of them are assigned in the same class, that class will be defined to the unassigned slots; otherwise the unassigned slots are not declared with any class in order to retain any margin between the classes.

A segment of slots is illustrated for the DRS2C idea as shown in Figure 3.3. The above slots are declared with a class based on the basic DRS technique. Grey slots are defined with either class A or B, and white slots have no class assigned. As can be noticed, slot 17 and 18 are not defined with a class, and the nearest slots in the left and right of them are class B. Consequently, DRS2C assigns class B to slot 17 and 18 as presented in the slots shown at the bottom of the figure. Conversely, slots 12, 13 and 21 are not declared with any class because the nearest slots to the left and right of them are assigned different classes, so the approach does nothing to them.

Normal DRS identifies a specific range of slots from -250 to +250. Is it too large for a small number of an output range or enough to cover a wide output

range? In order to reduce the restriction, DRS2C defines an arbitrary range of slots according to the number of classes used. Using this technique guarantees that every slot has a higher opportunity to be employed than the original approach.

The last proposed idea of DRS2C is to measure class confidence. Certainly, if all members falling in a slot have the same class, it is logical to label it with that class. However, if a slot contains members with different classes, such as 8 instances of class A, 4 instances of class B and 2 instances of class C, confidence computing of each class is based on the proportion of members that lead to the slot assigned by the winning class.

For image processing, segmentation is a common first step to retrieve an object of interest. If the segmented shape is not clear or not close to the real object, it will have an impact on object analysis. Accordingly, the user can verify the evolved segmenter on the training set. If the segmented result is not good enough, the user can repeat this step until it generates an effective one.

In step A5, after the training images are segmented, every segmented object is defined a class. This process needs to be done by the user. Each class of inputs has to be created to the system, and the user just chooses a class and clicks on an object area to define the class. Figure 3.4 presents a segmented object whose class label has been identified by the user.

A6 concerns feature selection for object classification. This process is similar to process A3. LDA, IG and PCM are again provided to evaluate the function effectiveness. Nevertheless, operations for object classification are different from those for background subtraction. The functions contain feature extraction based on shape, location-based and grey-scaled properties.

A7, the last main process for the training part, is generating an object classifier. This process is based on genetic programming, and the dynamic range

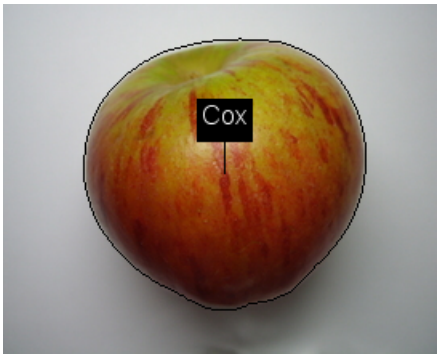


Figure 3.4: Segmented object defined a class

selection and thresholding approaches are again provided to produce an object classifier. Furthermore, *Intelligent Classification System* (ICS) based on binary decomposition was proposed in [5] to be an excellent choice for this task because it is able to cope with binary and multi-class classification problems.

Binary decomposition manages a multi-class problem by breaking it up into smaller pieces rather than solving the whole problem at one time. In other words, it decomposes the problem into a set of binary classification problems. One strategy of class partitioning methods is a hierarchical decomposition [132]. A sub-classifier is built to distinguish each class from the others that have not been involved. For example, a multi-class problem P comprises of n classes: $P = \{c_1, c_2, ..., c_n\}$. The first binary classifier deals with c_1 and $P - \{c_1\}$. When this classifier has been evolved, the training data for c_1 can be discarded, simplifying subsequent evolutions of binary classifiers. The second sub-classifier distinguishes between c_2 and $P - \{c_1, c_2\}$ then discards the training data for c_2. Sub-classifiers will similarly be generated until the last pair between c_{n-1} and c_n proceeds. Finally, all sub-classifiers are formed as a sequence of binary classifiers to solve the multi-class problem. For testing, if a sub-classifier of class A returns true, class A is defined to the instance; otherwise, a sub-classifier of class B pro-

ceeds and so on. Unlabelled data will fall down to the final stage. This can lead to a problem of pattern leakage down the hierarchical chain as addressed in [132].

ICS solves a multi-class problem by adapting the hierarchical decomposition approach. If the early sub-classifiers in the sequence are not accurate, errors will occur at the beginning and accumulate. In order to reduce the problem of a leakage of patterns down the chain, [5] proposed that the sub-classifiers should be chained in order of the class difficulty. First of all, the difficulty of each class of the training data is evaluated by a k-nearest-neighbour approach, resulting in the class confusion matrix to order the classes in terms of difficulty. A sub-classifier of each class is generated based on the binary decomposition principle, starting with the easiest to harder, and the hardest. Finally, the generated sub-classifiers are then assembled into a single multi-class classifier.

In steps B1–B3, after the system has evolved an object classifier, testing on unseen data is performed. For the testing part, the processes involve background subtraction and object classification respectively. Thus, the segmenter and the object classifier are employed again on the testing images. After an image is segmented, and a segmented object area is classified, then the result is presented via the interface.

3.2 Shortcomings of Jasmine

Jasmine is efficiently able to cope with many classification problems as presented in [5], such as skin lesion classification, number plate identification and hand gesture recognition. Nevertheless, some procedures and user interaction processes are not flexible enough. When Jasmine was applied to the research experiments focusing on agricultural produce, several seriously inconvenient difficulties were

identified.

1. It is not appropriate for use on large datasets because some processes need interaction from the user to deal with every image or object, such as class identification (process A5), segmentation (process B2) and object classification (process B3).

2. It is not automatic for class identification on training data. The user has to use a segmenter to segment every image and then identify the class of every segmented object. If there are a huge number of input images or each image contain many objects, it is a big burden for the user to manage this process.

3. Applying a segmenter and an object classifier on test data is not automatic. This problem is similar to the above item because the user has to apply the segmenter to every test image and then the object classifier to manage segmented object(s) in every image. Therefore, it is not only a serious difficulty for the training part, it is also a huge responsibility for the testing.

4. It does not provide enough extraction functions for object classification: the original system provides only shape, location-based and grey-scaled features. For instance, sample shape property extraction consists of bounding area, rectangularity, roundness, aspect ratio, balance, *etc.* Examples of location-based features are distance from the top, bottom, left and right, while grey-scale extraction comprises of computing on mean and standard deviation. These features can be used effectively for classification of some object types as shown in the experiments from [5], such as recognition on pasta types, hand gestures, skin lesion, handwriting digits and car plate numbers. However, other types of problem probably need further features to deal with.

For example, RGB, HSI and textural characteristics play a key role in agricultural produce grading as mentioned in section 2.2. Accordingly, more object properties need to be available in Jasmine for it to work on a wide range of input types.

5. It is difficult to analyse classification results. As the original program only returns classification results via the interface for each image, it is difficult to collect the program outputs over a complete database in order to analyse the classification accuracy.

The above shortcomings were improved in order to use the system more flexibly, effectively and appropriately for agricultural produce grading. The enhancements to Jasmine are presented in the next sections.

3.3 Enhancements for Large-scale Evaluations

For the processes described in section 3.1, some need user interaction in each image, such as process A2 and A5. If there is a bulk of images, it is time-consuming for the user. However, some images can be selected to employ for process A2. For process A5, every segmented object has to be defined a class; therefore, for some processes the author adapted the program to proceed more automatically.

3.3.1 Automatic class identification on training images

After the system acquires a background-subtractor, each training image has to be segmented, and obtained objects have to be assigned a class. The processes need to be managed by the user who must repeat the processes on all training images. If an image contains many objects, it is unavoidable for the user to define a class

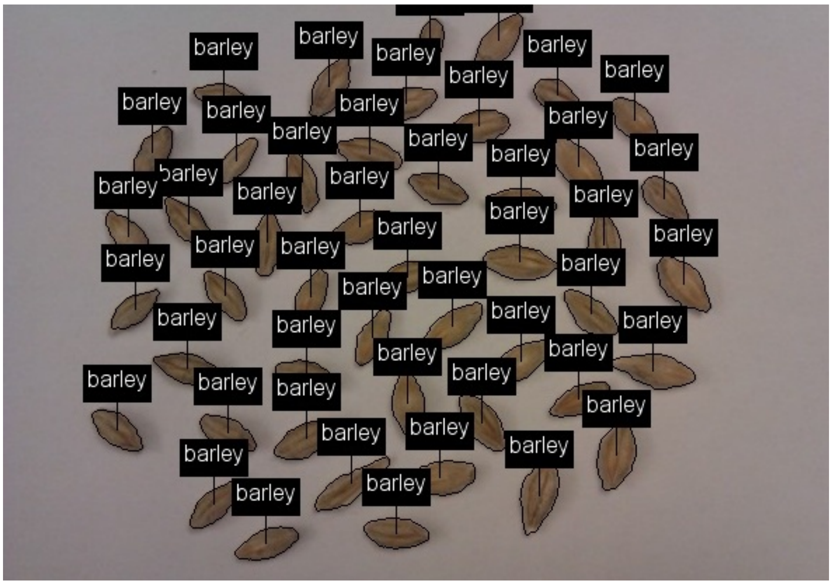


Figure 3.5: Segmented multiple objects defined a class automatically

in every single object. This problem is solved by the simple expedient of ensuring that image names start with the name of class. The program will assign a class name to every object by clicking only on a button. An example image assigned a class automatically is shown in Figure 3.5. The image named 'barley1' contains barley kernels, so all segmented objects are assigned class 'barley' automatically.

3.3.2 Automatic class classification on testing images

After an object classifier is generated, it is ready to use for testing classification. The original program provided an operation to classify an object image and show the result on the interface. The user has to do this process for every test image, so it is not appropriate for a large number of testing data. Thus, the program was modified to classify all testing images automatically and export the results to a file for classification accuracy analysis. The results presented in later chapters all use this functionality.

3.4 Extended Colour and Texture Operations for Object Classification

It is well known that colour is used significantly for segmentation tasks. Various colour and texture operations were provided in the original Jasmine for background subtraction. For object classification, only shape, location-based and grey-scale properties were supported by the system. However, section 2.2 has observed that many colour and texture extraction functions are employed to extract object properties for agricultural produce inspection.

For example, RGB and HSI modes are mainly employed to evaluate the maturity and quality of various fruit and vegetables: Lino *et al* [27] demonstrated the ripening progress of tomatoes based on the RGB colour space. They pointed out that red colour was noticed to increase while green and blue intensities decreased according to the ripeness levels. A threshold on $\frac{R}{G}$ was effectively used for pomegranate arils grading shown in Blasco *et al* [19]. RGB was also employed significantly to estimate ripeness of oil palm fruit bunches [20–22].

Not only RGB was applied usefully to extract features for apple grading presented by Leemans and Destain [51], HSI was also employed for this work as described by Xiaobo *et al* [48]. Xu and Zhao [57] proved the performance of using hue in the light intensity conditions, and presented that the hue component could be applied efficiently to determine the maturity of strawberries. Wen *et al* [54] also employed hue values to characterize colour features of citrus fruits. Jin *et al* [23] created a threshold based on HSI to identify defective areas of potatoes.

In addition to using colour to inspect the quality and maturity, it can be employed effectively for variety discrimination. For instance, the RGB and HSI components were used widely to generate colour features to classify grain and rice

varieties [59, 65–68, 72].

Texture determination is also useful for agricultural produce inspection. For example, Paliwal *et al* [66] applied grey level co-occurrence and grey level length matrix models to grain kernel identification. The models were used to generate various functions to extract textural features. Similarly, Kavdir and Guyer [49] adopted the grey co-occurrence method to extract textural properties to classify different quality categories of apples. Shahin and Symons [62] also indicated that textural characteristics derived from the grey level co-occurrence matrix were well employed to inspect lentils. Instead of using grey level texture, Kim *et al* [58] applied the previous method to the HSI space to create HSI texture features for inspection on grapefruit peel diseases.

Accordingly, colour and texture features should be provided in Jasmine to support the system for object classification. Therefore, the system was extended with colour operations to determine object regions in binary, RGB and HSI colour spaces, making them available for use in classification as well as segmentation. Sample colour extraction functions consist of normalization, mean, standard deviation, $c_1c_2c_3$ and $l_1l_2l_3$. Furthermore, textural operations, such as co-occurrence and run length approaches, were reinforced and implemented in RGB and HSI components. The mentioned colour and texture extraction functions were installed in the system because they are simple, fast to calculate and used by others [23, 25, 44, 47–49, 54, 58–60, 62, 65–68, 72].

With the above operators incorporated, the modified Jasmine contains 29 colour and 123 texture operators. It retains the original morphological features to analyse objects in 2D, of which there are 32 operations. Sample morphological operations are shown in section 2.1.2. In total, it obtains 184 operations for 2D object analysis.

As the program operates automatically, it is simple and friendly to use via the user interface. Moreover, it was designed to support a large number of samples, and the functions of the original Jasmine were well organized. Consequently, it does not takes a long time to extract all the features of a big dataset (approximately 1 second per image). Although many feature extraction functions are provided, they are evaluated, and the most effective functions are employed to generate an object classifier.

3.4.1 Colour features

For the original Jasmine, colour features were used for segmentation, and morphological and intensity operators were employed for object classification. For most agricultural produce, colour characteristics are significant for manual inspection and play a key role in existing research on agricultural produce grading. Consequently, various colour operators should be incorporated in the object classification part. Therefore, the colour extraction functions used for background subtraction from the original system were applied to the modified program for object classification.

For 2D object analysis, colour properties are directly extracted from segmented object areas. The colour extraction functions return outputs in terms of binary, RGB and HSI modes. Sample operations used to extract them are described in section 2.1.3 including normalization, mean, standard deviation, $c_1c_2c_3$ and $l_1l_2l_3$.

3.4.2 Textural features

Texture is normally considered as characteristics of a variation in brightness of images. Local texture analysis involves comparing the brightness of a pixel to

neighbour pixels or a small region. Therefore, texture cannot be identified for a point.

For 2D object analysis, texture features are also retrieved directly from segmented object regions. The presented functions based on grey level co-occurrence and grey level run length approaches below were employed to compute textural properties, and the functions were also adapted to extract textural features in RGB and HSI components.

Grey level co-occurrence

The grey level co-occurrence matrix (GLCM) [13] is an approach to present the distance and angular spatial relationships of grey levels in a specified region of an image. The relationship occurrence changes rapidly with distance in fine textures, slowly in coarse textures.

Element P_{ij} of the grey level co-occurrence is a measurement of the occurrence possibility of grey value i and j separated in a defined distance (d) in four directions including horizontal (0°), vertical (90°), right diagonal (45°: bottom left to top right), and left diagonal (135°: bottom right to top left). The unnormalized frequencies for the four directions are defined as shown in (3.1)-(3.4).

$$P(i,j,d,0) = \#\{\ [(k,l),(m,n)]\ :\ k-m=0,\ |l-n|=d,\ I(k,l)=i, Iat(m,n)=j\ \} \tag{3.1}$$

$$P(i,j,d,45) = \#\{\ [(k,l),(m,n)]\ :\ (k-m=d,\ l-n=-d)\ or\ (k-m=-d,\ l-n=d),\ I(k,l)=i,\ I(m,n)=j\ \} \tag{3.2}$$

(1,1)	(1,2)	(1,3)	(1,4)
(2,1)	(2,2)	(2,3)	(2,4)
(3,1)	(3,2)	(3,3)	(3,4)
(4,1)	(4,2)	(4,3)	(4,4)

Figure 3.6: Sub-image illustrated by coordinates

$$P(i,j,d,90^\circ) = \#\{\ [(k,l),(m,n)]\ :\ k-m=d,\ l-n=0,$$
$$I(k,l)=i,\ I(m,n)=j\ \} \tag{3.3}$$
$$P(i,j,d,135^\circ) = \#\{\ [(k,l),(m,n)]\ :\ (k-m=d,\ l-n=d)\ or$$
$$(k-m=-d,\ l-n=-d),\ I(k,l)=i,\ I(m,n)=j\ \} \tag{3.4}$$

where # is the number of elements in the set, (k,l) is coordinates of the image I with grey level i, and (m,n) is coordinates of the image I with grey level j.

From a sub-image with 4×4 pixels as illustrated in Figure 3.6, 0° angular relationships (R_{0°) from grey level i to grey level j for distance $d = 1$ can be sampled to analyse by using (3.1), and it results in a set of the relationship as shown in (3.5).

$$\begin{aligned} R_{0^\circ} = \{&((1,1),(1,2)),((1,2),(1,1)),((1,2),(1,3)),((1,3),(1,2)),\\ &((1,3),(1,4)),((1,4),(1,3)),((2,1),(2,2)),((2,2),(2,1)),\\ &((2,2),(2,3)),((2,3),(2,2)),((2,3),(2,4)),((2,4),(2,3)),\\ &((3,1),(3,2)),((3,2),(3,1)),((3,2),(3,3)),((3,3),(3,2)),\\ &((3,3),(3,4)),((3,4),(3,3)),((4,1),(4.2)),((4,2),(4,1)),\\ &((4,2),(4,3)),((4,3),(4,2)),((4,3),(4,4)),((4,4),(4,3))\} \end{aligned} \tag{3.5}$$

0	0	1	1
0	0	1	2
1	1	2	3
2	3	3	3

Figure 3.7: Sub-image of grey levels

GLCM		Grey level			
		0	1	2	3
Grey level	0	#(0,0)	#(0,1)	#(0,2)	#(0,3)
	1	#(1,0)	#(1,1)	#(1,2)	#(1,3)
	2	#(2,0)	#(2,1)	#(2,2)	#(2,3)
	3	#(3,0)	#(3,1)	#(3,2)	#(3,3)

Figure 3.8: Spatial co-occurrence matrix form

Figure 3.7 illustrates a small segment (4×4 pixels) of an image. It presents in grey levels, ranging from 0 to 3. A normal form of a spatial co-occurrence matrix is sampled in Figure 3.8. Because there are 4 grey levels, the matrix is formed symmetrically with 4×4 elements. A couple of examples to elaborate co-occurrence calculations for the distance $d = 1$ in direction 0° of the sub-image is described as follows.

The element $P_{0^\circ}(0,0)$ represents the probability of occurrence that a pixel with grey level 0 is horizontally next to a neightbour that also has the grey level 0. Due to matrix symmetry, it is scanned in both the left to right and right to left directions. Therefore, in Figure 3.7 $P_{0^\circ}(0,0)$ happens four times including {(0,0),(0,1)}, {(1,0),(1,1)}, {(0,1),(0,0)} and {(1,1),(1,0)}.

The other example for the element $P_{0^\circ}(0,1)$, it is the number of times that a pixel with grey value 0 is horizontally adjacent to a pixel with grey value 1. It is still looked in the left to right and also right to left. Thus, there are two times for

0°		Grey level			
		0	1	2	3
Grey level	0	4	2	0	0
	1	2	4	2	0
	2	0	2	0	2
	3	0	0	2	4

(a) Direction 0°

45°		Grey level			
		0	1	2	3
Grey level	0	2	2	0	0
	1	2	4	1	0
	2	0	1	2	1
	3	0	0	1	2

(b) Direction 45°

90°		Grey level			
		0	1	2	3
Grey level	0	4	2	0	0
	1	2	2	3	1
	2	0	3	0	2
	3	0	1	2	2

(c) Direction 90°

135°		Grey level			
		0	1	2	3
Grey level	0	2	2	1	0
	1	2	0	1	3
	2	1	1	0	1
	3	0	3	1	0

(d) Direction 135°

Figure 3.9: Spatial co-occurrence calculations in each direction

GLCM		Grey level			
		0	1	2	3
Grey level	0	12	8	1	0
	1	8	10	7	4
	2	1	7	2	6
	3	0	4	6	8

Figure 3.10: Grey level co-occurrence matrix of the sub-image shown in Figure 3.7

$P_{0°}(0,1)$ comprising of {(0,1),(0,2)} and {(1,1),(1,2)}.

In conclusion, spatial co-occurrence calculations for each direction are shown in Figure 3.9 that result in a total grey level co-occurrence matrix of the sample sub-image shown in Figure 3.10. Texture features can be derived from the co-occurrence matrix, and operations in [75] adopted for Jasmine to extract textural properties based on GLCM are shown in (3.6)–(3.13).

$$Mean\,(\mu) = \sum_{i=1}^{N}\sum_{j=1}^{N} i.p(i,j) \tag{3.6}$$

$$Variance\,(\sigma^2) = \sum_{i=1}^{N}\sum_{j=1}^{N} (i-\mu)^2.p(i,j) \tag{3.7}$$

$$Uniformity = \sum_{i=1}^{N}\sum_{j=1}^{N} (p(i,j))^2 \tag{3.8}$$

$$Entropy = -\sum_{i=1}^{N}\sum_{j=1}^{N} p(i,j)log(p(i,j)) \tag{3.9}$$

$$Maximum\,probability = max(p(i,j)) \tag{3.10}$$

$$Correlation = \sum_{i=1}^{N}\sum_{j=1}^{N} \frac{(i-\mu)(j-\mu)p(i,j)}{\sigma^2} \tag{3.11}$$

$$Homogeneity = \sum_{i=1}^{N}\sum_{j=1}^{N} \frac{p(i,j)}{1+(i-j)^2} \tag{3.12}$$

$$Cluster\,shade = \sum_{i=1}^{N}\sum_{j=1}^{N} (i+j-2\mu)^3 p(i,j) \tag{3.13}$$

where $p(i,j)$ is the $(i,j)^{th}$ entry in the GLCM, i and j are grey levels, and N is the number of grey levels.

Grey level run lengths

The grey level run length matrix (GRLM) [14] is an idea to evaluate a set of consecutive pixels with the same grey value in 0°, 45°, 90° and 135° directions.

0°		Run length			
		1	2	3	4
Grey level	0	0	2	0	0
	1	1	2	0	0
	2	3	0	0	0
	3	1	0	1	0

(a) Direction 0°

45°		Run length			
		1	2	3	4
Grey level	0	2	1	0	0
	1	2	0	1	0
	2	1	1	0	0
	3	2	1	0	0

(b) Direction 45°

90°		Run length			
		1	2	3	4
Grey level	0	0	2	0	0
	1	3	1	0	0
	2	3	0	0	0
	3	2	1	0	0

(c) Direction 90°

135°		Run length			
		1	2	3	4
Grey level	0	2	1	0	0
	1	5	0	0	0
	2	3	0	0	0
	3	4	0	0	0

(d) Direction 135°

Figure 3.11: Spatial run length calculations in each direction

The number of neighbouring pixels with the same grey-scale is able to present fineness or coarseness of textures: a smaller number of these indicates finer texture whereas a larger number illustrates coarser texture. The run lengths in different directions can also elaborate texture description.

A grey level run length matrix includes grey level i, run length j and the number of times that a run of length j occurs for level i in the given direction. For instance, in Figure 3.7 in case of direction 0, level 0 and level 1 lie on run length 2 twice; meanwhile, level 3 lies on run length 3 once. Therefore, grey level run length calculations of all principal directions and a summation of grey level run length matrix of Figure 3.7 can be presented as shown in Figure 3.11 and Figure 3.12 respectively. Various formulae were created to estimate textural features based on GLRM [75], and they are applied to use as shown in (3.14)–(3.19).

GLRM		Run length			
		1	2	3	4
Grey level	0	4	6	0	0
	1	11	3	1	0
	2	10	1	0	0
	3	9	2	1	0

Figure 3.12: Grey level run length matrix of Figure 3.7

$$R = \sum_{i=1}^{N}\sum_{j=1}^{M} q(i,j)$$

$$\textit{Short runs} = \frac{1}{R}\sum_{i=1}^{N}\sum_{j=1}^{M}\frac{q(i,j)}{j^2} \tag{3.14}$$

$$\textit{Long runs} = \frac{1}{R}\sum_{i=1}^{N}\sum_{j=1}^{M} j^2 q(i,j) \tag{3.15}$$

$$\textit{Grey level non-uniformity} = \frac{1}{R}\sum_{i=1}^{N}\left(\sum_{j=1}^{M} q(i,j)\right)^2 \tag{3.16}$$

$$\textit{Run length non-uniformity} = \frac{1}{R}\sum_{j=1}^{M}\left(\sum_{i=1}^{N} q(i,j)\right)^2 \tag{3.17}$$

$$\textit{Run percentage} = \frac{R}{\sum_{i=1}^{N}\sum_{j=1}^{M} j.q(i,j)} \tag{3.18}$$

$$\textit{Entropy} = \frac{1}{R}\sum_{i=1}^{N}\sum_{j=1}^{M} q(i,j)log(q(i,j)) \tag{3.19}$$

where $q(i,j)$ is the $(i,j)^{th}$ entry in the GLRM, i is a grey level, j is a run length value, N is the number of grey levels, and M is the number of different run lengths

occurring.

3.5 Assessing Vision System Performance

An important aim of this work is to assess how effective evolved vision systems are, both in isolation and compared to other approaches. Assessing an algorithm in isolation is now conventionally done by calculating the number of true positives *etc.* and hence summary measures such as sensitivity and specificity. For comparing algorithms however, no single approach has yet become well-established. Following [133–135], an appropriate approach is to use McNemar's test as that makes no assumptions about any error distributions involved and guarantees that binomial statistics apply, which are well-understood. Before introducing McNemar's test, the use of null hypothesis testing is described, and the remainder of this section presents the Bonferroni correction and the principles of the confusion matrix.

3.5.1 Null hypothesis testing

Hypothesis testing aims to reach a conclusion to reject or not reject a hypothesis about data. There are two basic types of hypotheses: the null hypothesis (H_0) and the alternative hypothesis (H_1) [136, 137]. For example, comparing the mean of ages of people in a capital and another city can be made. A null hypothesis can be formulated as

H_0*: the mean values of people's ages in the cities have no difference*

written in statistics as $\mu_1 = \mu_2$ or $\mu_1 - \mu_2 = 0$. Any hypothesis different from H_0 can be an alternative hypothesis, such as

H_1*: mean values of people's ages in the cities are different*

represented as $\mu_1 \neq \mu_2$ or $\mu_1 - \mu_2 \neq 0$. H_1 indicates that the means differ, but it does not indicate which mean value is higher, so the hypothesis is considered using a two-tailed test. If H_1 is formulated in a particular case, such as

H_1: the mean value of the capital is higher than that of the other city

written as $\mu_1 > \mu_2$ or $\mu_1 - \mu_2 > 0$. This hypothesis is considered by a one-tailed test.

There are two fundamental errors of hypothesis testing: Type I error and Type II error. A Type I error happens if a true null hypothesis is inadvertently rejected. α is the probability of a Type I error, and the value of α involves the significance level of the test. A Type II error occurs when a false null hypothesis is accepted. β represents the probability of committing a Type II error.

For testing a null hypothesis, a decision rule is established to define the significance level. Commonly, statistical tests use $\alpha = 0.05$, meaning there are about 5 chances in 100 that the hypothesis would be rejected when it should be accepted, and the probability of making the right decision is 95%. This can establish the boundaries of the regions of acceptance and rejection under a standard distribution (illustrated later). Table 3.1 presents the association between degrees of confidence and Z scores (critical values) for two-tailed and one-tailed predictions. For example, using 95% confidence for a two-tailed test of Normally-distributed data obtains the critical Z value of 1.96.

Figure 3.13 shows a Normal distribution and illustrates these parameters. Considering a two-tailed test, the white area under the Normal curve represents the region of acceptance $(1-\alpha)$. The grey areas represent the regions of rejection, and the area of each of the two rejection regions is $\frac{\alpha}{2}$. The null hypothesis is rejected if the result falls in either of the two rejection regions; otherwise, it is accepted.

Table 3.1: Association between Z scores and confidence limits

Z value	Degree of confidence Two-tailed prediction	Degree of confidence One-tailed prediction
1.645	90.0%	95.0%
1.960	95.0%	97.5%
2.326	98.0%	99.0%
2.576	99.0%	99.5%

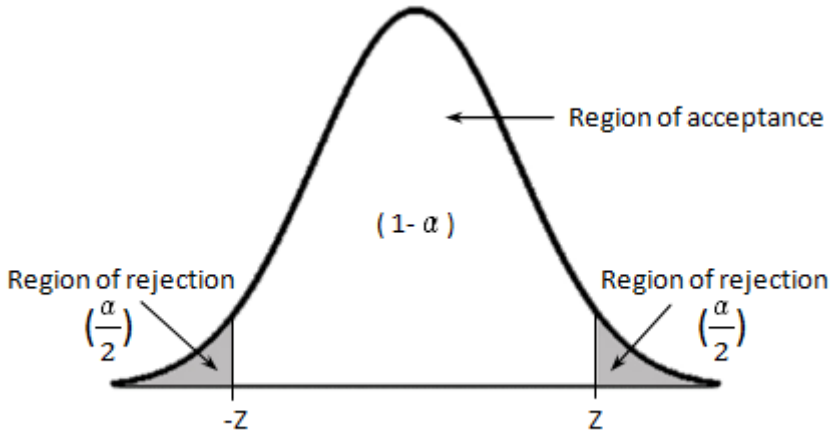


Figure 3.13: Illustration on the regions of acceptance and rejection

3.5.2 McNemar's test

Some statistical tests are applied to comparisons of algorithms in order to prove whether the differences are real or random [138]. For example, the Wilcoxon signed-ranks test is equivalent to the paired t-test in non-parametric statistical procedures, and it is appropriate for comparison of two classifiers. The Friedman test is a non-parametric alternative to the repeated-measures ANOVA, and it can be employed for comparison of multiple classifiers when the distributions involved are known to be Normal.

An alternative non-parametric test employed in this research is McNemar's test, which is appropriate for non-Normally distributed data, or when the data distribution is unknown. In statistics, McNemar's test is used to assess the significance of the difference between paired proportions, such as comparing the ability of trainees before and after training. In computing, it can be employed to make a comparison of the performance of two algorithms. If there are more than two algorithms, matched pairs of algorithms must be compared. There are three possible outcomes from a comparison: both algorithms yield the same result (either success or failure); or the first algorithm succeeds and the second fails; or the first fails and the second succeeds. This forces the results into a trinomial distribution; but since we are not interested in the case where the algorithms yield the same result, McNemar's test considers only the latter two cases. The marginal distribution for two outcomes of a trinomial is a binomial distribution [139]. Of course, for a reasonable number of samples, the binomial distribution is closely approximated by a Normal one, so the critical value of 1.96 for 95% confidence alluded to earlier is commonly used. A 2×2 table is used to analyse data:

	Algorithm B succeeded	Algorithm B failed
Algorithm A succeeded	N_{ss}	N_{sf}
Algorithm A failed	N_{fs}	N_{ff}

where N_{ss} is the number of tests in which both algorithms succeeded, N_{sf} is the number of tests for which algorithm A succeeded and algorithm B failed, N_{fs} is the number of tests for which algorithm A failed and algorithm B succeeded, and N_{ff} is the number of tests where both algorithms failed.

McNemar's test is formed as follows:

$$\chi^2 = \frac{(|N_{sf} - N_{fs}| - 1)^2}{N_{sf} + N_{fs}} \tag{3.20}$$

where the -1 is a continuity correction. The Z score (standard score) is calculated using the square root of this:

$$Z = \frac{|N_{sf} - N_{fs}| - 1}{\sqrt{N_{sf} + N_{fs}}} \tag{3.21}$$

McNemar's test not only records outcomes of algorithms (success or failure) but also encapsulates performance differences of the algorithms. If the performances of the algorithms are similar, the Z value will be close to 0. In contrast, if one algorithm's performance exceeds the other's, the Z score increases. Confidence limits are related to Z scores considered in two-tailed and one-tailed predictions as presented in Table 3.1. The two-tailed prediction is appropriate to examine whether the performances of algorithms *differ*, while the one-tailed prediction is used to assess whether the performance of one algorithm *is superior* to that of the other.

As mentioned above, statistical tests commonly use the degree of confidence of 95.0% for two-tailed prediction ($\alpha = 0.05$). It means the probability of the difference in performance arising by chance is $\sim \frac{1}{20}$. This leads to the critical value = 1.96. If a calculated Z value is higher than the critical value (1.96), the performance differences of algorithms are statistically significant. N_{sf} and N_{fs} represent the reliability of the difference. If both N_{sf} and N_{fs} are large, *i.e.* $N_{sf} + N_{fs} \gtrsim 20$, one can be confident that this performance difference is genuine and not an artefact of the data.

3.5.3 Bonferroni correction

If the number of hypotheses increases, the probability of obtaining a significant result also increases [140]. In order to avoid many spurious positives, the value of α needs to be lowered. A method used to reduce α is the Bonferroni correction, which is an adjustment made to α when several statistical tests are performed simultaneously on a dataset [141]. The significance cut-off is calculated by $\frac{\alpha}{n}$, where α is the significance level, such as 0.05, and n is the number of hypotheses or comparisons.

For example, if there are 3 comparisons using the significance level of 0.05 (Z=1.96), the cut-off should be $\frac{0.05}{3} = 0.0167$. Now, $\alpha = 0.0167$, and it can be translated to a new critical Z value using the standard normal distribution table shown in Table A.1. An input value (P value) used to translate is calculated by $1 - \frac{\alpha}{2}$. The input of the example is $1 - \frac{0.0167}{2} \approx 0.99165$, so the adjusted critical Z value will be 2.39.

Other alternative corrections were reviewed in [143]. For example, Holm [144] proposed the sequentially rejective Bonferroni test to modify a single-stage testing

Table 3.2: Confusion matrix of two-class classification

	Predicted True	Predicted False
Actual True	True Positive (TP)	False Negative (FN)
Actual False	False Positive (FP)	True Negative (TN)

of the Bonferroni method in order to perform multi-stage testing. Hochberg [145] adapted the hypothesis rejection condition of Holm's procedure. Many approaches were presented to control the family-wise error rate while other ideas were shown to control the false discovery rate as proposed by Benjamini and Hockberg [146].

3.5.4 Confusion matrix

A confusion matrix summarises the ability of a classification system by indicating its correct predictions and errors. For instance, Table 3.2 demonstrates a confusion matrix for two-class classification, where the two-letter acronyms have the following meanings:

TP: the number of occasions where an algorithm succeeded and returned the correct class when it should have done;

TN: the number of occasions where an algorithm failed when it should have done;

FP: the number of occasions where an algorithm succeeded but returned an incorrect class;

FN: the number of occasions where an algorithm failed when it should have succeeded;

Several commonly-used measures of performance can be calculated from these values [147].

$$Precision = \frac{TP}{TP + FP} \tag{3.22}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.23}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.24}$$

$$F\text{-}measure = \frac{2(Precision \times Recall)}{(Precision + Recall)} \tag{3.25}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{3.26}$$

$$Specificity = \frac{TN}{TN + FP} \tag{3.27}$$

$$Positive\ likelihood = \frac{Sensitivity}{1 - Specificity} \tag{3.28}$$

$$Negative\ likelihood = \frac{Specificity}{1 - Sensitivity} \tag{3.29}$$

3.6 Assessment of the Performance of the Enhanced Jasmine

As mentioned previously, the original Jasmine employs only shape, location-based and grey-scale feature extraction for object classification. Hence, we should expect it to be inadequate to recognize some types of material with similar shape but different colour or texture. Therefore, the system would solve a wider range of classification problems if further operations were provided. Accordingly, the modified system was extended with new colour and texture operations as described in 3.4.

In order to verify the adapted program, classification experiments on sticky index markers and agricultural produce including persimmons, oranges and apples were done[1]. Each experiment consists of twenty sub-experiments, and the samples were randomly selected for training and test sets. McNemar's test was employed to compare the classification performance between the original system and the modified program.

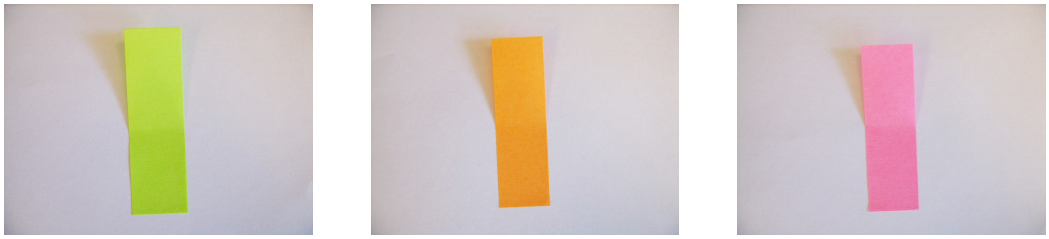
3.6.1 Experiment on sticky index marker classification

This task intends to classify objects of the same size and shape but different colours. A set of sticky index markers is a good type of objects to demonstrate this task because their size and shape are equal, and they are differently coloured. The set of images consisted of sticky index markers coloured green, orange and pink, and there were forty of each. Half the numbers of samples were selected randomly for training, the remaining samples being used for testing. Sample images of the sticky index markers are shown in Figure 3.14.

The classification results are presented in Table 3.3. It is obvious that all classifiers generated by the modified system were able to achieve perfect accuracy. Although the operations of the original program are based on morphological feature extraction, it contains a few functions on grey-scale extraction. Accordingly, the original engine could reach a peak performance of 66.7%, but many results are less than 50% accurate.

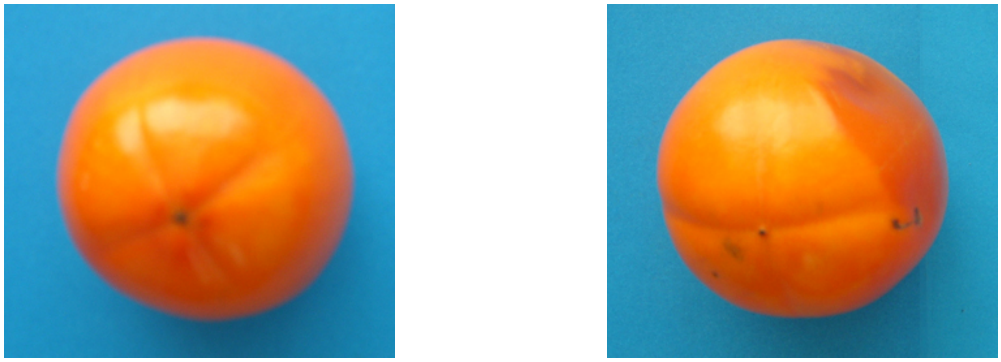
The tables of results in this thesis contain numbers that have *not* been rounded to a particular number of significant figures. This is partly because of the fact that the binary nature of computer arithmetic itself can introduce errors (for example,

[1]The images used for the experiments of this thesis are available at http://vase.essex.ac.uk/datasets/



(a) Green colour (b) Orange colour (c) Pink colour

Figure 3.14: Sample images of sticky index markers

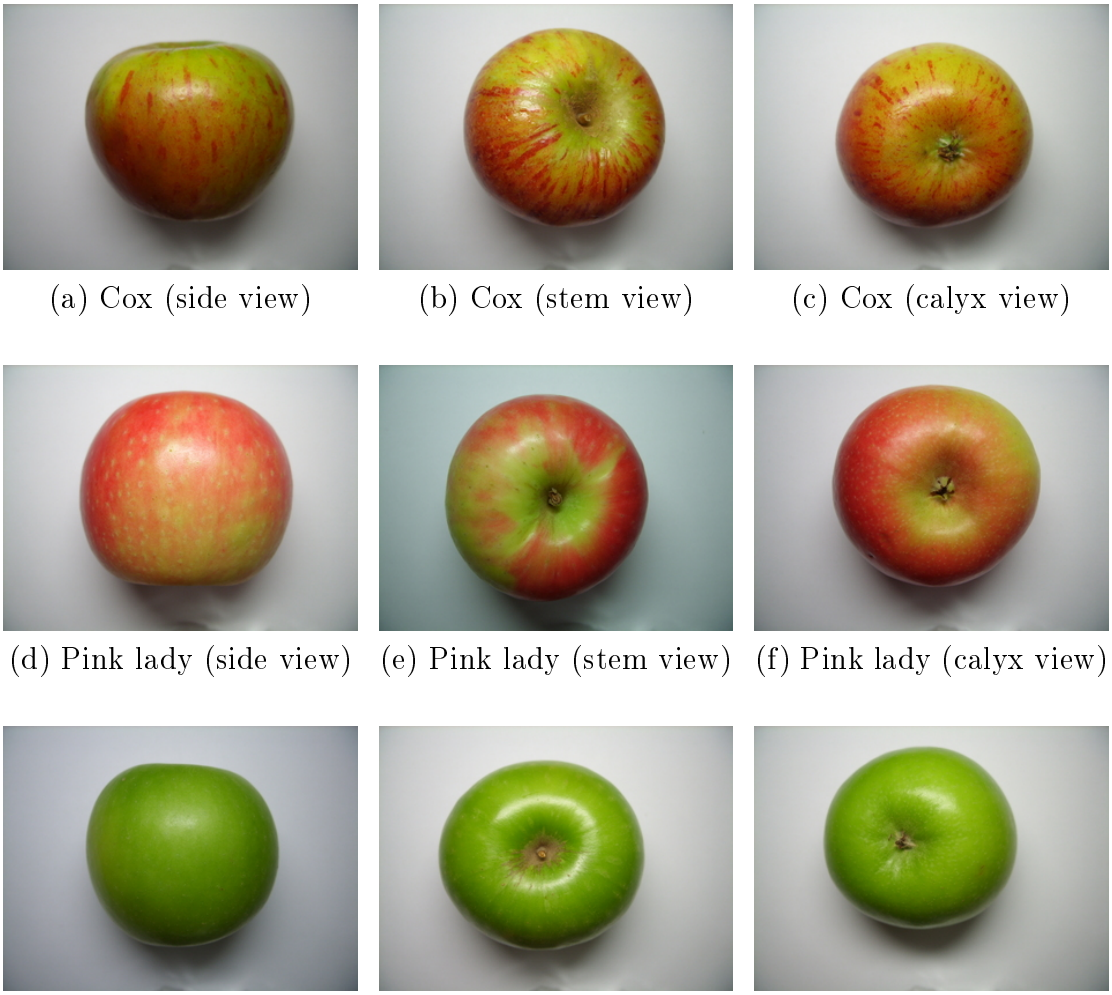


(a) Good colour skin (b) Bruised skin

Figure 3.15: Sample images of persimmons

(a) Smoothly-coloured skin (b) Scarred skin

Figure 3.16: Sample images of mandarins



(a) Cox (side view) (b) Cox (stem view) (c) Cox (calyx view)

(d) Pink lady (side view) (e) Pink lady (stem view) (f) Pink lady (calyx view)

(g) Granny smith (side view) (h) Granny smith (stem view) (i) Granny smith (calyx view)

Figure 3.17: Sample images of different varieties of apples captured in several views

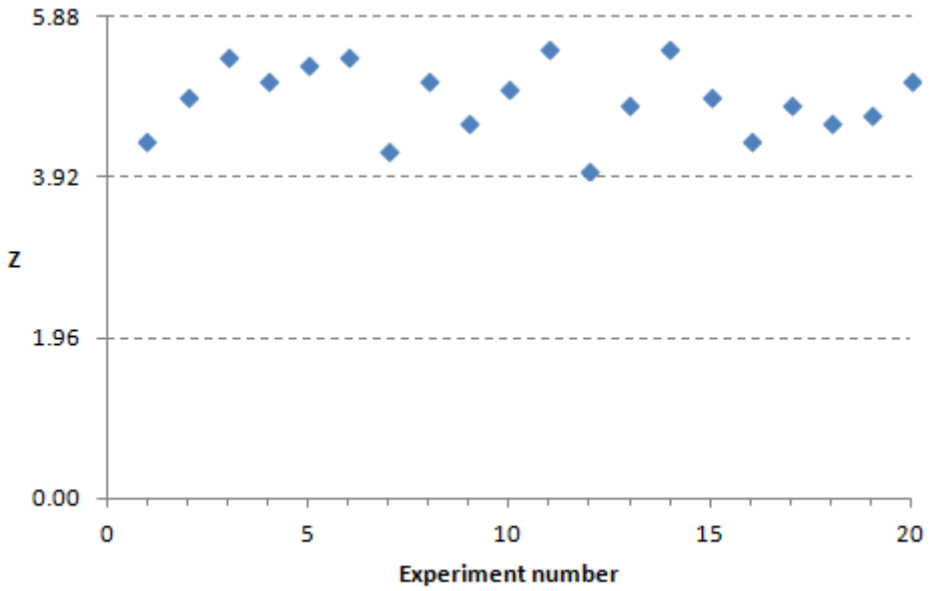


Figure 3.18: Illustration on Z scores of the experiment on sticky index marker classification

$\frac{1}{3}$ cannot be represented accurately) and partly because each experiment involves a different number of detected features, so that rounding results would have to be done manually, an error-prone process. Hence, values presented in tables need to be interpreted in light of the number of features involved in the experiments.

The corresponding Z values are shown in Table 3.3 and illustrated in Figure 3.18. All Z values are over 4, significantly greater than the critical value (1.96). This demonstrates that differences in performance are significant, and the classification achievement of the modified system dramatically exceeds that of the original.

Table 3.3: Classification results and Z scores of the experiment on sticky index marker classification

Experiment number	Classification accuracy results (%)		Z score	Superior performance
	Original system	Modified system		
1	65.0	100.0	4.36	Proposed system
2	56.7	100.0	4.90	Proposed system
3	48.3	100.0	5.39	Proposed system
4	53.3	100.0	5.10	Proposed system
5	50.0	100.0	5.29	Proposed system
6	48.3	100.0	5.39	Proposed system
7	66.7	100.0	4.25	Proposed system
8	53.3	100.0	5.10	Proposed system
9	61.7	100.0	4.59	Proposed system
10	55.0	100.0	5.00	Proposed system
11	46.7	100.0	5.48	Proposed system
12	70.0	100.0	4.01	Proposed system
13	58.3	100.0	4.80	Proposed system
14	46.7	100.0	5.48	Proposed system
15	56.7	100.0	4.90	Proposed system
16	65.0	100.0	4.36	Proposed system
17	58.3	100.0	4.80	Proposed system
18	61.7	100.0	4.59	Proposed system
19	60.0	100.0	4.69	Proposed system
20	53.3	100.0	5.10	Proposed system
Best	66.7	100.0		
Mean	56.6	100.0		
s.d.	6.9	0.0		

3.6.2 Experiment on persimmon bruise discrimination

A reason for causing bruising injury of persimmons is dropping them from height when they are harvested. Immediately after the bruising, the injury is still invisible, but it gradually appears during storage. The samples consisted of good colour skin (56 images) and blemished skin (60 images). They were captured only in a top view, and an example image of each group is shown in Figure 3.15. They were randomly selected, 40% for training process and 60% for testing.

Table 3.4 presents the classification results and performance comparison by Z scores. Although the original system was able to reach the highest performance three times, it only hit at 75.7% accuracy. Whereas, using the additional colour and texture features achieved the peak at 97.1% accuracy, and all results are over 87%. The Z values are also shown in Figure 3.19. Not all experiments demonstrate significant differences in performance: the Z scores of 19 sub-experiments are greater than the critical value while the other is only 1.66. However, almost all sub-experiments of the modified program significantly outperformed the original.

3.6.3 Experiment on mandarin peel inspection

This task aims to inspect the defective skin of mandarins in a cultivar of Sai Nam Pueng. 50 images of smoothly-coloured skin and 50 images exhibiting scarred skin were employed for inspection; sample images are shown in Figure 3.16. 30% of the samples were selected randomly for training processes and the remainder for testing.

The classification accuracy results and Z scores are presented in Table 3.5, and the Z scores are also shown in Figure 3.16. As can be seen, all accuracy results of the proposed classifiers are better than those generated by the original Jasmine,

Table 3.4: Classification results and Z scores of the experiment on persimmon bruise discrimination

Experiment number	Classification accuracy results (%)		Z score	Superior performance
	Original system	Modified system		
1	65.7	90.0	3.34	Proposed system
2	82.9	92.9	1.66	-
3	71.4	91.4	2.77	Proposed system
4	61.4	90.0	3.59	Proposed system
5	68.6	91.4	3.20	Proposed system
6	64.3	88.6	3.08	Proposed system
7	64.3	92.9	3.59	Proposed system
8	74.3	97.1	3.35	Proposed system
9	64.3	92.9	3.88	Proposed system
10	68.6	88.6	2.77	Proposed system
11	67.1	91.4	3.08	Proposed system
12	75.7	94.3	3.33	Proposed system
13	72.9	87.1	2.01	Proposed system
14	56.6	90.8	4.29	Proposed system
15	75.7	97.1	3.21	Proposed system
16	72.9	95.7	3.35	Proposed system
17	75.7	94.3	2.75	Proposed system
18	72.9	88.6	2.18	Proposed system
19	68.6	92.9	3.34	Proposed system
20	64.3	90.0	3.33	Proposed system
Best	75.7	97.1		
Mean	69.4	91.9		
s.d.	6.2	2.8		

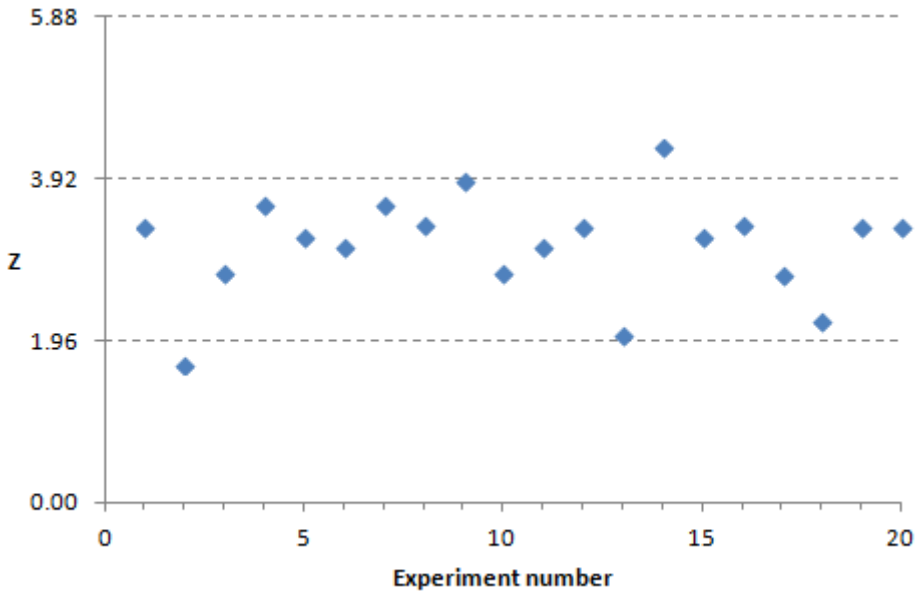


Figure 3.19: Illustration on Z scores of the experiment on persimmon bruise discrimination

and their achievements differed by more than 20% for 13 sub-experiments. The modified program achieved 89.6% mean accuracy, over 24% better than that produced by the original system. In addition, all Z values are significantly larger than the critical value. The obvious conclusion here is that the inclusion of texture description and analysis operators greatly improves the ability of Jasmine to solve problems in the agricultural domain. Therefore, the additional texture properties were effectively used for this task.

3.6.4 Experiment on apple variety classification

Three varieties of apples (pink lady, cox and granny smith) were captured from several views comprising stem-end, calyx and side areas. There were 48 images for pink lady, 56 images for cox and 60 images for granny smith. Sample images are presented in Figure 3.17. The samples were randomly chosen, 30% for training

Table 3.5: Classification results and Z scores of the experiment on mandarin peel inspection

Experiment number	Classification accuracy results (%)		Z score	Superior performance
	Original system	Modified system		
1	54.3	91.4	4.56	Proposed system
2	67.1	95.7	3.73	Proposed system
3	68.6	87.1	2.62	Proposed system
4	60.0	87.1	3.34	Proposed system
5	67.1	85.7	2.40	Proposed system
6	67.1	87.1	2.46	Proposed system
7	71.4	88.6	2.25	Proposed system
8	70.0	85.7	2.29	Proposed system
9	65.7	85.7	2.55	Proposed system
10	55.7	90.0	4.35	Proposed system
11	68.6	91.4	2.83	Proposed system
12	78.6	90.0	2.47	Proposed system
13	64.3	91.4	3.46	Proposed system
14	71.4	90.0	2.62	Proposed system
15	52.9	94.3	4.73	Proposed system
16	65.7	87.1	2.92	Proposed system
17	70.0	90.0	2.65	Proposed system
18	68.6	87.1	2.50	Proposed system
19	57.1	94.3	4.56	Proposed system
20	54.3	92.9	4.83	Proposed system
Best	78.6	95.7		
Mean	64.9	89.6		
s.d.	7.0	3.1		

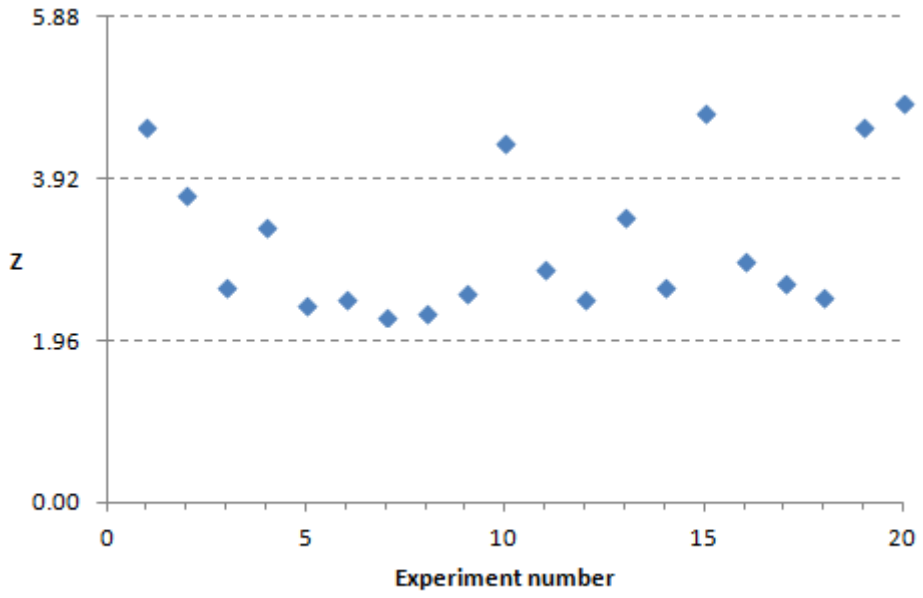


Figure 3.20: Illustration on Z scores of the experiment on mandarin peel inspection

processes and 70% for testing.

The experimental results are shown in Table 3.6. It is apparent that the modified system performed enormously better than the original. The modified classifier was able to reach 93.9% mean accuracy while the original achieved only 46.1%. All accuracy results generated by the adapted program are between 37–53% superior to those produced by the original. The Z values shown in Table 3.6 and Figure 3.21 are between 5.2 and 6.4, dramatically higher than the critical value. It demonstrates that the performance differences are significant, and this experiment was also able to take advantage of the additional features.

3.6.5 Experimental Conclusions

This chapter mainly aims to verify object classification performance of the system improved by additional colour and texture feature extraction. The verification was

Table 3.6: Classification results and Z scores of the experiment on apple variety identification

Experiment number	Classification accuracy results (%)		Z score	Superior performance
	Original system	Modified system		
1	41.5	95.1	6.34	Proposed system
2	45.1	93.9	5.88	Proposed system
3	45.1	93.9	5.88	Proposed system
4	46.3	96.3	6.25	Proposed system
5	42.7	95.1	6.13	Proposed system
6	47.6	90.2	5.44	Proposed system
7	41.5	95.1	6.21	Proposed system
8	47.6	91.5	5.40	Proposed system
9	54.9	92.7	4.93	Proposed system
10	50.0	96.3	6.00	Proposed system
11	41.5	90.2	5.21	Proposed system
12	34.1	90.2	6.24	Proposed system
13	47.6	97.6	5.96	Proposed system
14	45.1	93.9	5.88	Proposed system
15	51.2	95.1	5.40	Proposed system
16	53.7	96.3	5.59	Proposed system
17	52.4	95.1	5.59	Proposed system
18	45.1	93.9	5.75	Proposed system
19	43.9	95.1	6.05	Proposed system
20	45.1	90.2	5.62	Proposed system
Best	54.9	97.6		
Mean	46.1	93.9		
s.d.	4.9	2.3		

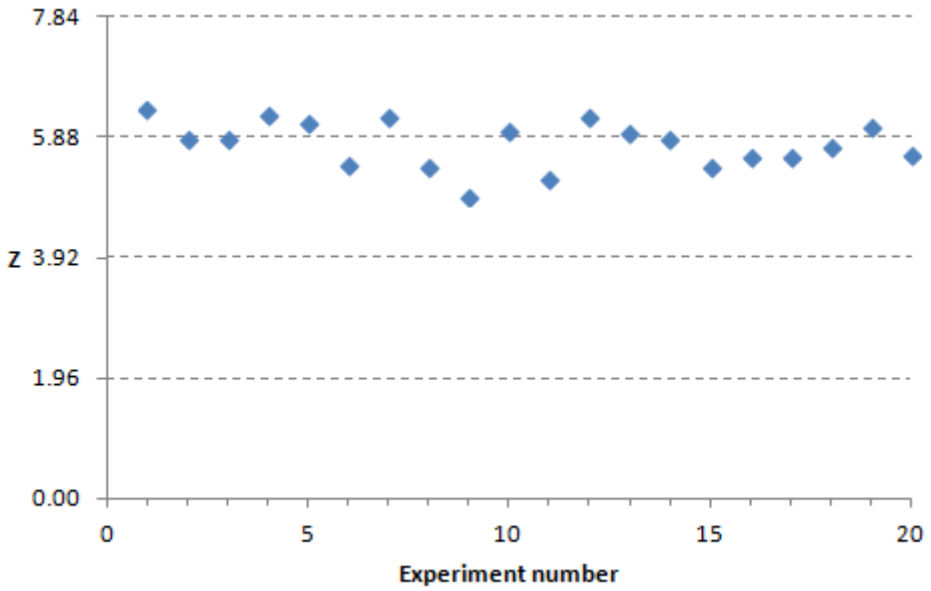


Figure 3.21: Illustration on Z scores of the experiment on apple variety classification

done on experiments in terms of a general type of object and agricultural produce, the focus of this research. The classification performance of the enhanced program was compared with the original Jasmine using McNemar's test.

For the first experiment, sticky index markers with the same shape and size but four different colours were employed. Although the original Jasmine contains extraction functions on grey-scale, it was found that additional colour and texture operators were able to improve the classification performance. Two example classifiers of the modified system are presented in Table 3.7. The adjusted program achieved perfect accuracy at all times, and the Z values are in the range 4.0–5.5. It demonstrates that the performance differences are obviously significant, and using the additional operations was sharply superior to performing by the original system.

The additional colour and texture features also enhanced the system achieve-

ment for experiments on agricultural produce grading. Example classifiers of the modified system of each experiment are shown in Table 3.7. The classification accuracy of the adapted program exceeded that of the original system in all sub-experiments. In terms of the Z score comparison, only one sub-experiment on persimmon inspection returned the Z value slightly less than the critical point. However, the other results indicate that the performance differences are significant. In the experiment on apple variety identification in particular, all Z scores are over 5, substantially higher than the critical value.

Using the critical value of 1.96 indicates that the algorithms are supposed to differ by chance only one time in twenty (0.05). All experiments demonstrated that achievement probability of the modified system is significantly superior to the original Jasmine. Furthermore, all experiments contained sufficient samples to prove that the sub-experiments presenting significant performance differences had large enough values of $N_{sf} + N_{fs}$ as McNemar's test requires. In other words, the total number of N_{sf} and N_{fs} were equal to or greater than 20 for all sub-experiments showing the reliability that the differences in performance of the proposed program and the original are significant. Therefore, they clearly demonstrate that the classification performance of the modified system with extended colour and texture properties was superior to the original Jasmine performance.

Table 3.7: Example classifiers of the modified system improved by the additional colour and texture operations

Experiment	Example classifiers
Sticky index markers	Classifier1, sub-classifier[Green]: l1l2l3Mean(GB)
	Classifier1, sub-classifier[Orange]: HueGLCM(correlation)
	Classifier2, sub-classifier[Green]: HueGLCM(mean)
	Classifier2, sub-classifier[Orange]: Hue(mean)
Persimmon bruises	Classifier1: (/ (cu GreenGLRM(shortRun)) (* HueGLRM(shortRun)
	(* (* HueGLRM(shortRun) IntensityGLCM(homogeneity)) BlueGLRM(runEntropy))))
	Classifier2: (+ (cos (hypot (cu HueGLRM(shortRun)) (min BlueGLRM(nonUniformity)
	GreenGLRM(runPercentage)))) (sq GreenGLRM(shortRun)))
Mandarin peel	Classifier1: (* IntensityGLCM(uniformity) (- HueGLRM(runPercentage)
	(> (sq (exp HueGLCM(entropy))) (cu (sq 50.800877149700284)))))
	Classifier2: (sqrt (* HueGLRM(longRun) IntensityGLCM(uniformity)))
Apple varieties	Classifier1, sub-classifier[GrannySmith]: (cos (+ Roundness() HueGLRM(shortRun)))
	Classifier1, sub-classifier[Cox]: NormalisedRed(mean)
	Classifier2, sub-classifier[GrannySmith]: (sq HueGLRM(runPercentage))
	Classifier2, sub-classifier[Cox]: NormalisedGreen(mean)

CHAPTER 4

3D RECONSTRUCTION AND ITS USE

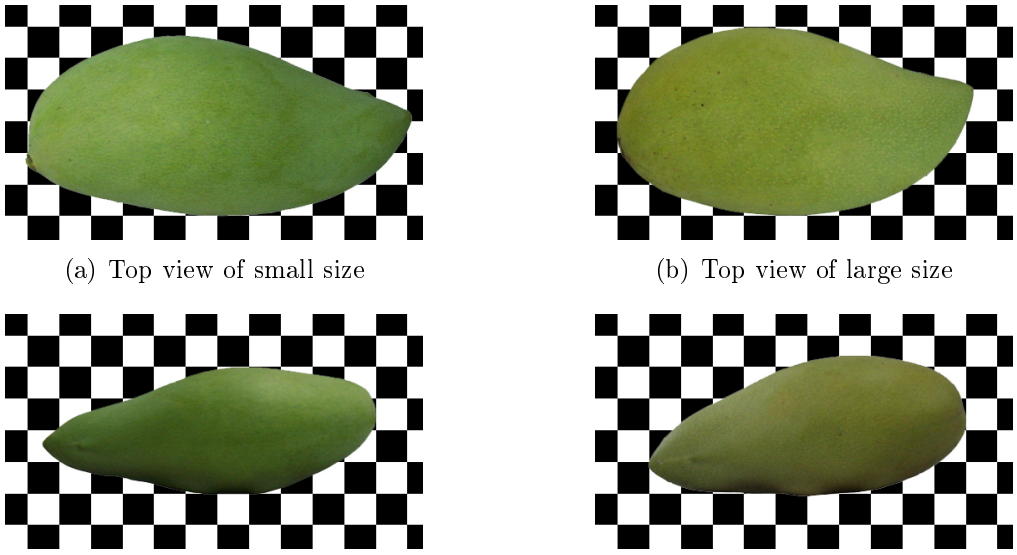
In terms of agricultural produce grading, most produce types do not have a simple geometric shapes, such as mangoes, rose apples and strawberries, so it is sometimes difficult to determine their size. Thus, the agricultural industry generally uses weighing to solve the difficulty (in case weight varies directly to size). As there are an enormous number of fruits, this is time consuming for human operators. Consequently, sorter machines have been developed to use instead of human labour; for example, Silanam *et al* [148] invented a sorter machine based on object weighing for mango size sorting.

However, not only size is a necessary property used for grading on most types of agricultural produce, but other characteristics (such as qualities, defects and the maturity) are also important for inspection. If a grading machine can inspect objects in several qualities like humans do, it will be more worthwhile. Various grading systems have been put forward, and object estimation in 3D plays an

important role in them.

As discussed in section 2.2, many researchers have tried to analyse object properties in 3D and employed them for object classification. For instance, Hahn and Sanchez [34] evaluated the growth and quality of carrots based on volume estimation. They presented two techniques to calculate 3D volume of carrots. Khojastehnazhand *et al* used two cameras to capture side views of objects and adapted sector calculation to estimate 3D volume for lemon [60] and orange [40] grading. Conversely, Rashidi and Gholami [32] calculated volume by using ellipsoid approximation, to determine kiwi size. Beyer *et al* [41] analysed cherry shape development based on contours in 3D. Chalidabhongse *et al* [61] extracted 3D shape properties, such as volume and surface area, and used them with other extracted 2D shape features for mango sorting. Their experiments indicated that using automatic classification with these extracted features resulted in more accuracy than grading by experienced humans.

For agricultural produce sorting based on vision, the projected area can be employed to estimate size; however, only one view of objects for some produce types would not be able to characterize the overall shape. For example, Figure 4.1(a) and 4.1(b) show top view pictures of sample mangoes with small size and large size respectively; however, the projected area of the small fruit is bigger than that of the large one. Therefore, they are probably grouped in the wrong size category if the system uses only projected area as the major feature for sorting. Figure 4.1(c) and 4.1(d) present their side view images. Apparently, the side view areas look significantly different in size, and the left fruit should be grouped as being of smaller size than the right one. For this type of agricultural produce, we do not know which viewpoint can best represent the size for the whole object. Consequently, it would be better to estimate their size from their overall shape.



(a) Top view of small size (b) Top view of large size

(c) Side view of small size (d) Side view of large size

Figure 4.1: Top view and side view images of small and large mangoes

(a) Top view (b) Side view 1 (c) Side view 2

Figure 4.2: Top view and side view images of a defective guava

Similarly, other physical properties employed to discriminate the maturity, qualities or varieties need to be evaluated from the whole appearance for most agricultural produce. Using only one viewpoint of an object is probably not sufficient to estimate the entire appearance. For instance, Figure 4.2 shows multiple view images of a defective guava: the top view in Figure 4.2(a) suggests good quality but skin defects can be seen in the other views as shown in Figure 4.2(b) and 4.2(c).

Some researchers do not aim to analyse objects in 3D; however, they still need to set up their systems to retrieve pictures containing the object appearance as much as they can. For example, Li *et al* [50] equipped the machine with two cameras and two mirrors in order to gain different views of an object. Grading machines built by Xiaobo *et al* [48] and Leemans and Destain [51] were designed to be able to rotate an object so as to capture multiple view images. Khojastehnazhand *et al* [60] developed a machine vision system containing two cameras to gain two different side views of an object.

Clearly, using 3D characteristics and multiple views of objects has often been employed for agricultural produce grading. Since a computer vision approach based on 3D object analysis is able to cope with this task, this chapter applies vision techniques to evaluate various physical properties for agricultural produce inspection in order to develop a grading technique that performs as well as possible, and potentially as well as humans do.

One goal of computer vision is to generate a 3D model by recovering information from 2D images. The recovery requires knowledge about material in the scene and projection geometry. As images are projected from objects of the real world from 3D to 2D, 3D information cannot be completely preserved in an image. Accordingly, every content cannot be recovered precisely to create a model.

The human visual system takes advantage of two eyes to predict depth information. Binocular or stereo vision is a theory of the human visual system — seeing with two eyes is capable of perception on the content of a scene. The stereo vision concept supports human vision abilities so as to examine the relative distance of objects seen from two viewpoints. Even though humans are able to see with only one eye, the eye-brain system can make use of prior knowledge about material and the natural world to recognize shape and size features. For example, humans are familiar with the physical properties of vehicles, although there are many types of them. As humans see them in everyday life, they probably can estimate the physical properties of vehicles by a monocular view. Furthermore, they may evaluate their size by comparing them with other surrounding objects.

It is known that computer vision aims to imitate the abilities of the human visual system. Seeing from one viewpoint by computer vision is able to estimate 2D properties of objects, such as width, height and area; however, it is impossible to determine 3D properties, such as depth, shape and volume. Therefore, more than one point of view needs to be employed for 3D analysis.

The principles of 3D reconstruction are described in the following section. Subsequently, reviews on previous techniques are presented. Proposed ideas to evaluate voxel colour and extract features are shown in the rest of the chapter.

4.1 2D-3D Coordinate Transformations

The transformation between object coordinates in 2D and 3D can be thought of as involving five distinct coordinate frames. The 2D-3D coordinate transformation details are explained in [149, 150], and briefly summarized below.

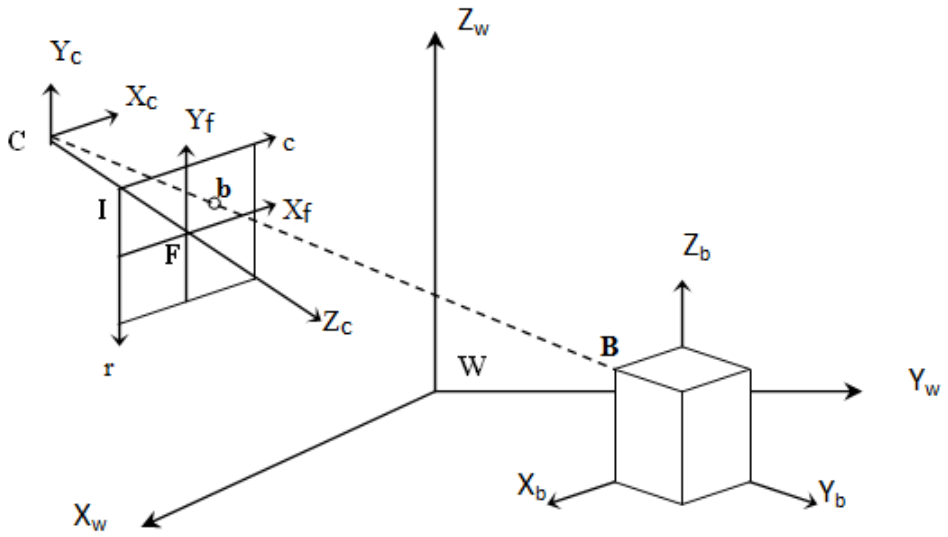


Figure 4.3: Point B at the object projected to the image plane at point b

4.1.1 Object coordinate frame

An object coordinate frame is a 3D coordinate system. It is generally used in terms of computer graphics and computer vision. In Figure 4.3, the X_b, Y_b and Z_b axes present the object coordinates. This frame is used to analyse an object straightforwardly. The axes are the object's own axes because they can be considered as being fixed on the object to be used to refer to a position on it. The coordinates of any positions do not change even when the object is translated and rotated. For example, the coordinates of spot B at the corner of the object are always referred to as (X_b, 0, Z_b) in the object coordinate frame. If the object is translated or rotated, the spot B remains (X_b, 0, Z_b).

4.1.2 World coordinate frame

A world coordinate frame shows in a 3D view. In Figure 4.3, coordinates in this frame are represented by the X_w, Y_w and Z_w axes, and point W is the origin of

this frame. This representation is often used to identify an object position before and after moving it or to estimate the distance between objects.

4.1.3 Camera coordinate frame

A camera coordinate frame is considered in a 3D space. It aims to present an object's location relative to the camera position. In Figure 4.3, the X_c, Y_c and Z_c axes illustrate the camera coordinate frame, and point C is the origin or camera centre.

4.1.4 Image plane coordinate frame

An image plane coordinate frame is a 2D coordinate system. It is employed to represent a projected object in the camera, and coordinates of the image are specified as real numbers. In Figure 4.3, the image plane is spanned by the X_f and Y_f axes. The X_f axis points to the right, and the Y_f axis points up. The spot B on the object is projected onto the image plane as point b.

4.1.5 Pixel coordinate frame

A pixel coordinate frame is a 2D array of pixels implemented in computer programs. It is obtained from an image plane, and it describes object pixels in terms of rows and columns. In Figure 4.4, this frame is represented by the r and c axes. A pixel position is specified by row and column indices identified by integer numbers. Pixel [0,0] is conventionally located at the top left corner of the image array. Point b on the image plane is referred to as pixel a[i,j] on the image array at row i and column j. Note that the indices i and j of a pixel in an image array are referred differently from the coordinate (x,y) in an image plane as their axes

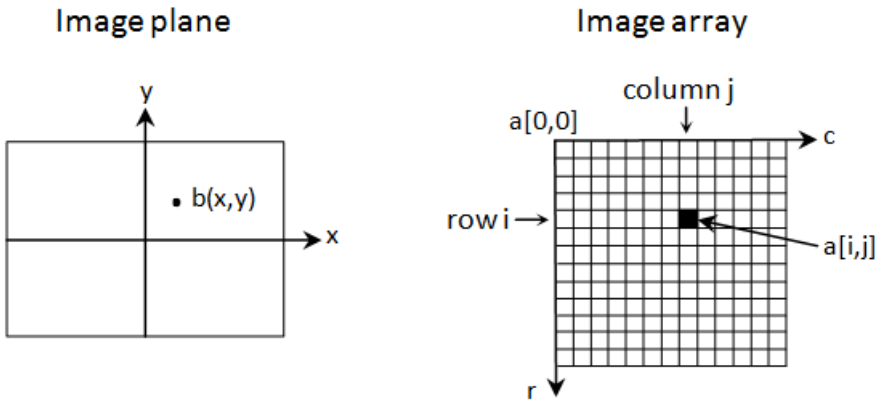


Figure 4.4: Relationship of point b between in the image plane and image array

are swapped.

In order to transform from object coordinates in three dimensions to image pixels in two dimensions, the five coordinate frames mentioned are implicated in this process; furthermore, extrinsic and intrinsic camera parameters are involved in the transformation. Extrinsic camera parameters are employed to transform between world and camera coordinate systems; intrinsic camera parameters are used to deal with coordinate translation on camera, image plane and pixel coordinate frames. The camera parameters will be described in detail in section 4.4. The sequence of coordinate frame transformations and relevant parameters are presented in Figure 4.5.

4.2 Image Projection

Image projection projects an object in a world space to a 2D image plane. This results in reduction of 3D properties to 2D characteristics of an object. In other words, 3D shape features generally include width, length and thickness, and they are usually represented in X, Y and Z axes. After transforming to a 2D image,

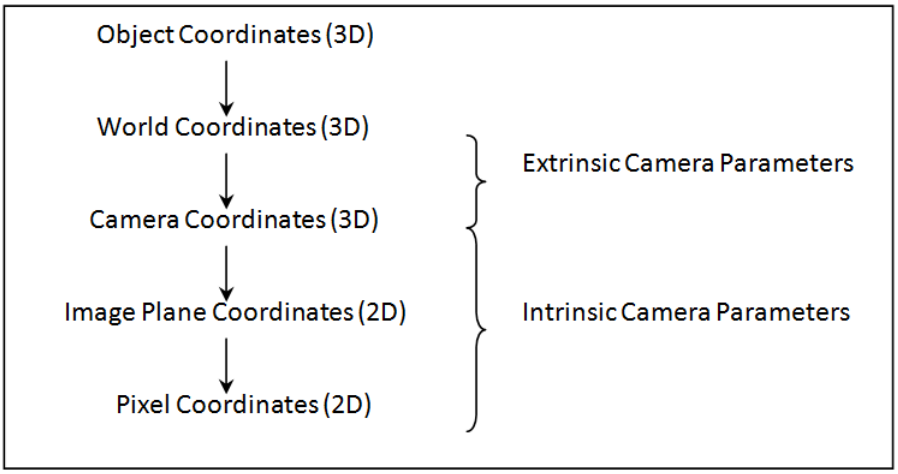


Figure 4.5: Transformation sequence of the coordinate frames and relevant camera parameters

the object properties in a view of the Z axis will be discarded. Image projection can be instantiated in two different methods as follows.

4.2.1 Orthographic projection

An idea of orthographic projection is based on viewing from a very long distant point or the focal length (f) approaching infinity ($f \to \infty$). It implies that the value of $\frac{f}{Z}$ is close to 1. Rays of light from an object project parallelly to an image plane as shown in Figure 4.6. The lines are also known as lines of projection. The size of a projected object is approximately equal to the real object. Moreover, when the object moves forwards and backwards, it does not impact on the size of the projected object on the image plane. In other words, although the position of the object changes on the Z axis, the image remains stable on the X and Y axes. Therefore, the size of the image is roughly equal to that of the object placed at a long distance from the camera.

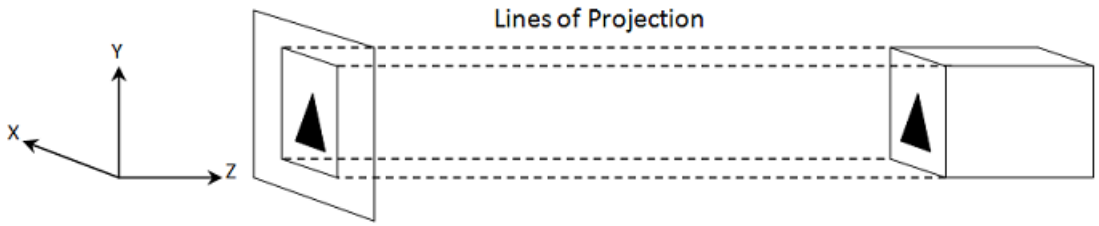


Figure 4.6: Orthographic projection

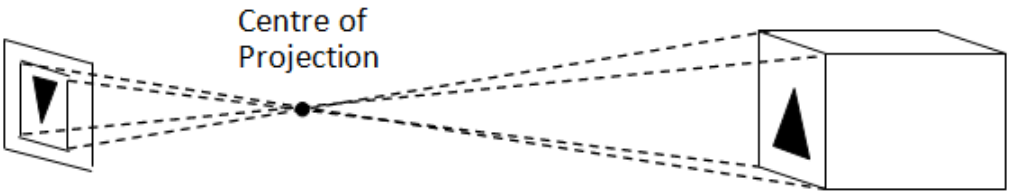


Figure 4.7: Perspective projection

4.2.2 Perspective projection

Perspective projection is demonstrated as general photograph taking. The line of sight of perspective projection projecting to an image plane is different from the orthographic projection. It passes through the centre of projection (or camera centre) and falls on an image plane behind (usually) the centre of projection. Consequently, the projected object is inverted on the image as presented in Figure 4.7. The image size depends on the distance between the object and the camera; the size of a projected image is bigger if the object is closer to the camera. Commonly, when an object is projected to an image plane by perspective projection, many real details disappear.

A camera involves perspective projection. To avoid the inversion, it is common to place the image plane in front of the centre of projection as shown in Figure 4.8, which illustrates where the projection of a point in the scene is located in the image plane. Point C is a camera centre that is the origin of the camera

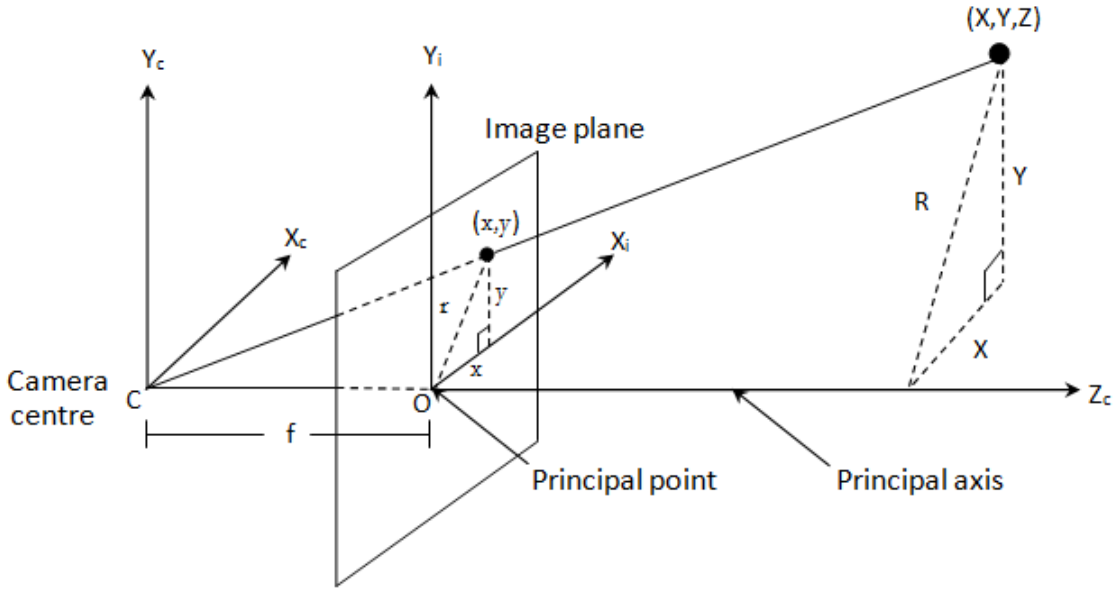


Figure 4.8: Perspective projection of a point

coordinate frame. The image plane is parallel to the X_c and Y_c axes and spanned by the X_i and Y_i axes to represent 2D coordinates of the position of a projected object. The axis passing from the camera centre through the image plane is called the principal axis or optical axis. Point O, the principal point, is the intersection point between the optical axis and the image plane. Note that the principal point is not always on the centre of the image plane. The distance between the centre of projection and the principal point is called focal length (f).

Figure 4.8 shows a perspective projection of point (X,Y,Z) in the world coordinate system projected on the image plane at point (x,y). According to the similar triangles theory, it results in $\frac{f}{Z} = \frac{r}{R}$ and leads to $\frac{x}{X} = \frac{y}{Y} = \frac{r}{R}$. The coordinates of the point (x,y) on the image plane projected from the point (X,Y,Z) is calculated by (4.1)–(4.2). It can be noticed that any points lying on the same line of the

projection will be projected on the image plane at the same point.

$$x = \frac{Xf}{Z} \tag{4.1}$$

$$y = \frac{Yf}{Z} \tag{4.2}$$

4.3 Camera Setup

This section moves on to present ideas in order to equip a camera or cameras for image acquisition. The number of cameras and camera placement are an important factor for acquiring a complete image of an object. If there are more viewpoints, the reconstructed shape will be closer to the real object. Moreover, a technique used for object reconstruction is the main determinant of setting a camera.

On the one hand, some researchers used only one camera to capture an object on a turntable [151–153]. A camera is placed higher than an object position, and it looks down on the object in order to capture (foreshortened) top and side areas of the object simultaneously. Each image is taken when the table is turned a few degrees. A sample model of this camera setup is illustrated in Figure 4.9.

Conversely, a set of cameras can be placed or hung surrounding an object in order to take photos of an inanimate object [61,154] or record a sequence of images of human motions [154–157]. Cameras are usually installed around a ceiling as presented in Figure 4.10.

In terms of agricultural produce grading, existing grading machine processes only one top view image [27, 28, 54, 55, 58] or a few multi-view images [43, 50,

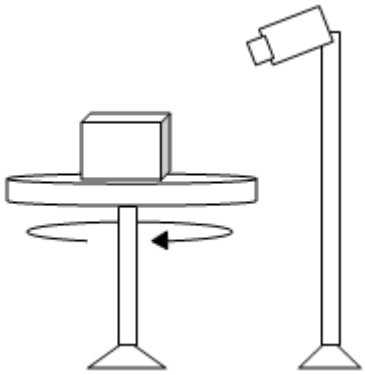


Figure 4.9: Sample model of one camera setup

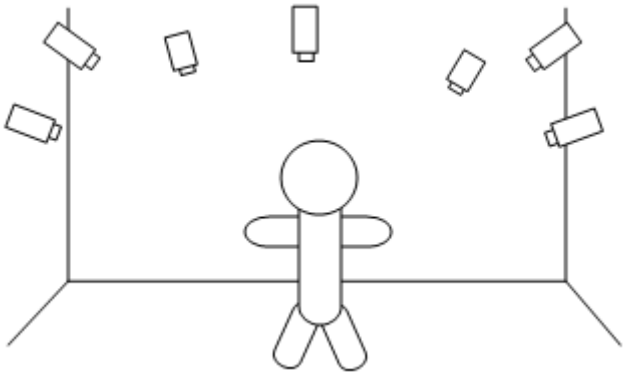


Figure 4.10: Sample model of multiple camera setup for human motion analysis

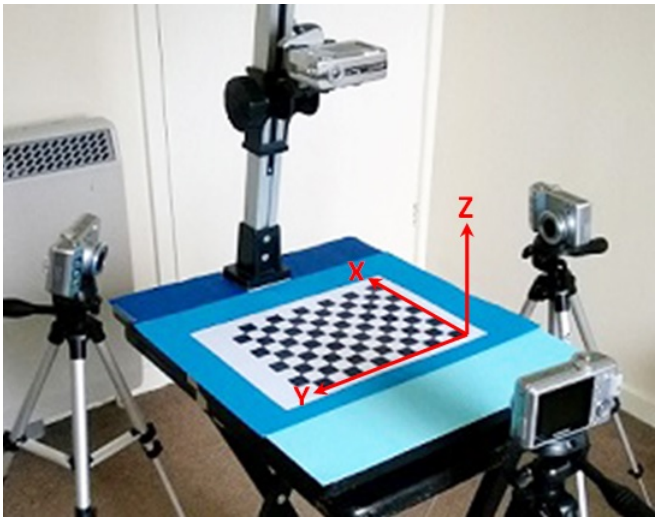


Figure 4.11: Camera setup for research experiments

(a) Viewpoint 1 (b) Viewpoint 2 (c) Viewpoint 3 (d) Viewpoint 4

Figure 4.12: Segmented images taken from different viewpoints

51, 60, 158] of an object. As this research aims to grade agricultural produce by employing the physical properties of reconstructed 3D objects, four digital cameras[1] lying around an object were used for image acquisition: one camera captures a top view and three cameras capture side views, as shown in Figure 4.11. The equipment set-up needed a camera to capture a top view in order to use it for 2D object analysis. For the side view capturing, three cameras were laid nearly in the corners of triangle: one facing against the X axis of the world coordinate frame, another one facing against the Y axis and the other facing opposite the origin. This would cover the whole appearance of an object. A planar checkerboard on the object base performed as a world coordinate reference, and it was used for camera calibration. Segmented images of a box with a different colour in each side, taken from the four different viewpoints, are shown in Figure 4.12.

4.4 Camera Calibration

Camera calibration is a technique to bring world and camera coordinate systems into coincidence. It aims to connect the locations of object points from the 3D scene to positions of pixels in the image array. Three steps are involved: firstly,

[1]Using digital cameras of Samsung S850

Figure 4.13: Sample checkerboard images used for camera calibration

the origin of the world coordinate frame must be translated to the origin of the camera coordinate reference. Secondly, rotating the axes of the world coordinate frame is needed to align with the camera coordinate system. Finally, one must transform the object coordinates to the image plane.

The transformations involve camera parameters comprising of extrinsic and intrinsic camera parameters extracted by a camera calibration method. A checkerboard is usually used for this process. The size and corner positions of the squares can be usefully employed to evaluate properties, location and orientation of the camera. A checkerboard lying on an object base as shown in Figure 4.11 is calibrated to retrieve extrinsic camera parameters. Images of a checkerboard taken from different angles, as sampled in Figure 4.13, are calibrated to extract intrinsic camera parameters. Currently, there are several potential tools to deal with camera calibration. One of them that this research used was implemented by Bouguet [159]. Camera parameters and transformation formulae are explained in [160] and represented as follows.

4.4.1 Extrinsic camera parameters

Extrinsic camera parameters consist of the location and orientation of a camera referenced in a world coordinate frame. Commonly, camera and world coordinate systems would not be in the same position and direction. For example, in Figure 4.11 the world coordinate reference is set to lie on the checkerboard on the table,

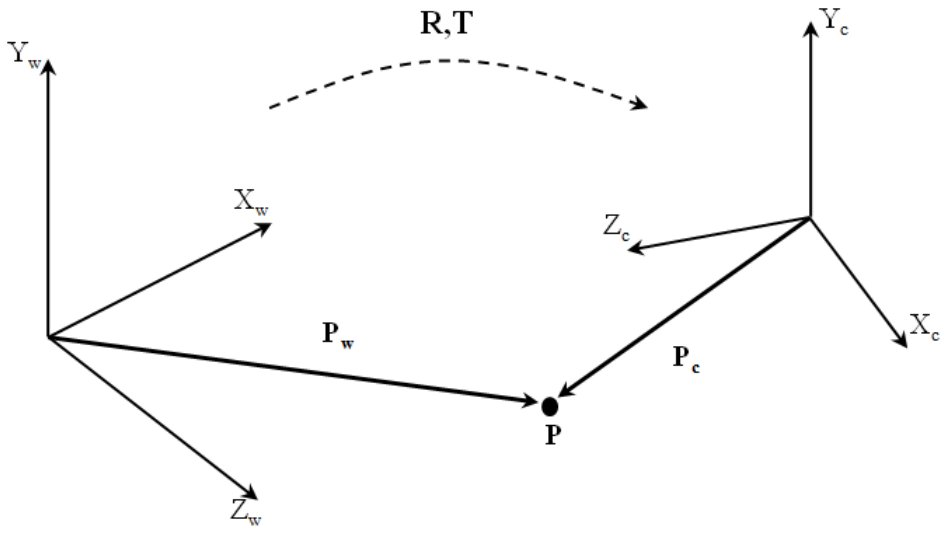


Figure 4.14: Relationship of point P in between the world and camera coordinate frames

and each camera has its own camera coordinate frame. All cameras lie in the same world coordinate frame. In order to analyse an object from each camera, the world and the camera axes have to be aligned coincidently. The origin of the world coordinate frame is moved to lie in the same position of that of the camera coordinate system by a translation vector (T). Furthermore, the axes of the world coordinate reference are rotated to point in the same direction of the camera axes using a rotation matrix (R).

Figure 4.14 illustrates a relationship between camera and world coordinate frames of point P. Point P in the world coordinate frame ($P_w = (x_w, y_w, z_w)$) and in the camera coordinate frame ($P_c = (x_c, y_c, z_c)$) are related to the extrinsic camera parameters as presented in (4.3)–(4.4) [160].

$$P_c = RP_w + T \tag{4.3}$$

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

$$T = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}$$

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{4.4}$$

where P_c is coordinates of point P in the camera coordinate reference, P_w is coordinates of point P in the world coordinate system, R is a rotation matrix, and T is a translation vector.

4.4.2 Intrinsic camera parameters

Intrinsic camera parameters play a key role in transforming the coordinates of an object in a camera coordinate frame to pixel coordinates in an image plane. They comprise of a focal length, principal point coordinates, a skew factor and

distortion coefficients as follows.

Focal length

The focal length is the distance from a centre of projection to the principal point. As shown in Figure 4.8, f is a focal length. Unit translation is managed by a scale factor that consists of f_x and f_y. It needs unit translation because a CCD camera with n×m rectangular grids of photo-sensors is millimetre scaled; on the other hand, a pixel image is an array of N×M pixels. Generally, n≠N and m≠M that lead to different coordinates on the CCD and the image array for the same object point.

Principal point coordinates

As discussed earlier, the principal point is an intersection between the principal axis and the image plane. In Figure 4.8, point O is a principal point. A principal point is the origin of the image plane coordinate system, and it is not always in the centre of an image plane. It also performs as the centre of the image array. O_x and O_y are referred as principal point coordinates in a pixel unit.

Skew factor

A skew factor is the angle between the X and Y axes as presented in Figure 4.15. A skew coefficient (S) is zero if the angle equals 90°. In general, the skew factor value is equal to zero for most cameras.

Distortion coefficients

Some photographs show odd curvatures of straight lines that appear around the periphery of the photos. It is explicitly noticeable for pictures of a grid structure.

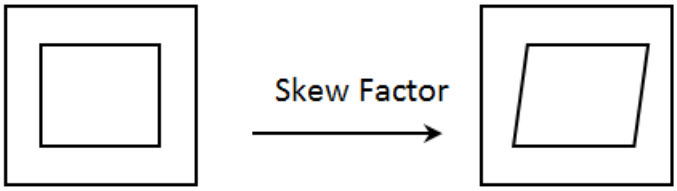


Figure 4.15: Skew effect

(a) Barrel distortion (b) Pincushion distortion

Figure 4.16: Barrel and pincushion distortions

It is caused by optical imperfections in the lens, and the problem is called distortion. It comprises of barrel and pincushion distortions as presented in Figure 4.16. For instance, a photograph of a tiled wall in Figure 4.17 shows barrel distortion. Image distortion coefficients (kc) include radial and tangential distortions [160].

4.4.3 Coordinate frame transformation formulae

In order to transform world coordinates to camera coordinates, values of x_c, y_c and z_c derived from calculation on extrinsic camera parameters as shown in (4.4) are involved. Intrinsic camera parameters generated by camera calibration (described in section 4.4) are employed to transform from camera coordinates to pixel

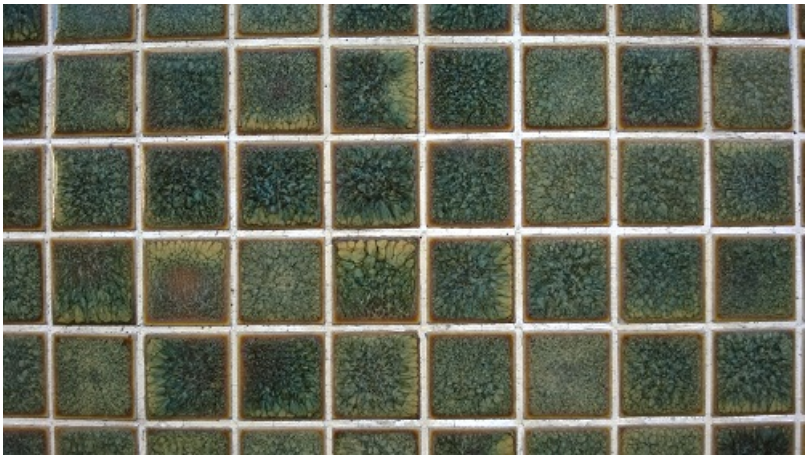


Figure 4.17: Photograph of a tiled wall showing barrel distortion

coordinates as shown in (4.5)–(4.9) [160].

$$x_n = \begin{bmatrix} x_c/z_c \\ \\ y_c/z_c \end{bmatrix} = \begin{bmatrix} x \\ \\ y \end{bmatrix} \tag{4.5}$$

$$r^2 = x^2 + y^2 \tag{4.6}$$

$$dx = \begin{bmatrix} 2kc_3xy + kc_4(r^2 + 2x^2) \\ \\ kc_3(r^2 + 2y^2) + 2kc_4xy \end{bmatrix} \tag{4.7}$$

$$\begin{bmatrix} x_{dx} \\ \\ y_{dy} \end{bmatrix} = (1 + kc_1r^2 + kc_2r^4 + kc_5r^6)\, x_n + dx \tag{4.8}$$

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & Sf_x & O_x \\ 0 & f_y & O_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{dx} \\ y_{dy} \\ 1 \end{bmatrix} \tag{4.9}$$

where x_n is normalized image projection, X_c, Y_c and Z_c are object coordinates in the camera coordinate frame, dx is tangential distortion, kc consists of radial and tangential distortion coefficients, f_x and f_y are focal lengths, O_x and O_y are principal point coordinates, S is a skew factor, and x_p and y_p are projected pixel coordinates on the image.

4.5 Previous 3D Reconstruction Techniques

A goal of volumetric reconstruction is to recover a 3D model by employing knowledge of the camera geometry from a set of object images. Various approaches have been developed to implement it in different contexts [161]. A well known technique is shape-from-silhouette, used widely for various tasks. The intersection of back-projection cones from camera centres passing through silhouette contours on image planes generate an approximation to 3D volume called the visual hull [162]. The visual hull is the maximal volume, and its shape will be closer to the real object if more images are employed.

Polyhedral and volumetric approaches can be used to manage the volume reconstruction [154]. For the polyhedral technique [163,164], silhouette contours on images are discretized into line segments resulting in a set of edges joining consecutive vertices of each image. The silhouette line segments are back-projected to the 3D space, where they are intersected with all other input silhouettes to

generate the resulting polyhedron, which that is able to be represented by a mesh representation.

For the volumetric approach, the visual hull is computed using a discrete 3D grid of volume elements (voxels). A large bounding box that is able to cover the whole object volume is initialized; then, it is divided into small voxels. Each voxel is projected onto each silhouette image. If the projection of a voxel falls inside all silhouettes, it will be kept in the volume; otherwise, it will be removed from the object space. This method is useful for volume measurement [61]. Due to the computational cost, an octree representation is used to speed up this procedure [154, 165].

Early tasks based on geometric modelling were able to reconstruct 3D structure precisely but could not present the surface information [166, 167]. Subsequently, various techniques have been put forward to provide better results. A photo-consistency idea was initiated and presented by Seitz and Dyer [151]. It was able to construct colour 3D models by measuring the colour consistency across many consecutive observations. One camera was employed and the camera setup as shown in Figure 4.9. A reconstructed object was formed by a set of voxels in a 3D space. First, a very large cube was initialized in the 3D space, and it was separated into small voxels equally. After that, the volume was traversed into a sequence of voxel layers. As the camera was placed in known position, the voxel layers were visited in depth-order. In other words, the closest layer of the camera was swept before the further layers. Occlusion-compatible order and marking pixels (footprint) [152] were employed to solve an occlusion issue. Projected pixels of a voxel were checked when the voxel was visited. If the projected pixels did not overlap any footprints of previously visited voxels, the projected pixels were marked as a footprint, and the voxel would be processed in a next step. If

the projected pixels overlapped previous footprints, it means that the voxel was occluded by other voxels, and the voxel would be discarded from a next process.

They also used a voxel consistency technique to decide whether to keep or delete a voxel. The technique evaluated the colour of the voxel from several images. If the colour was accepted from a colour consistency threshold, the voxel would be kept in the space and assigned the colour; otherwise, it would be deleted by being set transparent.

As the voxel colouring technique sweeps a single plane in the near-to-far direction related to the camera, it is unable to cope with arbitrary camera placements. Therefore, many techniques were put forward to solve the constraint. Space carving, proposed by Kutulakos and Seitz [168], is an initial idea to generate a 3D volume from multiple images taken at known but arbitrarily different viewpoints. Instead of sweeping a single plane, six directional sweeps including positive and negative directions of X, Y and Z axes, was applied. However, the concept of occlusion modelling (that is, visiting occluders before the voxels that they occluded) was still maintained. The idea of voxel consistency was also employed, but only a set of images in the sweeping scope were considered for the consistency checking in each sweep [169].

Generalized voxel colouring (GVC) presented by Culbertson and Malzbender [170] is a valid idea to eliminate restrictions on camera locations. Similarly to the space carving technique, initial voxels were set opaque, and the colour consistency of a group of images was used to delete inconsistent voxels. Furthermore, they proposed algorithms of GVC using item buffers (GVC-IB) and GVC using layered depth images (GVC-LDI) to manage voxel visibility. A concept of voxel visibility is matching each pixel of all images with the closest voxel projected to. For GVC-IB, an item buffer was constructed for each image to store each pixel and its closest

voxel. Meanwhile, GVC-LDI applied layered depth images to collect a list of all surface voxels projected to each pixel. The voxels were stored in order according to the distance from the camera. They found that the former needed less memory; on the other hand, the latter performed more efficiently to update the visibility information.

Embedded voxel colouring, presented by Leung and Lovell [171], is a potential technique to reconstruct 3D models from arbitrary camera locations. They defined a visibility relationship between voxels and pixels that a voxel was visible to a pixel if the voxel was projected to the pixel, and it was the closet voxel from the pixel; furthermore, a voxel was occluded with respect to a pixel if there was no clear line of sight between them. They also suggested using a low resolution volume would encounter the 'leaking projection' problem. An example of the leaking projection issue is illustrated in Figure 4.18. Voxel A, B and C are projected to the image from the viewpoint. Each voxel is represented by its centre. Voxel A is projected to pixel x, and pixel z is projected from voxel B. Voxel C can be also projected to the image. Its projected point is in the different position on the image from those projected by the voxel A and B. However, the voxel C is actually occluded by voxels A and B, so it should be invisible in the image by this viewpoint.

Accordingly, water-tightness was proposed to deal with voxel occlusion. Voxels were processed as volumes tessellating the object space, and image pixels acted as areas tessellating the image plane. Only surface voxels were employed to this process. They assumed that no ray could pass from an outside position of an object to an inside position without striking a surface voxel. When a voxel is projected to an image, it is important to take its projected pixels into account. The closest voxel belonging to the projected pixels occluded other, further voxels projecting to the pixels. Leung and Lovell also contributed monotonic carving order to arrange

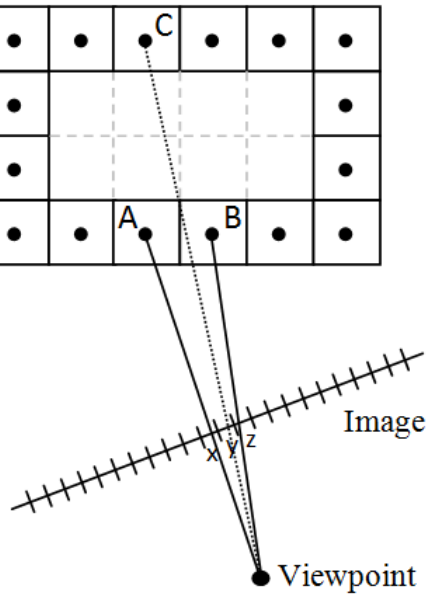


Figure 4.18: Illustration on leaking projection

the order of voxel removal. As voxel carving was based on colour consistency, a global threshold named the posteriori threshold selection was proposed. In addition, causality of carving was shown to emphasize the performance of carving by the consistency threshold idea.

Afterwards, some research [172,173] was reported that measures and compares the capability of existing colour consistency checks. Recently, Leung *et al* [174] reinforced the embedded voxel colouring framework with an adaptive thresholding algorithm to obtain more accurate results. As the mentioned research aimed to generate high quality results of a 3D model, many pictures were employed: 17 and 36 images for [174], 15 images for [175] and 21 images for [152]. A technique based on photo-consistency was appropriate to perform.

A huge amount of research has been developed to reconstruct a 3D model

from a moving human, often trying to display results in real-time. Some of them generated a 3D model based on the voxel-based construction. For example, an idea was proposed by Balan [176]: after background subtraction, each initial voxel in the 3D space was projected to all images. Because of using a low resolution of voxels, a projected region or a convex hull formed by the projected eight vertices of a voxel was considered for voxel carving. If a proportion of the convex hull fell outside a silhouette, the voxel would be deleted. While Cheung [156] presented a sparse pixel occupancy test (SPOT) to check a convex hull. Uniformly distributed pixels of the convex hull were selected and applied. If they overlapped silhouette pixels, the voxel was assigned to the object volume. The remaining voxels in the space acted as a reconstructed 3D object. The next process was implemented only on surface voxels, identified by considering six-connected neighbours. If one or more nearest neighbours in six directions of a voxel were empty, the voxel was considered as a surface voxel. As multiple voxels probably projected to the same pixel, the closest voxel of the image should belong to the pixel colour. Balan [176] and Kehl *et al* [157] employed depth buffers to detect occlusions, and the colour information was assigned to the corresponding closest surface voxels.

4.6 3D Shape Reconstruction and Measurements

This thesis employs four cameras for image capture: one camera for the top view and three cameras for the side views, as shown in Figure 4.11. Some object areas may be visible in multiple images; on the other hand, some regions may be seen in only one image. For example, in Figure 4.12 the red side appears in all images; in contrast, most areas of the other sides display in only one image. Therefore, the photo-consistency check technique is not involved in this research.

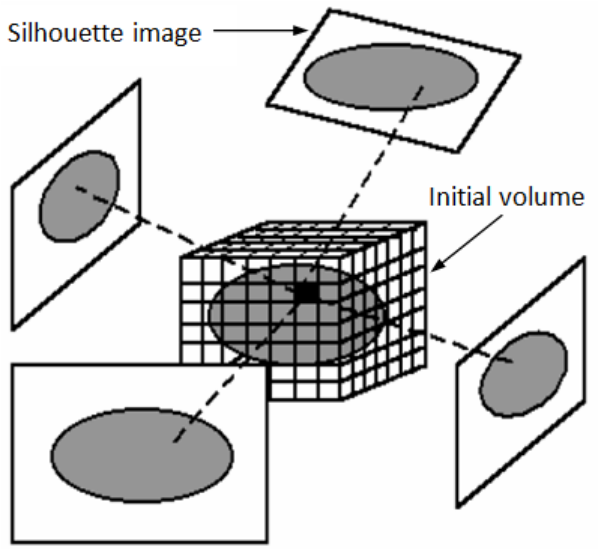


Figure 4.19: 3D volume reconstruction based on a basic space carving algorithm

In terms of development for agricultural applications, any number of cameras may be installed in any positions. Therefore, this research will discard any restriction on camera locations so that images can be taken at arbitrarily-distributed camera viewpoints. This research not only aims to construct 3D models but also measures 3D shape properties in order to employ them for object inspection. Accordingly, the voxel-based procedure is adopted for this research.

4.6.1 Approach used for voxel carving

A technique used for shape reconstruction in this thesis is based on the basic space carving algorithm presented in [155]. The generated 3D volume is represented by voxels as illustrated in Figure 4.19. Algorithm 1 presents the detailed process of space carving. A large bounding box is modelled as the initial volume enclosing the 3D object; then, the whole virtual volume is divided into equally sized voxels. All voxels are set opaque to be kept in the 3D space. Then the voxels are projected onto all silhouette images by using the corresponding camera parameters. If the

projected voxel falls outside a silhouette, it will be discarded from the volume by setting it to be transparent. The remaining voxels are an approximate object volume, roughly equal to or more than the real object volume. The generated voxels are then employed to extract 3D shape features.

Algorithm 1: Space carving algorithm

```
1  voxel initialization;
2  foreach v ϵ {voxels} do
3      v = 'opaque';
4      foreach I ϵ {images} do
5          project v to image I;
6          if the projected point falls outside a silhouette then
7              v = 'transparent';
8              goto ENDINNERLOOP
9          end
10     end
11     ENDINNERLOOP:
12 end
```

where {voxel} is a set of voxels, {image} is a set of image planes, 'opaque' is a value set as an object voxel, and 'transparent' is a value set as not an object voxel.

To project a voxel on an image plane, the equations described in section 4.4 are applied. A set of parameters obtained by camera calibration is used for a voxel projection consisting of focal lengths (f_x and f_y), principal point coordinates (O_x and O_y), distortion coefficients (kc_1, kc_2, kc_3, kc_4 and kc_5), a translation vector (T) and a rotation matrix (R). This research assumes that the skew factor is zero.

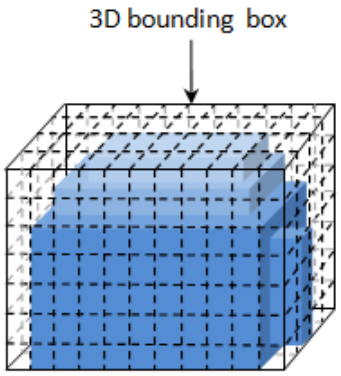


Figure 4.20: Illustration on a transparent 3D bounding box

4.6.2 Proposed techniques for shape extraction

Object grading by using 3D characteristics is an interesting issue for this thesis. It should give better results than using only 2D features. In the original Jasmine, morphological features were applied for object classification, extracted as 2D properties. However, if 3D characteristics are useful, extraction functions should compute them from reconstructed 3D objects. Consequently, the modified program retained the original morphological extraction functions for 2D analysis, and some 2D shape operations were adapted for 3D object measurement described as follows. As a result, the modified system contains 11 additional shape extraction functions for 3D analysis.

Bounding volume

A 3D bounding box is formed like a real box with width, length and height. After object voxels are reconstructed, the maximum and minimum positions of the voxels in X, Y and Z axes are defined as boundaries of a 3D bounding box. The number of voxels of the box is returned as the value of bounding volume. An example of a 3D bounding box is shown in Figure 4.20.

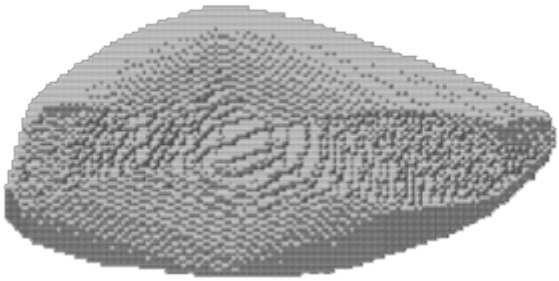


Figure 4.21: Reconstructed voxels of a mango

Number of voxels

For a 2D shape, the number of object pixels is simply used to represent pixel area. For a 3D analysis, the number of reconstructed object voxels is counted and returned as an output value. An example of assembled voxels of an object is sampled in Figure 4.21. This operation was used in [61] to deal with mango sorting.

Cuboid

The cuboid feature aims to evaluate object shape that is similar to a 2D rectangularity measurement but in 3D. It estimates how well the object fits into its 3D bounding box. It is the ratio of the number of object voxels to the bounding volume of the bounding box:

$$Cuboid = \frac{V_o}{V_b} \tag{4.10}$$

where V_o is the number of object voxels and V_b is the bounding volume.

Sphereness

After a 3D shape is reconstructed, surface voxels are employed to estimate the sphereness. The distance from every surface voxel to the centre of the shape is measured. The standard deviation of them is calculated and returned for this function.

Aspect ratio

The width, length and height of the 3D bounding box are measured. The aspect ratio function returns the three ratios:

$$Aspect\,ratio_1 = \frac{Width}{Length} \tag{4.11}$$

$$Aspect\,ratio_2 = \frac{Width}{Height} \tag{4.12}$$

$$Aspect\,ratio_3 = \frac{Length}{Height} \tag{4.13}$$

where $Width$, $Length$ and $Height$ represent the width, length and height of the 3D bounding box respectively.

Balance

The balance function also returns three terms. The centre coordinates of 3D object shape and 3D bounding box are calculated. The distance of them in each axis is compared with its bounding box size as shown in (4.14)–(4.16).

$$Balance_X = \frac{X_s - X_b}{Width} \tag{4.14}$$

$$Balance_Y = \frac{Y_s - Y_b}{Length} \tag{4.15}$$

$$Balance_Z = \frac{Z_s - Z_b}{Height} \tag{4.16}$$

where X_s, Y_s and Z_s are the coordinates of the 3D object shape centre, X_b, Y_b and Z_b are the coordinates of the 3D bounding box centre, $Width$, $Length$ and $Height$ are the width, length and height of the 3D bounding box respectively.

4.7 Proposed Technique for Voxel Colouring

After a 3D volume is generated, the next step is to assign colour to voxels. Only surface voxels will be considered in this step. Surface voxels can be defined using a six-connected neighbours technique. Internal voxels are completely surrounded by neighbours in six directions (the top, the bottom and four around side positions). Therefore, a voxel that does not have all six neighbours is set as a surface voxel. Retrieving voxel colour follows the idea that the closest voxel belongs to the pixel colour of the image, and the concepts of footprint [152] and water-tight models [171] are applied to manage the voxel occlusion problem.

The resulting combined technique, which is believed to be novel, is called **V**oxel **C**olouring based on a **F**ew cameras in **A**rbitrary positions (VCFA), and is proposed to evaluate a set of the closest surface voxels of each image and assign colour to the voxels. This idea not only estimates the colour of a reconstructed object but also paves the way for colour and texture extraction, described in the next section. Algorithm 2 presents the detailed process of the VCFA technique.

Algorithm 2: VCFA algorithm

```
1  foreach I ∈ {images} do                          /* iterate through images */
2      foreach v ∈ {surface voxels} do I[v] = dist(I, v) ;   /* iterate
       through surface voxels to measure the distance between the
       image and voxel */
3      ascendingSort(I[v]);   /* sort the voxels in ascending order */
4      foreach v ∈ {I[v]} do  /* iterate through the sorted voxels */
5          if the footprint area does not completely overlap the previous ones
           then
6              closestVoxel[I] = closestVoxel[I] ∪ v;   /* store the voxel in
               a set of the closest voxels of the image */
7              mark the footprint area;   /* mark projected pixels of the
               voxel */
8              colour[I][v] = mean(colour[footprint area]);   /* set mean
               colour of pixels in the footprint area to the voxel */
9          end
10     end
11 end
12 foreach v ∈ {surface voxels seen in multiple images} do   /* iterate
   through surface voxels seen in multiple images */
13     if v is visible in all images then   /* set colour of the top view
       image to the voxel */
14         colour[v] = colour of the top view image;
15     else   /* set colour of a corresponding image to the voxel */
16         colour[v] = colour of a corresponding image;
17     end
18 end
```

where $dist(I, v)$ is the distance between the voxel v and the image I, *ascending-Sort(I(v))* sorts the set of voxels of the viewpoint in ascending order according to the distance, *mean*(colour[footprint area]) calculates the mean value of pixel colour in the footprint area, and closestVoxel[I] is a set of the closest voxels of the image.

The algorithm will be described in two phases. The first part consists of line 1–11, and the rest will be detailed in the second part. In order to visit voxels in near-to-far ordering, the distance between surface voxels and images is measured, and it is the value of z_c as shown in (4.4) . Line 1–2 for each image I and each surface voxel v, the distance is calculated between the voxel and the image. In line 3, the voxels are sorted in ascending order according to distance. In lines 4–6, the voxels are projected to the image in order. If the bounding box of the voxel footprint completely overlaps previous ones, it indicates that the voxel is occluded. If not, the voxel is assigned into a set of the closest voxels of the image I. In line 7, the footprint area of the voxel is marked. In line 8, the mean colour of pixels in the footprint area of the image I is assigned to the voxel v.

After finishing this phase, each image will have its own set of the closest surface voxels already assigned a colour. It will also be known which voxels are seen in more than one image because they are assigned multiple colour values. An example of the results produced by the first part of VCFA is presented in Figure 4.23. They show estimated the colour of voxels seen in each viewpoint.

As some surface voxels can be visible in more than one image, the mean [175, 177] or median [176] of the colour set of related pixels in the corresponding images is assigned to the voxels. However, many factors, such as light effects, viewing angles, reflections and object material cause colour inconsistency between different images. Moreover, object areas seen in multiple images may have an effect on

colour and texture analysis. For instance, if duplicated areas present the defective colour of an object, the number of defective pixels would be more than it should be. For example, if the red voxels as shown in Figure 4.12 are defective, they are analysed in every image. Hence, it would be better if the duplicated areas are shown in only one image.

Accordingly, this research decided to select colour from only one viewpoint in order to set the voxels, reducing the colour error difficulty and alleviating the duplicated area problem. The question is which viewpoint is appropriate to assign to the voxels. Consequently, a couple of rules are made to choose a viewpoint for the voxels (in case of using camera setup similar to the research)[2].

Rule 1: if a voxel appears in all viewpoints, the top view will belong to the voxel.

Rule 2: for other cases, viewpoint labels of neighbours set with only one label in the defined scope are considered. The viewpoint assigned the most is assigned to the voxel.

Rule 2 is not only to select one viewpoint to a voxel but also able to reduce the projection error problem. An example can be illustrated in Figure 4.22. Numbers inside the voxels are referred to the assigned viewpoints. The orange and blue voxels are labelled by more than one viewpoint; therefore, they are considered by rule 2. It is correct that the orange voxel is labelled by viewpoint 2, but viewpoint 1 is probably assigned to it and the grey voxel because of the projection error. As the grey voxel is defined by only one viewpoint, so nothing can change. The viewpoint consideration goes on with the orange and blue voxels.

Assuming that neighbour size = 2, so viewpoint labels of neighbours in the scope with dashed borders are taken into account. Although the edge voxels la-

[2]For other styles of camera setup, using only rule 2 would be appropriate.

belled by viewpoint 2 and viewpoint 4 are in the defined scope, they are assigned by more than one label; thus, they are not involved in the consideration. Eighteen voxels in the scope are labelled by viewpoint 2, and one voxel is defined by viewpoint 1. Consequently, viewpoint 2 will be assigned to the orange voxel. For the blue voxels, this case commonly happens to edge voxels. They are considered by rule 2 similarly to the orange voxel; however, any suspected viewpoint can be assigned to them because it is supposed to be.

For instance, in the case of a reconstruction of the box with a different colour in each side, as shown in Figure 4.23, all red voxels are visible in all images, so the top view is set to them. It can be seen that some green edge voxels in viewpoint 3 are also presented in viewpoint 2, so they are considered by rule 2. If the other voxels are presented in only one image, they are not taken into account by the rules, and a related viewpoint is assigned to them.

These rules are applied in lines 12–18 of the VCFA algorithm as shown in Algorithm 2. Lines 12–14, for each surface voxel seen in multiple images, if the voxel is visible in all images, the related colour from the top view is assigned to the voxel. In lines 15–16, if the voxel is not able to be seen in all images, the viewpoint assigned the most in neighbour voxels that are set by one viewpoint in the defined scope belongs to the voxel. An example of final results generated from VCFA for the box reconstruction is shown in Figure 4.24. As can be seen, the duplicated areas are presented in only one image.

The bounding box area of a footprint, the 'footprint area', is illustrated in Figure 4.25. A footprint encloses the boundary of the projected vertices of a voxel, and the footprint area is an enclosing rectangle of the footprint. Figure 4.26 illustrates complete and partial footprint overlapping. As the blue voxel is closer to the viewpoint than the green one, it is visited first, and its footprint area

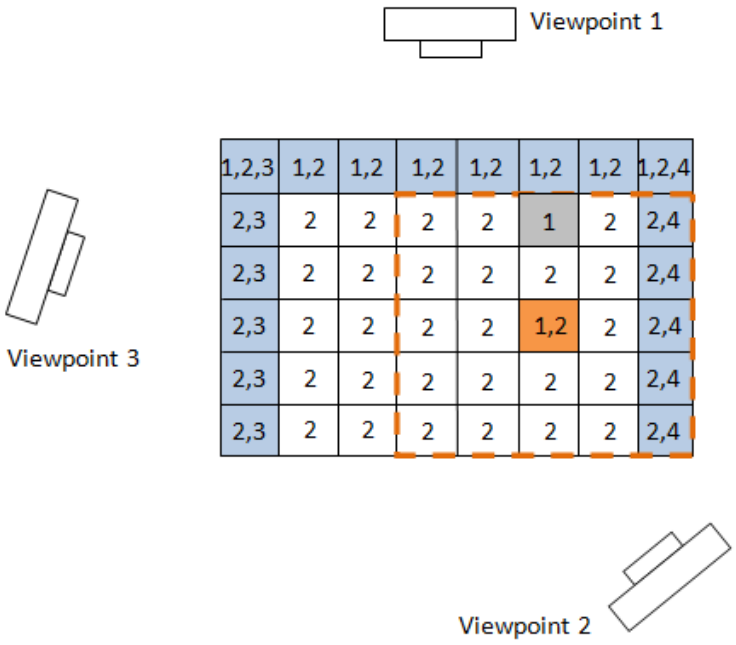
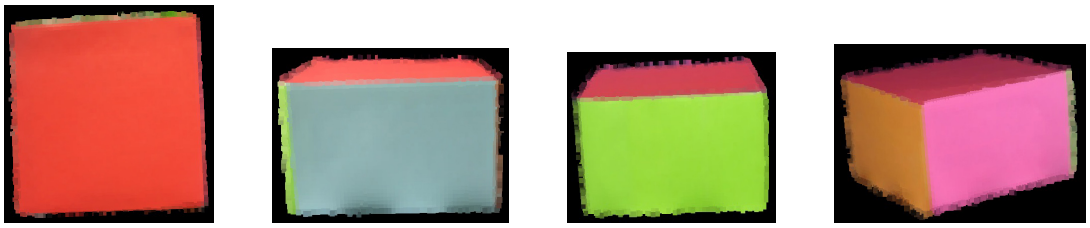
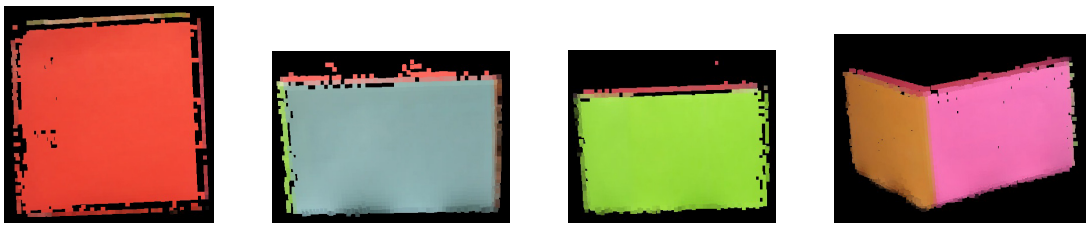


Figure 4.22: Illustration on defining a viewpoint to voxels assigned by more than one viewpoint labels



(a) Viewpoint 1 (b) Viewpoint 2 (c) Viewpoint 3 (d) Viewpoint 4

Figure 4.23: Voxel colour is generated in each viewpoint. The colour is calculated from mean colour of related pixels from the corresponding viewpoint.



(a) Viewpoint 1 (b) Viewpoint 2 (c) Viewpoint 3 (d) Viewpoint 4

Figure 4.24: Voxel colour is generated in each viewpoint. One voxel is shown in the only one viewpoint.

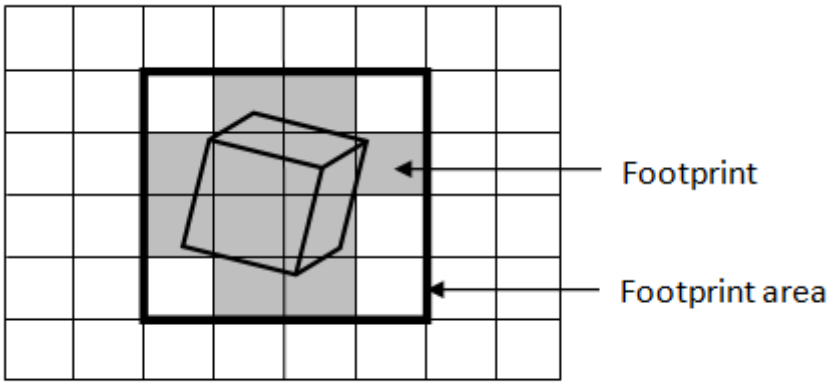
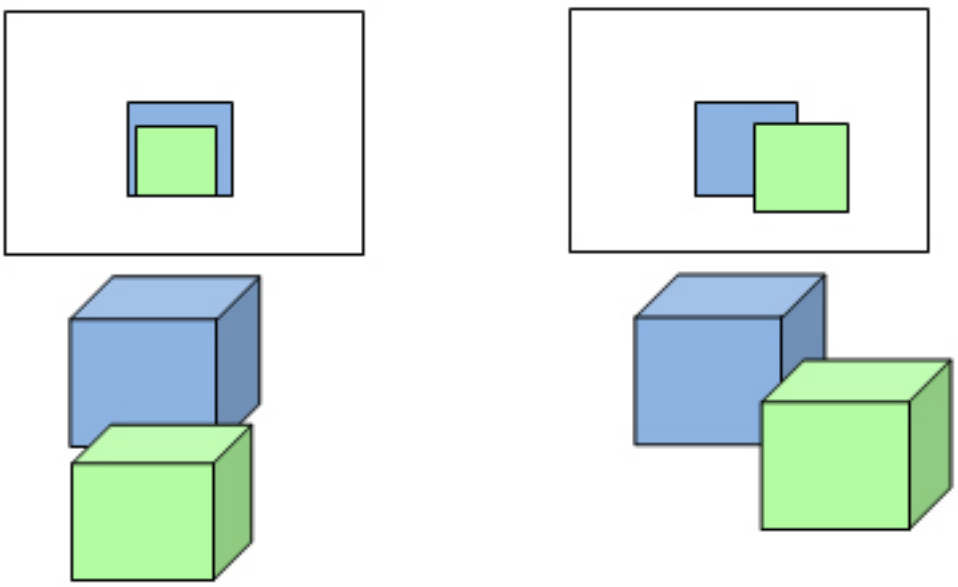


Figure 4.25: Footprint area of a projected voxel



(a) Complete footprint overlapping (b) Partial footprint overlapping

Figure 4.26: Illustration on a definition of footprint overlapping. For the research, the green voxel is occluded by the blue voxel in a case of complete footprint overlapping.

is marked. When the green voxel is visited, it can be seen that its footprint area completely overlaps the previous mark in Figure 4.26(a). It means that the green voxel is occluded; as a result, it will be discarded for the next process. In Figure 4.26(b), as the footprint area of the green voxel partially overlaps the previous mark, its footprint will be marked and it will be analysed for a next step.

The proposed approach was examined for agricultural produce using a defective guava. Segmented images of the sample and the results of the reconstructed sample present in each image are shown in Figure 4.27 and Figure 4.28 respectively. A sample 3D object was reconstructed with a voxel resolution of 44^3, and the VCFA method took approximately 10 seconds for voxel colouring (using images of resolution of 1024×768 and processing on Intel 2.27 GHz Core i5 with 8 GB of main memory under the Windows operating system). The proposed technique is appropriate for using a small number of cameras, and they can be set in any locations around an object. The proposed idea not only aims to construct a 3D model, but also pave the way to colour and texture feature extraction. The time taken for this is certainly longer than a system developed for real-time rendering. The higher the resolution is, the more accurate are the shape and colour obtained. Unavoidably, it will take more time to generate more refined 3D models [178].

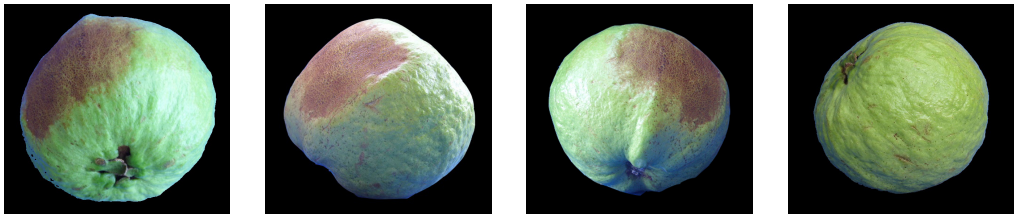
4.8 Proposed Techniques for Colour and Texture Extraction

After an object is reconstructed in 3D, and the voxel colours have been estimated, two techniques are proposed for colour and textural feature extraction. The first is termed Voxel Colour Analysis (VCA). As the colour of a surface voxel is as-

signed by the mean colour of its footprint area, so the colour values of all surface voxels are directly computed to generate colour features. For texture analysis, new images are generated by painting from the voxel colour without duplicated areas. Examples of a new image of each viewpoint are shown in Figure 4.29. They are used to extract texture properties.

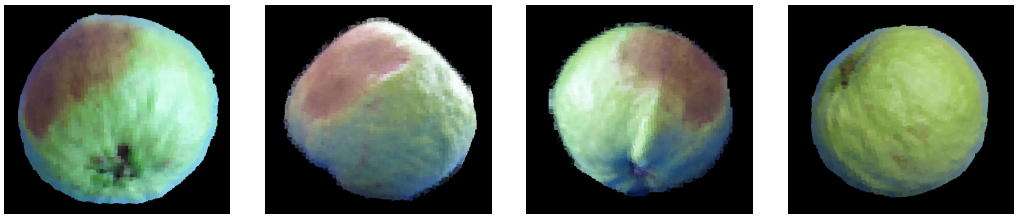
The second idea is termed Original Colour Analysis (OCA). This technique does not extract colour and texture characteristics from estimated voxel colour, but colour from the original images is employed instead. A new image of each viewpoint is generated, and object area of each new image is copied from the original image without duplicated object area. Newly-generated images are shown in Figure 4.30, and they are used to extract both colour and texture features.

Colour and texture extraction functions for 3D object analysis were developed similarly to the functions of 2D analysis. Colour operations extract features in binary, RGB and HSI components. Sample colour operations consist of normalization, mean, standard deviation, $c_1c_2c_3$, and $l_1l_2l_3$ as explained in section 2.1.3. The co-occurrence and run length methods were also used to extract textural characteristics for 3D object analysis. The texture operations work in RGB and HSI modes. For 3D property analysis, there are in total 163 feature extraction functions containing 11 shape, 29 colour and 123 texture operators. The VCA technique and 3D shape extraction took around 3 seconds while the OCA and 3D shape extraction took about 4 seconds to extract the entire features from a 3D reconstructed object (using image at a resolution of 1024×768 and processing on Intel 2.27 GHz Core i5 with 8 GB of main memory under the Windows operating system).



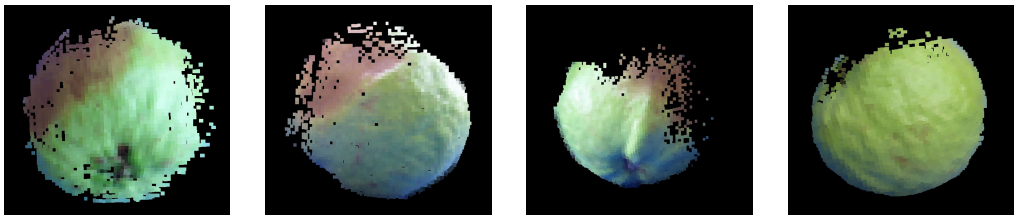
(a) Viewpoint 1 (b) Viewpoint 2 (c) Viewpoint 3 (d) Viewpoint 4

Figure 4.27: Segmented object images



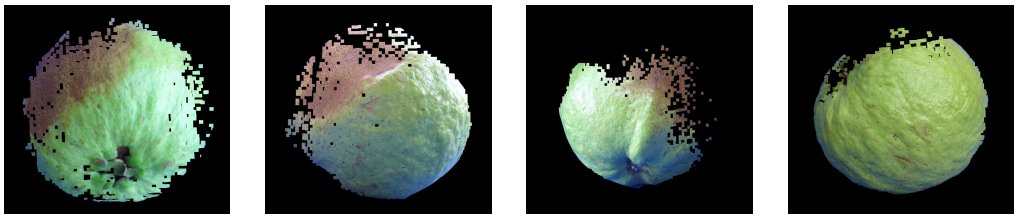
(a) Viewpoint 1 (b) Viewpoint 2 (c) Viewpoint 3 (d) Viewpoint 4

Figure 4.28: Voxel colour generated in each viewpoint



(a) Viewpoint 1 (b) Viewpoint 2 (c) Viewpoint 3 (d) Viewpoint 4

Figure 4.29: New images are painted by projected voxel colour, and each surface voxel is managed to be visible in one viewpoint.



(a) Viewpoint 1 (b) Viewpoint 2 (c) Viewpoint 3 (d) Viewpoint 4

Figure 4.30: New images are generated by the original object images with the scope of voxel footprint areas, and each surface voxel is managed to be visible in one viewpoint.

CHAPTER 5

EXPERIMENTAL VERIFICATION

In terms of agricultural produce sorting based on a vision technique, a projected area property can be employed to measure the size, but for some types of agricultural produce using only one viewpoint would not be able to present the overall shape. It would be sensible to estimate size from the overall shape. This is similar to physical property grading because only one side of objects would not be sufficient to represent the whole appearance. Therefore, input data analysed in 3D should result in more reliability than in 2D, and the following experiments were done to evaluate this hypothesis.

The experiments can be divided into two groups. The first one was done on rose apple sorting in order to compare the classification performance between using 2D and 3D shape features. The 2D shape features were extracted from top view images, and the 3D shape features were estimated from reconstructed 3D voxels. The second group included experiments on passion fruit, guava and

apple grading. These were done to estimate the possibility of using 3D properties extracted by the proposed techniques.

The classification accuracy results of using the proposed 3D features were compared with the results generated using 2D features. Each experiment included twenty sub-experiments with different training data. The classification performance of the classifiers was compared by McNemar's test. In addition, the GP-produced classifiers were compared in terms of classification accuracy with other learning approaches: k-nearest neighbour (kNN)[1], neural network (NN)[2] and support vector machine (SVM)[3]. These classifications were done by an open source software called Weka[4]. Note that this means all classifications were done using the same features, produced by GP-evolved segmentation.

5.1 Rose apple sorting

Rose apples are usually grouped according to size, and the size varies as weight. When farmers grade them, they estimate the size by their experience. If it is not clear, weight measurement will be employed to solve the difficulty. The samples were divided into 3 size groups comprising of big, medium and small, and there were 29, 34 and 35 images respectively. Sample images of each group from top and side views are shown in Figure 5.3. 20% of the numbers of samples in each category were randomly selected for training and 80% for testing.

Only top view images of the samples were used to extract 2D shape properties, whereas 3D shape features were evaluated from 3D objects reconstructed from top

[1]Using $k = 1$

[2]The number of units in the hidden layer approximately equals the mean value of input and output nodes, and the inputs are scaled down to the -1 to 1 range.

[3]Using a linear kernel

[4]http://www.cs.waikato.ac.nz/ml/weka/index.html

view and side view images. As can be seen, the top view shown in Figure 5.3(a–c) may not be used in order to estimate size correctly. In contrast, the side view images shown in 5.3(d–f) could be useful to sort the samples by size.

Table 5.1 shows that all classification results of employing the proposed 3D features are over 11% greater than performing by the original Jasmine, with 2D features. Using 3D shape, classifiers were able to achieve perfect classification three times, and the mean accuracy result is 95.1%, almost 20% better than that of using 2D shape classifiers. The Z values are shown in Table 5.1 and Figure 5.1. All Z scores are higher than 1.96, indicating that their performances differ significantly in the statistical sense, with the classification performance of employing 3D shape features exceeding that of using 2D shape features.

Table 5.2 presents the classification results of kNN, NN and SVM with extracted 2D and 3D shape properties, and Figure 5.2 shows the mean accuracy of them and GP classifiers. All learning approaches employing 2D shape properties gave poorer mean accuracy than employing 3D shape. Although the NN using 2D shape was able to achieve perfect classification once, its mean result was over 10% less than that when employing 3D shape. The performances of the learning approaches using 2D shape differed a little. The NN returned the highest mean result at 81.5%, marginally better than the others, and about 4% better than that generated by GP. Although the GP performance was slightly inferior to the others for using 2D shape, it was capable of achieving the highest mean accuracy, at 95.1%, using 3D shape, while the NN reached 92.6% and the kNN 90.3%. The GP achievement is also roughly 10% better than using the SVM, the least accurate classifier of the group. As mentioned previously, the GP achieved perfect classification three times; in contrast, it did not occur from the others. All classifiers employing 3D shape achieved better accuracy than using 2D shape.

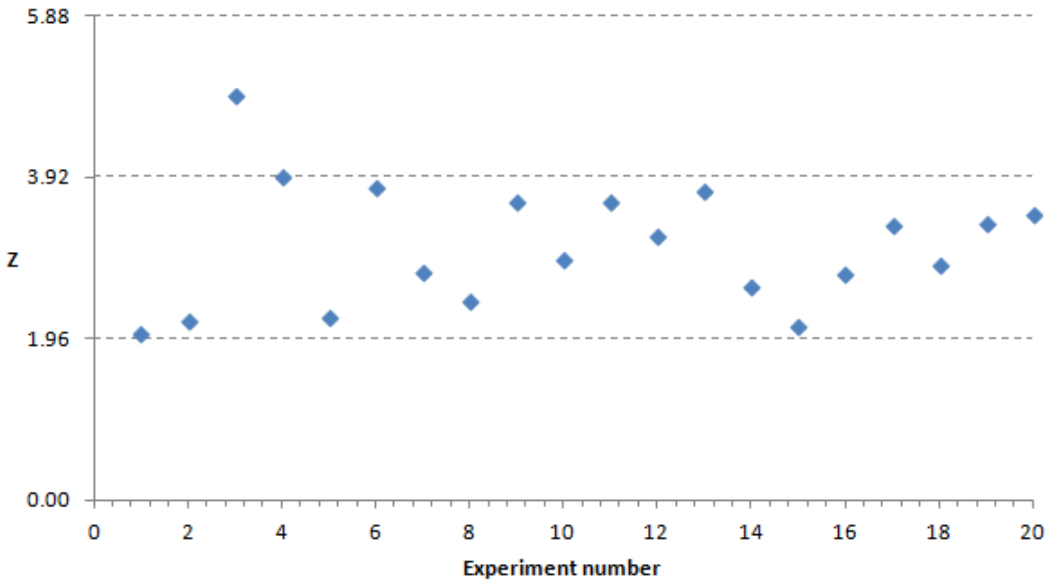


Figure 5.1: Illustration on Z scores of the experiment on rose apple sorting

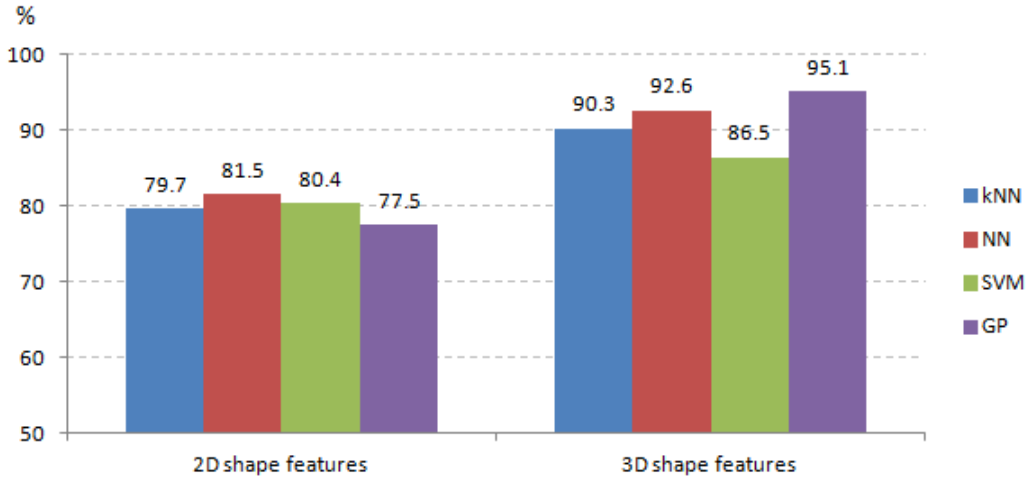


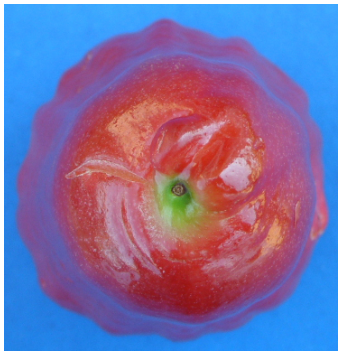
Figure 5.2: Mean accuracy of learning approaches classifying with different feature datasets of the experiment on rose apple sorting

Table 5.1: Classification results and Z scores of the experiment on rose apple sorting

Experiment number	Classification accuracy results (%)		Z score	Superior performance
	2D shape	3D shape		
1	74.4	87.2	2.01	3D shape classifiers
2	73.1	87.2	2.18	3D shape classifiers
3	66.7	100.0	4.90	3D shape classifiers
4	74.4	98.7	3.93	3D shape classifiers
5	84.6	96.2	2.22	3D shape classifiers
6	74.4	97.4	3.80	3D shape classifiers
7	80.8	94.9	2.77	3D shape classifiers
8	78.2	92.3	2.43	3D shape classifiers
9	80.8	100.0	3.61	3D shape classifiers
10	75.6	92.3	2.91	3D shape classifiers
11	79.5	98.7	3.61	3D shape classifiers
12	76.9	96.2	3.21	3D shape classifiers
13	76.9	97.4	3.75	3D shape classifiers
14	74.4	88.5	2.58	3D shape classifiers
15	76.9	89.7	2.12	3D shape classifiers
16	78.2	93.6	2.75	3D shape classifiers
17	82.1	98.7	3.33	3D shape classifiers
18	85.9	98.7	2.85	3D shape classifiers
19	74.4	94.9	3.35	3D shape classifiers
20	82.1	100.0	3.47	3D shape classifiers
Best	85.9	100.0		
Mean	77.5	95.1		
s.d.	4.5	4.3		

Table 5.2: Classification results of the other classifiers for the experiment on rose apple sorting

Experiment number	Classification accuracy results (%)					
	2D shape features			3D shape features		
	kNN	NN	SVM	kNN	NN	SVM
1	83.3	83.3	82.1	88.5	88.5	87.2
2	87.2	91.0	78.2	87.2	91.0	78.2
3	78.2	74.4	84.6	91.0	96.2	94.9
4	80.8	79.5	84.6	91.0	94.9	84.6
5	78.2	76.9	82.1	78.2	76.9	74.4
6	82.1	79.5	84.6	92.3	94.9	87.2
7	82.1	85.9	84.6	92.3	94.9	82.1
8	75.6	69.2	65.4	91.0	91.0	84.6
9	82.1	82.1	79.5	91.0	92.3	93.6
10	75.6	76.9	74.4	92.3	92.3	93.6
11	75.6	79.5	69.2	88.5	92.3	92.3
12	87.2	87.2	87.2	92.3	97.4	89.7
13	78.2	100.0	79.5	94.9	96.2	83.3
14	74.4	83.3	84.6	91.0	92.3	91.0
15	85.9	83.3	82.1	93.6	94.9	80.8
16	76.9	83.3	87.2	93.6	94.9	85.9
17	76.9	75.6	80.8	92.3	100.0	82.1
18	70.5	76.9	84.6	89.7	87.2	85.9
19	79.5	76.9	83.3	92.3	96.2	93.6
20	83.3	85.9	70.5	82.1	88.5	84.6
Best	87.2	100.0	87.2	94.9	97.4	94.9
Mean	79.7	81.5	80.5	90.3	92.6	86.5
s.d.	4.5	6.7	6.1	4.0	5.0	5.6



(a) Big size (top view)

(b) Medium size (top view)

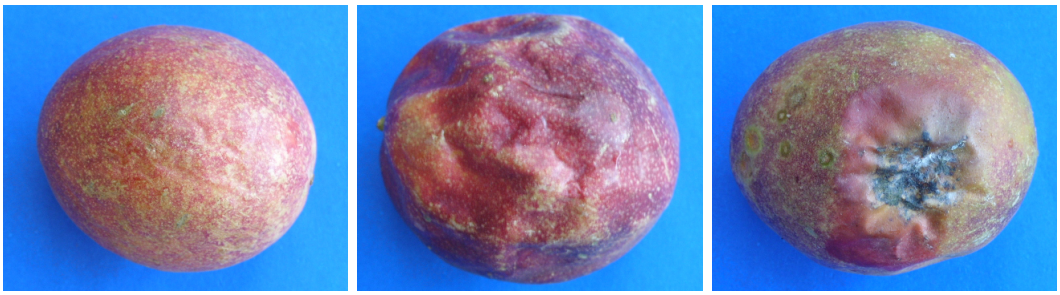
(c) Small size (top view)

(d) Big size (side view)

(e) Medium size (side view)

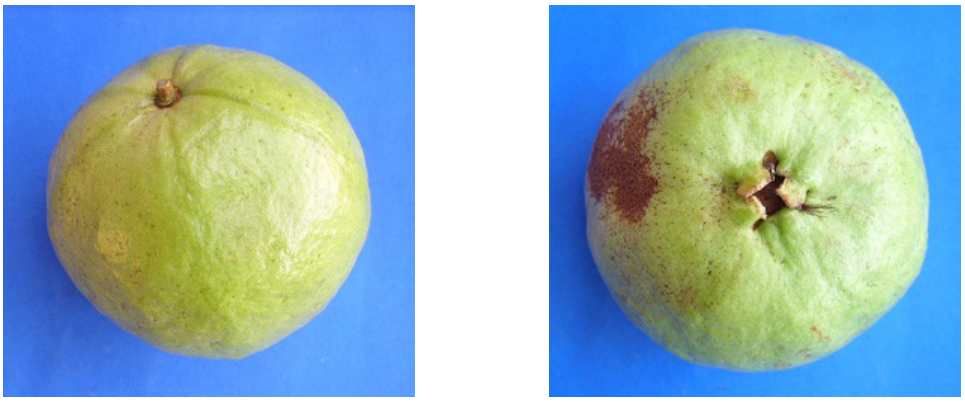
(f) Small size (side view)

Figure 5.3: Top view and side view images of rose apples



(a) Smooth skin (b) Rough skin (c) Rotten skin

Figure 5.4: Sample images of passion fruit



(a) Good skin colour (b) Defective skin

Figure 5.5: Sample images of guavas

(a) Good skin colour (b) Defective skin

Figure 5.6: Sample images of apples

5.2 Passion fruit grading

The appearances of good purple passion fruit are smooth, smoothly-coloured, undistorted, and not cracked or rotten. For passion fruit grading, fruits with a smooth skin are separated to be further graded according to their size and weight. Rough fruits are grouped in lower grades, and cracked and rotten fruits are discarded. The samples for this experiment included smooth (60 fruits), rough (42 fruits) and rotten (23 fruits) skin. Example images of each type are presented in Figure 5.4. About half the numbers of samples were randomly selected for training and the remainder for testing.

The experimental results demonstrate the benefit of using features extracted by VCA because all sub-experiments employing VCA returned accuracy results, shown in Table 5.3, that are higher than all results of using features extracted by OCA and 2D features. The mean result produced by VCA reached 90% accuracy, over 28% better than employing 2D properties and around 15% better than using features from OCA.

From Table 5.4 and Figure 5.7, all sub-experiments show Z scores between VCA and 2D features between 2.25 and 4.40, much higher than the critical value (1.96), meaning that their performance differences are significant. Half the numbers of Z values between VCA and OCA show that differences in performance are significant, and those Z scores are in the range 2–3. In order to reduce the chances of obtaining false-positive results (Type I errors), the Bonferroni correction was employed to adjust the significance level as described in section 3.5.3, so the new critical value of 2.39 was used to estimate the performance differences. The brackets indicate that differences in performance are insignificant based on the adjusted critical Z value. The performance differences between using 2D features

and VCA features are insignificant for only one sub-experiment, while Z values of the rest are still higher than the adjusted critical value. The other comparisons have fewer significant tests. The experiment demonstrates that using features extracted by VCA achieved the most accuracy for this task.

The classification results of the other classifiers are presented in Table 5.5, and the mean classification results of them and GP are compared in Figure 5.8. Results can be divided into three groups: 2D features, 3D features extracted by VCA and 3D features extracted by OCA. Overall for the first group, their mean results fall in the range 60–70% accuracy. SVM was the leader of this group. It could reach 68.1% mean accuracy, about 1.4%, 6.6% and 7.4% better than using NN, GP and kNN respectively.

For classification using 3D properties extracted by the VCA method, all learning approaches achieved over 88% mean accuracy. Their classification performances did not differ greatly. GP took the lead for this group, its mean result reaching 90%, a 0.4% difference from that produced by SVM. NN returned the poorest accuracy at 88.1%, about 0.7% less than that of kNN.

The mean results of the classifiers are moderately different for classification based on 3D features extracted from the OCA method. The NN classifiers were able to reach the best accuracy at 91.9% (twice) and achieved the best mean at 84.0%. The mean result produced by SVM is 79.5%, while 75.2% was returned from GP. The performance of kNN could only reach 74.4%, approximately 20% less than the leader.

It is noticeable that using 3D features derived from the VCA technique produced the highest accuracy for all learning approaches, while their accuracies exhibited little variation. Employing 2D features gained the poorest accuracy for classification, and the mean results of the classifiers are over 10% and 20% poorer

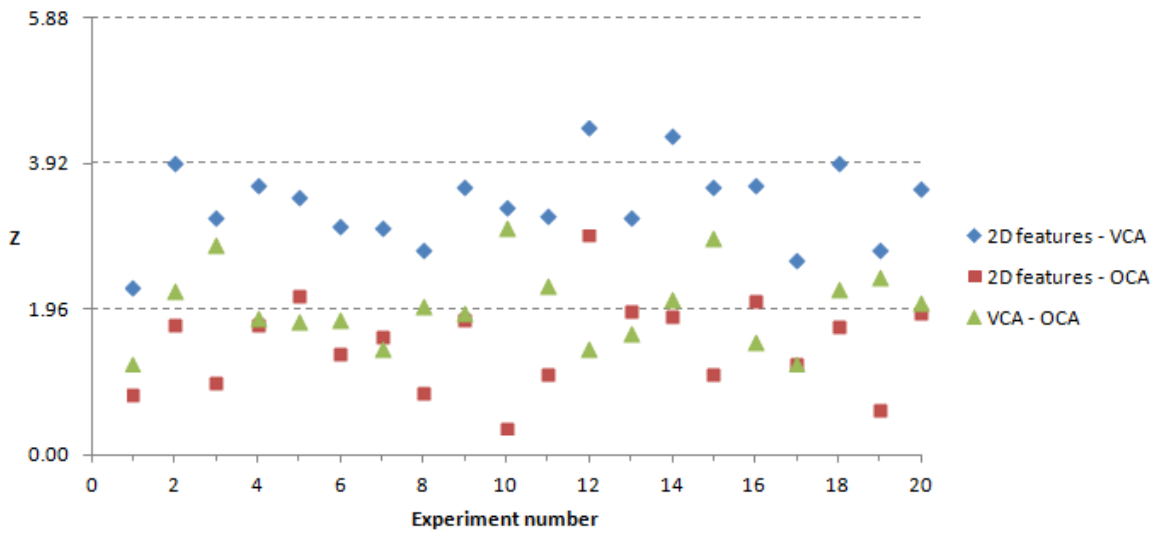


Figure 5.7: Illustration on Z scores of the experiment on passion fruit grading

than those produced by using the OCA and VCA features respectively.

5.3 Guava inspection

Guavas with smoothly-coloured skin are more attractive to customers than those with defective skin. Most defects are dark or bruised, and they are noticeable. Good and defective skin samples were employed for this task. There were 37 images of each, and sample images are shown in Figure 5.5. 10% of the numbers of samples were randomly selected for training and the remainder for testing.

This experiment employed a rather small number of training samples. Some top view images of the training sets did not show defective area, and it could have a severe impact on 2D feature analysis. Table 5.6 shows the classification results of GP using 2D features, 3D features extracted by VCA and OCA. The

Table 5.3: Classification results of the experiment on passion fruit grading

Experiment	Classification accuracy results (%)		
number	2D features	VCA	OCA
1	66.1	90.3	74.2
2	61.3	91.9	77.4
3	61.3	87.1	71.0
4	58.1	88.7	74.2
5	59.7	90.3	79.0
6	59.7	90.3	72.6
7	64.5	90.3	79.0
8	64.5	87.1	74.2
9	61.3	88.7	74.2
10	59.7	87.1	64.5
11	61.3	90.3	72.6
12	54.8	91.9	82.3
13	62.9	88.7	77.4
14	54.8	90.3	74.2
15	62.9	93.5	74.2
16	61.3	90.3	79.0
17	69.4	91.9	80.6
18	58.1	88.7	72.6
19	69.4	91.9	75.8
20	58.1	90.3	75.8
Best	69.4	93.5	82.3
Mean	61.5	90.0	75.2
s.d.	4.0	1.8	4.0

Table 5.4: Z scores of the experiment on passion fruit grading

Experiment	2D features - VCA		2D features - OCA		VCA - OCA	
number	Z	Superior	Z	Superior	Z	Superior
1	2.25	(VCA)	0.83	-	1.25	-
2	3.93	VCA	1.77	-	2.22	(VCA)
3	3.20	VCA	0.98	-	2.85	VCA
4	3.62	VCA	1.77	-	1.87	-
5	3.46	VCA	2.16	(OCA)	1.81	-
6	3.08	VCA	1.37	-	1.84	-
7	3.06	VCA	1.60	-	1.46	-
8	2.77	VCA	0.86	-	2.02	(VCA)
9	3.60	VCA	1.84	-	1.94	-
10	3.34	VCA	0.37	-	3.06	VCA
11	3.21	VCA	1.11	-	2.29	(VCA)
12	4.40	VCA	2.97	OCA	1.44	-
13	3.20	VCA	1.94	-	1.66	-
14	4.29	VCA	1.89	-	2.12	(VCA)
15	3.60	VCA	1.11	-	2.94	VCA
16	3.62	VCA	2.09	(OCA)	1.55	-
17	2.62	VCA	1.25	-	1.25	-
18	3.93	VCA	1.75	-	2.25	(VCA)
19	2.77	VCA	0.61	-	2.41	VCA
20	3.59	VCA	1.92	-	2.07	(VCA)

Table 5.5: Classification results of the other classifiers for the experiment on passion fruit grading

Experiment number	Classification accuracy results (%)								
	2D features			VCA			OCA		
	kNN	NN	SVM	kNN	NN	SVM	kNN	NN	SVM
1	62.9	64.5	64.5	91.9	90.3	91.9	71.0	77.4	71.0
2	69.4	75.8	67.7	80.6	90.3	79.0	74.2	91.9	79.0
3	66.1	64.5	54.8	90.3	88.7	87.1	69.4	74.2	72.6
4	56.5	62.9	69.4	87.1	87.1	90.3	72.6	75.8	72.6
5	61.3	62.9	71.0	93.5	87.1	90.3	72.6	87.1	82.3
6	59.7	67.7	66.1	83.9	79.0	88.7	82.3	85.5	80.6
7	53.2	62.9	62.9	87.1	82.3	83.9	80.6	82.3	79.0
8	53.2	74.2	71.0	90.3	85.5	87.1	77.4	82.3	79.0
9	58.1	71.0	71.0	91.9	87.1	93.5	72.6	83.9	82.3
10	53.2	64.5	59.7	91.9	83.9	90.3	82.3	83.9	85.5
11	69.4	72.6	69.4	80.6	90.3	90.3	74.2	83.9	80.6
12	61.3	59.7	69.4	90.3	91.9	90.3	79.0	90.3	82.3
13	59.7	69.4	71.0	87.1	87.1	91.9	64.5	74.2	71.0
14	59.7	67.7	61.3	87.1	87.1	91.9	69.4	90.3	82.3
15	59.7	66.1	71.0	91.9	87.1	90.3	74.2	83.9	79.0
16	61.3	69.4	71.0	93.5	87.1	90.3	69.4	80.6	83.9
17	62.9	72.6	74.2	88.7	91.9	91.9	72.6	87.1	85.5
18	61.3	61.3	71.0	87.1	88.7	88.7	79.0	91.9	83.9
19	67.7	59.7	69.4	90.3	93.5	91.9	75.8	87.1	75.8
20	58.1	64.5	75.8	90.3	95.2	91.9	74.2	87.1	82.3
Best	69.4	75.8	75.8	93.5	95.2	93.5	82.3	91.9	85.5
Mean	60.7	66.7	68.1	88.8	88.1	89.6	74.4	84.0	79.5
s.d.	4.8	4.8	5.1	3.8	3.8	3.3	4.7	5.5	4.6

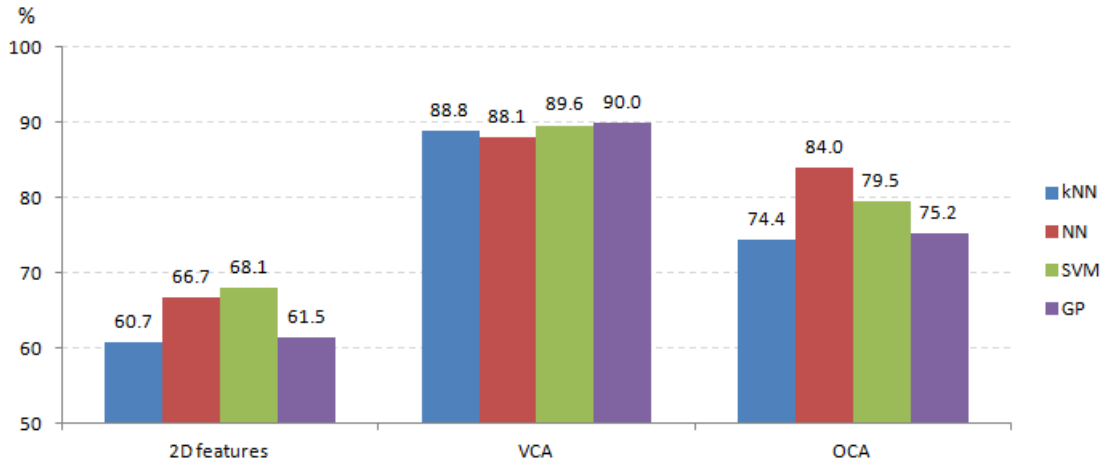


Figure 5.8: Mean accuracy of learning approaches classifying with different feature datasets of the experiment on passion fruit grading

3D classifiers based on OCA achieved perfect accuracy once, and generally over 90% accuracy. The mean result from OCA reached 94.5%, around 18% greater than that of 2D classifiers and 5% higher than that of using VCA. Table 5.7 and Figure 5.9 presents Z values of this experiment. The Z scores between features extracted by OCA and 2D features exceed the critical value 18 times, with values between 2.12 and 3.61. Performance differences between using properties of VCA and 2D characteristics are significant 11 times, and the Z values were produced in the range 2–3.

After adjustment on the critical value (Z = 2.39), the comparisons of 2D features vs. VCA and 2D features vs. OCA have fewer significant tests. However, more than half the numbers of tests of 2D features vs. OCA are still significant, and there is no performance difference of VCA vs. OCA. This experiment demonstrates that employing features of OCA improved performance in this type of task.

Table 5.8 shows classification results produced by kNN, NN and SVM in three

groups, using different feature datasets consisting of 2D features, 3D features based on VCA and 3D features based on OCA. Figure 5.10 presents the mean results of GP and the other approaches. It is clear that the GP classifiers achieve the best mean accuracy in every group. The other approaches produced mean classification results that are not much different to each other, except for the last group, for which SVM did not perform well. Its performance was dramatically inferior to the other learning approaches, its mean accuracy of 75.5% being nearly 20% poorer than that of the GP and close to that of using 2D features.

The kNN classifier was able to achieve the second position in all groups, with mean results of 75.8%, 82.4% and 87.0% respectively. These are slightly more those produced by the NN classifiers in each group. The kNN, NN and GP approaches with using 3D features extracted by the OCA method performed better than employing the other feature sets. The achievement of GP, kNN and NN is of the same order in all groups. However, the VCA features were used more appropriately than the OCA features for the SVM approach.

5.4 Apple defect discrimination

Dropping and crashing apples result in skin defects. The bruise areas are darker than the normal skin. They may be not visible immediately, but they gradually appear. Royal gala apples were inspected for this task. This experiment processed without calyx and stem end detection. The samples consisted of 76 images of good quality apples and 136 images of defective ones; example images are shown in Figure 5.6. They were randomly selected with 30% for training and 70% for testing.

From Table 5.9, the mean results of using features of VCA and OCA reach over

Table 5.6: Classification results of the experiment on guava grading

Experiment	Classification accuracy results (%)		
number	2D features	VCA	OCA
1	80.3	92.4	95.5
2	71.2	89.4	92.4
3	78.8	92.4	93.9
4	71.2	90.9	90.9
5	77.3	92.4	97.0
6	75.8	86.4	90.9
7	77.3	87.9	93.9
8	77.3	87.9	92.4
9	80.3	92.4	98.5
10	83.3	89.4	97.0
11	74.2	90.9	90.9
12	75.8	90.9	92.4
13	75.8	93.9	95.5
14	78.8	90.9	97.0
15	72.7	86.4	95.5
16	75.8	86.4	93.9
17	80.3	89.4	100.0
18	78.8	92.4	92.4
19	81.8	92.4	93.9
20	74.2	92.4	95.5
Best	83.3	93.9	100.0
Mean	77.1	90.4	94.5
s.d.	3.3	2.4	2.6

Table 5.7: Z scores of the experiment on guava grading

Experiment	2D features - VCA		2D features - OCA		VCA - OCA	
number	Z	Superior	Z	Superior	Z	Superior
1	2.02	(VCA)	2.60	OCA	0.41	-
2	2.46	VCA	2.91	OCA	0.32	-
3	2.41	VCA	2.12	(OCA)	0.00	-
4	2.50	VCA	2.91	OCA	-0.32	-
5	2.12	(VCA)	2.75	OCA	0.89	-
6	1.66	-	2.12	(OCA)	0.55	-
7	1.46	-	2.43	OCA	0.87	-
8	1.55	-	2.25	(OCA)	0.60	-
9	2.02	(VCA)	3.18	OCA	1.50	-
10	0.87	-	2.22	(OCA)	1.33	-
11	2.43	VCA	2.43	OCA	-0.32	-
12	1.92	-	2.29	(OCA)	0.00	-
13	2.59	VCA	2.91	OCA	0.00	-
14	2.94	VCA	0.00	-	1.22	-
15	1.84	-	3.06	OCA	1.44	-
16	1.55	-	2.94	OCA	1.21	-
17	1.25	-	3.33	OCA	2.27	(VCA)
18	2.22	(VCA)	1.94	-	-0.35	-
19	1.81	-	1.87	-	0.00	-
20	2.75	VCA	3.61	OCA	0.41	-

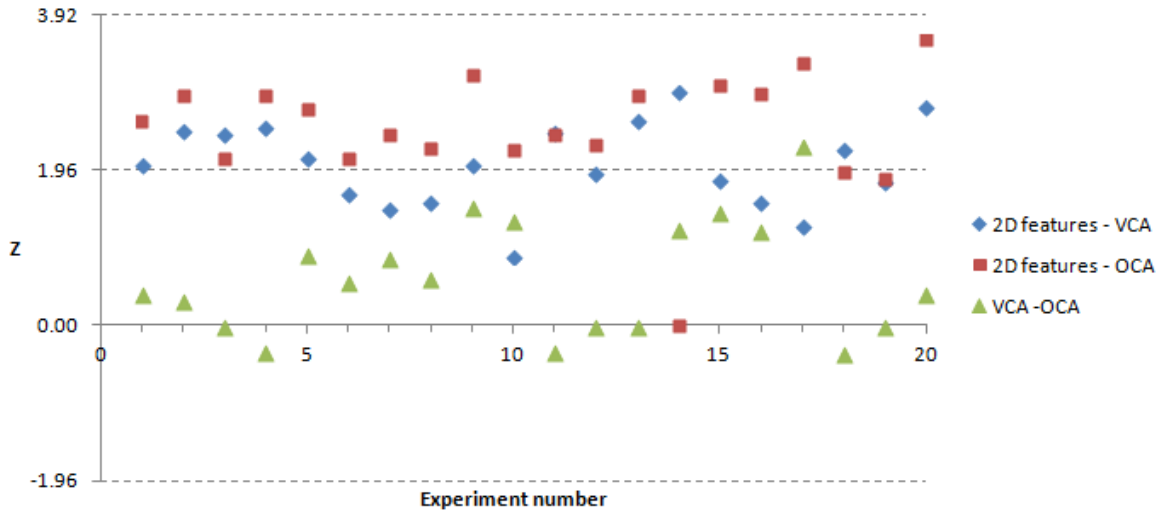


Figure 5.9: Illustration on Z scores of the experiment on guava grading

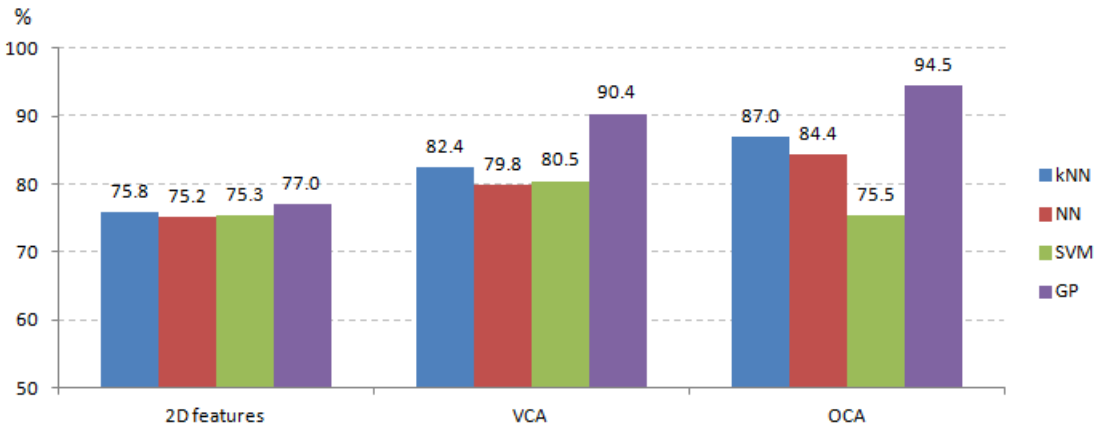


Figure 5.10: Mean accuracy of learning approaches classifying with different feature datasets of the experiment on guava grading

Table 5.8: Classification results of the other classifiers for the experiment on guava grading

Experiment number	Classification accuracy results (%)								
	2D features			VCA			OCA		
	kNN	NN	SVM	kNN	NN	SVM	kNN	NN	SVM
1	71.2	66.7	69.7	78.8	78.8	81.8	83.3	75.8	68.2
2	54.5	78.8	74.2	78.8	68.2	69.7	90.9	78.8	84.8
3	77.3	83.3	77.3	84.8	86.4	87.9	89.4	97.0	81.8
4	72.7	72.7	74.2	87.9	83.3	78.8	90.9	87.9	68.2
5	83.3	81.8	77.3	77.3	80.3	86.4	90.9	86.4	86.4
6	71.2	68.2	71.2	81.8	80.3	78.8	84.8	83.3	78.8
7	81.8	75.8	72.7	89.4	87.9	86.4	81.8	81.8	66.7
8	89.4	87.9	87.9	83.3	68.2	66.7	89.4	78.8	72.7
9	84.8	87.9	78.8	81.8	80.3	84.8	90.9	83.3	77.3
10	92.4	89.4	80.3	89.4	87.9	87.9	89.4	89.4	83.3
11	81.8	75.8	71.2	89.4	80.3	78.8	81.8	92.4	71.2
12	69.7	71.2	68.2	84.8	78.8	84.8	90.9	72.7	74.2
13	83.3	74.2	77.3	92.4	87.9	89.4	92.4	81.8	83.3
14	78.8	81.8	81.8	86.4	86.4	86.4	84.8	97.0	83.3
15	60.6	53.0	60.6	80.3	71.2	69.7	95.5	95.5	62.1
16	86.4	78.8	84.8	78.8	78.8	81.8	84.8	92.4	75.8
17	45.5	47.0	69.7	80.3	83.3	87.9	77.3	84.8	65.2
18	74.2	75.8	72.7	63.6	65.2	65.2	78.8	71.2	68.2
19	84.8	83.3	78.8	83.3	84.8	80.3	87.9	87.9	86.4
20	72.7	69.7	77.3	75.8	77.3	75.8	83.3	69.7	71.2
Best	92.4	89.4	87.9	92.4	87.9	89.4	92.4	97.0	86.4
Mean	75.8	75.2	75.3	82.4	79.8	80.5	87.0	84.4	75.5
s.d.	11.8	10.8	6.2	6.4	6.9	7.5	4.9	8.3	7.7

95% accuracy and are about 15% better than that of employing 2D features. Using VCA and OCA properties classifiers could reach 98.6% and 98.0% respectively, whereas with 2D features, the highest accuracy was 85.1%. The Z values are shown in Table 5.10 and Figure 5.11. They show that the performances in all sub-experiments are significant for employing features extracted from VCA and OCA compared with using 2D features. Their Z values are in the range 2–5. In contrast, differences in performance using VCA and OCA are insignificant because all Z scores are substantially less than the critical value of 1.96. The Bonferroni correction was used to adjust the critical value so as to decrease the opportunities of obtaining false-positive results (Type I errors). The comparison between using 2D features and VCA features has one non-significant test, and the Z values of the other sub-experiments are still higher than 2.39 (the adjusted critical value). Two tests of the comparison of 2D features vs. OCA are non-significant, while the performance differences of the other remain significant.

The classification results of the other learning principles, with 2D features, 3D features estimated from VCA, and 3D features from OCA are shown in Table 5.11. Figure 5.12 presents the mean classification results of all learning approaches. Overall, using 2D features achieved lower accuracy than employing 3D features derived from either technique. All learning approaches with 3D features extracted from either method were able to achieve higher accuracy, and their mean results are in the range 92–96% accuracy.

The SVM classifiers were able to return the best accuracy for all groups (88.5% using 2D features, 98.6% using VCA features and 98.6% using OCA features). The SVM also achieves the best mean accuracy at 82.3% for using 2D features. As SVM finds linear boundaries between classes, this is presumably a case in which this works particularly well. GP does not have this linear constraint and

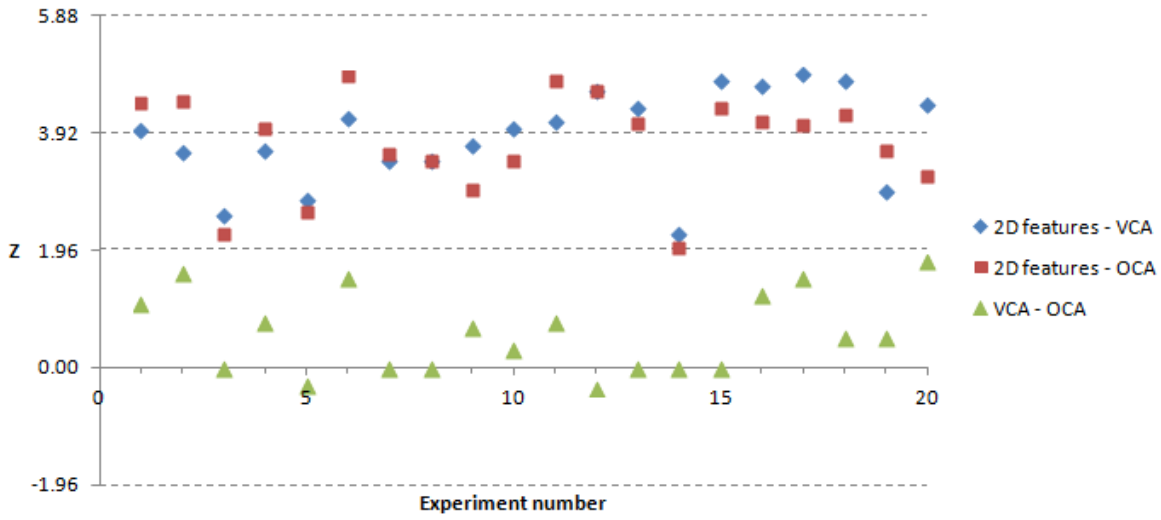


Figure 5.11: Illustration on Z scores of the experiment on apple grading

presumably has found alternative boundaries between classes that work less well in this case. Whereas the highest mean results with 3D features were produced by the NN classifiers, 95.3% for VCA and 95.7% for OCA. The GP approach achieved the second position in every group. Its mean accuracy was a little less than that of the leader in each group, and it is exactly equal to that of the SVM for the last group. Processing by the kNN classifiers resulted in the poorest accuracy for all groups, but they are not much less than those of the superior ones.

5.5 Experimental conclusions

These experiments were done for two purposes: the first is to make a comparison between the classification performance of 2D and 3D shape classifiers. Rose apple sorting was applied for this task. The second aim was to evaluate the value

Table 5.9: Classification results of the experiment on apple grading

Experiment	Classification accuracy results (%)		
number	2D features	VCA	OCA
1	81.1	93.2	95.9
2	81.1	93.9	98.0
3	83.1	92.6	91.9
4	82.4	94.6	96.6
5	73.0	95.3	95.3
6	77.0	93.9	96.6
7	81.1	93.9	94.6
8	82.4	94.6	95.3
9	83.1	93.9	94.6
10	81.1	95.3	93.9
11	80.4	95.3	97.3
12	79.7	96.6	96.6
13	79.7	95.3	94.6
14	85.1	93.2	92.6
15	79.7	97.3	96.6
16	79.1	96.6	93.9
17	74.3	95.9	93.2
18	81.8	98.6	97.3
19	83.8	94.6	95.9
20	82.4	96.6	92.6
Best	85.1	98.6	98.0
Mean	80.6	95.1	95.2
s.d.	3.0	1.5	1.8

Table 5.10: Z scores of the experiment on apple grading

Experiment	2D features - VCA		2D features - OCA		VCA - OCA	
number	Z	Superior	Z	Superior	Z	Superior
1	3.97	VCA	4.42	OCA	1.06	-
2	3.60	VCA	4.46	OCA	1.58	-
3	2.55	VCA	2.23	(OCA)	0.00	-
4	3.62	VCA	4.00	OCA	0.76	-
5	2.80	VCA	2.60	OCA	-0.29	-
6	4.18	VCA	4.87	OCA	1.50	-
7	3.46	VCA	3.59	OCA	0.00	-
8	3.47	VCA	3.46	OCA	0.00	-
9	3.73	VCA	2.97	OCA	0.67	-
10	4.00	VCA	3.46	OCA	0.29	-
11	4.12	VCA	4.80	OCA	0.76	-
12	4.62	VCA	4.62	OCA	-0.35	-
13	4.35	VCA	4.09	OCA	0.00	-
14	2.25	(VCA)	2.00	(OCA)	0.00	-
15	4.80	VCA	4.35	OCA	0.00	-
16	4.72	VCA	4.12	OCA	1.22	-
17	4.90	VCA	4.07	OCA	1.50	-
18	4.80	VCA	4.23	OCA	0.50	-
19	2.94	VCA	3.62	OCA	0.50	-
20	4.40	VCA	3.20	OCA	1.77	-

Table 5.11: Classification results of the other classifiers for the experiment on apple grading

Experiment number	Classification accuracy results (%)								
	2D features			VCA			OCA		
	kNN	NN	SVM	kNN	NN	SVM	kNN	NN	SVM
1	73.0	66.9	75.7	93.2	94.6	93.9	92.6	93.9	94.6
2	77.7	81.1	83.8	90.5	98.0	93.9	93.9	98.6	94.6
3	80.4	83.1	86.5	92.6	95.9	94.6	91.9	96.6	93.9
4	74.3	70.9	76.4	91.9	93.9	92.6	91.2	96.6	94.6
5	75.0	84.5	76.4	93.2	96.6	95.3	92.6	95.3	95.3
6	77.7	81.8	79.7	93.9	96.6	95.9	91.9	95.3	93.9
7	81.8	82.4	83.1	96.6	95.9	97.3	92.6	96.6	95.3
8	75.7	84.5	85.1	93.2	98.0	95.9	94.6	95.3	95.9
9	77.7	83.1	87.8	93.9	95.3	95.9	95.3	96.6	95.9
10	77.0	79.7	81.1	94.6	94.6	95.9	96.6	95.9	97.3
11	81.8	79.7	81.8	91.9	95.3	93.9	93.9	98.0	98.6
12	76.4	77.7	83.8	92.6	93.9	93.9	95.3	97.3	95.3
13	71.6	73.6	81.1	85.8	92.6	84.5	87.2	95.3	88.5
14	83.8	85.1	88.5	97.3	95.3	95.3	92.6	94.6	95.3
15	77.7	80.4	77.0	95.3	98.0	97.3	91.9	93.9	94.6
16	78.4	73.6	83.1	93.2	91.2	95.9	94.6	94.6	95.3
17	77.0	74.3	78.4	93.2	91.9	93.2	92.6	95.9	93.9
18	84.5	84.5	87.2	92.6	95.3	98.6	92.6	95.9	98.6
19	77.0	83.8	85.1	95.3	98.6	94.6	91.9	95.3	95.9
20	81.8	85.1	84.5	95.9	94.6	94.6	90.5	92.6	95.9
Best	84.5	85.1	88.5	97.3	98.0	98.6	96.6	98.6	98.6
Mean	78.0	79.8	82.3	93.3	95.3	94.7	92.8	95.7	95.2
s.d.	3.4	5.3	4.0	2.4	2.0	2.8	2.0	1.4	2.1

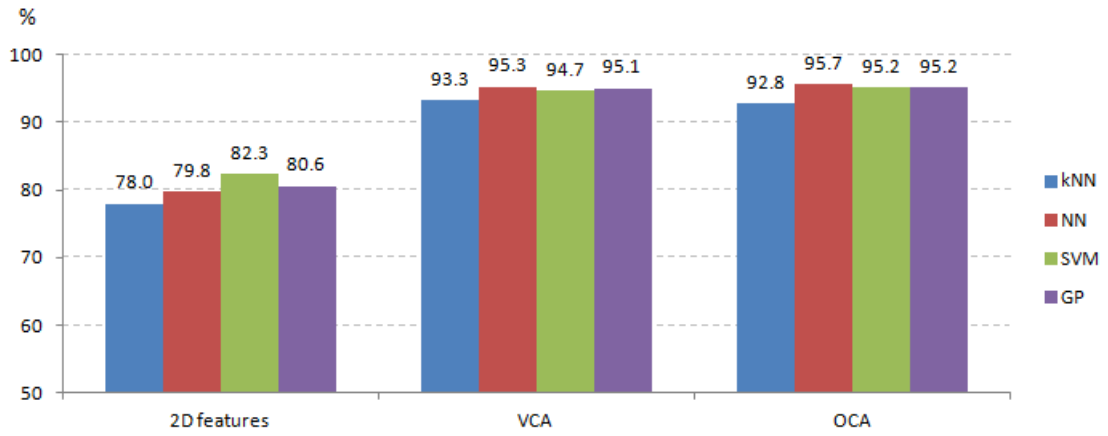


Figure 5.12: Mean accuracy of learning approaches classifying with different feature datasets of the experiment on apple grading

of the 3D features (including shape, colour and texture properties) extracted by the proposed methods, VCA and OCA. These experiments were done on passion fruit, guava and apple grading to compare the classification performance between employing 2D and 3D operations. 2D extraction functions analysed only the top view images of objects while the 3D features were estimated from 3D objects reconstructed from multiple view images. As experiments using only one viewpoint of objects might not be adequate for some types of agricultural produce, grading of rose apples, passion fruit, guavas and apples were chosen because when they are laid naturally for inspection, approximately half their bottom shape tends to be behind the top side.

A statistically-valid comparison between the classification performance of using 2D and 3D features was made using McNemar's test. All experiments were also performed using other machine learning algorithms, k-nearest neighbour, neural network and support vector machine. Their classification accuracy was compared with that produced by the proposed GP system.

For the first group of experiments, it was found that 3D shape features were dramatically effective for rose apple sorting. All 3D classifiers were superior to the 2D classifiers, and the generated Z scores fall between 2 and 5. However, this is likely to be a consequence of the advantage of using side view images for 3D feature analysis because they presented the size of rose apples much better than top view images, and they were not employed for 2D property extraction.

The other group of experiments were done for a comparison on the classification performance between using all 2D and 3D operations (shape, colour and texture extraction). Since some important or defective areas were not presented in top view images, the inspection reliability tended towards 3D property analysis. Using 3D characteristics was sharply superior to employing 2D features for all experiments. The 3D features extracted from VCA resulted in the best results for the experiment on passion fruit grading, whereas employing 3D features extracted from OCA took the lead for the experiment on guava defect inspection. In addition, features extracted from both methods demonstrated their value for the experiment on apple defect detection.

Other learning approaches were used on the same data. Overall, the classification results of the other classifiers tend to be similar to the results generated by the GP classifiers. Again, using 3D features extracted from both proposed techniques out-performed employing 2D features. The similarity in performances suggests that there is not much more that can be obtained from the inputs to the classifiers than they achieve. Although the results from GP and other machine learning techniques do not vary greatly, one advantage of using GP rather than the others is that the segmentation and feature description stages were all performed using GP — conventional vision systems would require them to be written from scratch by a human — so GP provides a consistent framework for all the

stages of a complete vision system.

It can be seen that the 3D classifiers achieved more accuracy than the 2D classifiers for all experiments and for the learning approaches applied, the performance differences being significant for all tasks. It indicates that images with only one view of objects can not always represent the whole physical properties for some types of agricultural produce, so using models generated from multiple view images of objects should be better to estimate their shape and qualities. This result is consistent with what one would expect.

CHAPTER 6

EXTENDING EXISTING SYSTEMS

Genetic programming (GP) is a learning approach that is applied to solve a wide range of classification-related tasks [95], in fields such as medicine, finance and engineering. The GP idea offers many interesting advantages. One of them is the flexibility that allows the technique to be adjusted to a particular problem. GP programs are presented as trees that are able to fit in various domains. Section 2.4 presents the GP evolution process and various ideas to adapt a GP process in order to produce more effective GP classifiers.

An issue is considered for this thesis: the need to re-train a multi-class classifier that has initially been evolved using GP to accommodate new features or new classes. Therefore, this chapter addresses two particular questions. Firstly, when a solution has already been evolved using a particular set of features, is there some way in which a further feature can be accommodated? Secondly, if a system has been evolved to distinguish N classes, can an $(N+1)$th class be added? In

both cases, extending the capability of the system needs to be done with less computation than evolving a new solution from scratch.

The following sections will present techniques proposed to clarify the arisen questions. The presented ideas will be proved by experiments involving agricultural produce grading. This domain has a high tendency to be related to the suggested questions because agricultural produce is natural material that is probably found further characteristics and cultivars. However, the approaches that are proposed should be able to be adapted to any domain.

Parts of this work were published in Electronics, Computer and Computation (ICECCO), 2013 International Conference on [179].

6.1 Incorporating Additional Features Using Mutation

Firstly, let us consider the case in which an existing, automatically-derived vision system for grading produce is extended to accommodate a new feature. After the vision system is trained on samples of produce to generate a working solution to a particular problem, and that solution has been validated to work well on unseen test data, it can be used in the field. However, if additional useful properties of the samples are subsequently found, it may be that the classification accuracy could be improved by incorporating the new properties. Clearly, it will be advantageous if this can be done by adapting the already-trained classifier more quickly than training from scratch.

Mutation is a genetic operation that changes a small portion of a GP tree with a randomly generated sub-tree. [180, 181], cited in [131], suggested that the

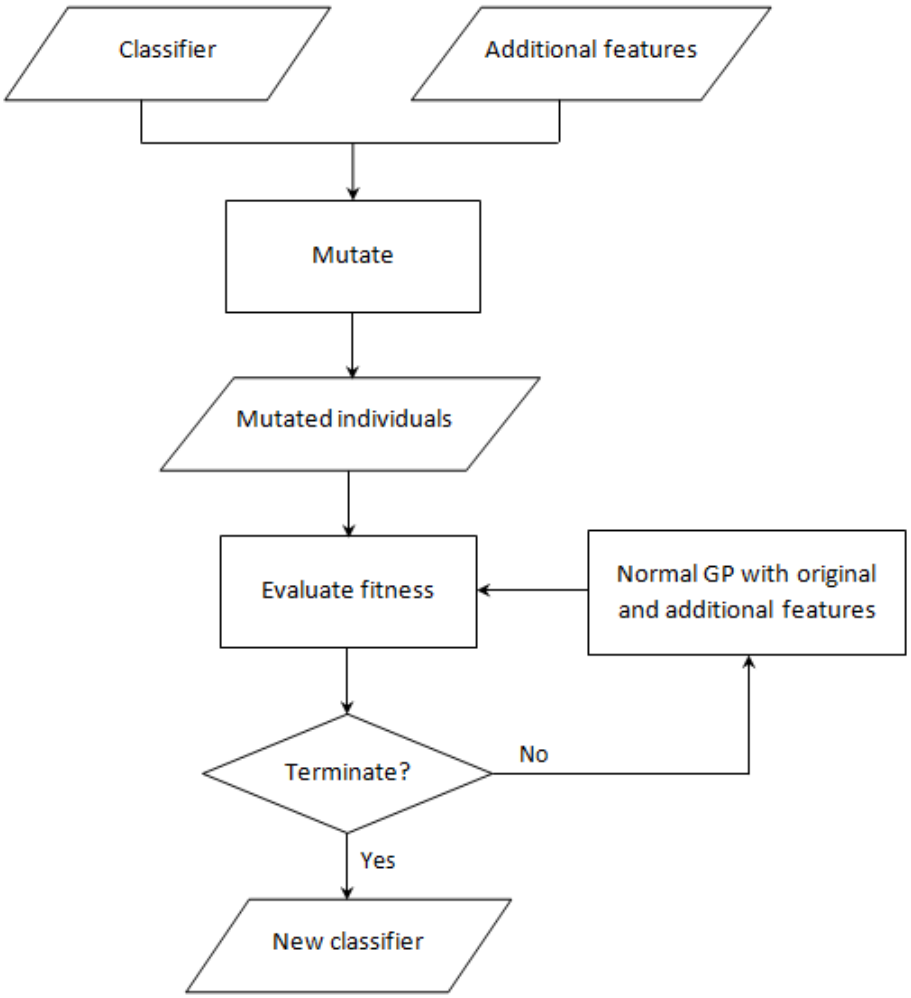


Figure 6.1: Incorporating new features using mutation

achievement of mutation is actually superior or equal to that of crossover. In this research, the mutation operator is applied to adjust the existing solution with additional operators that introduce the new features: first, after the additional features are evaluated for their effectiveness, the best n[1] features are selected randomly to form new sub-trees. The original classifier is then mutated with them to produce an initial population so that all individuals of the initial population contain the previously involved solution mutated with additional operators. Third, the fitness of the mutated programs is determined on the training data. If the computed fitness of a program can reach optimum performance, the system will return it as a new solution immediately; otherwise, normal GP evolution proceeds in the usual way. For normal GP processing, effective original and additional features are used to produce programs. This method is illustrated in Figure 6.1.

6.2 Experiments on Introducing New Features Using Mutation

Using the original Jasmine, object classifiers were generated based on shape, object position and grey-scale features. Then, the additional features described in section 3.4, including colour and texture properties, were added to the GP 'engine' and mutated with the original classifier as described in the previous section to give a population of new programs, which were then evolved.

This procedure is demonstrated in three experiments: small orange variety discrimination, lime maturity inspection, and the maturity and size determination of tomatoes. For small oranges and limes, the samples were roughly the same size,

[1] This research used $n \geqslant 20$.

but their colour and texture properties were different. For tomato samples, the colour differed moderately in each group, and the size of one category was obviously smaller than the others. Each experiment includes twenty sub-experiments with different training data. The experimental results of the proposed technique (classifiers generated from the original Jasmine mutated to include extended features comprising of colour and texture operators) are compared with the original classifiers (using only morphological and grey-scale features) and the use of all features *ab initio*. The classification performance of all techniques are compared in each pair by using McNemar's test, and Z values are shown and illustrated graphically.

6.2.1 Small orange cultivar discrimination

Small oranges of cultivars of Sadung and Masui were employed. The samples were roughly in the same size group but slightly different in colour and texture. There were 60 images per each type, and sample pictures are shown in Figure 6.5. They were randomly chosen with 40% for training processes and the remainder for testing.

The classification results are shown in Table 6.1. The *ab initio* evolution and the proposed method could both achieve perfect classification. Their classification results are higher than 90% whereas the results generated using the original operations are in 59–75% accuracy. The mean accuracy results of the *ab initio* and proposed approaches are 94.7% and 93.9% respectively, and they are over 25% better than that produced by the original system. Most importantly, the additional colour and texture features were effective enough to be accommodated in the existing classifiers; as a result, the proposed mutation process yielded all working

classifiers without necessitating further evolution. Considering on spending time to generate a new classifier (for sup-experiment #1), the *ab initio* evolution took about 95 milliseconds, while using the proposed technique spent approximately 94 milliseconds (processing on Intel 2.27 GHz Core i5 with 8 GB of main memory under the Windows operating system). It indicates that the spending time of them does not differ to generate a classifier; however, for the overall training process, the *ab initio* evolution needed reevaluation on the effectiveness of all the existing operations, while the proposed technique could skip that part.

Table 6.2 and Figure 6.2 presents the calculated Z values. All Z values of comparisons involving original features vs. *ab initio* evolution and original features vs. the proposed method are greater than 2.3, showing the accomplishment of the *ab initio* and proposed approaches compared with using the original features. Meanwhile, all Z values of *ab initio* evolution vs. the proposed method are dramatically less than 1.96, and most of them are about zero. This means that differences in performance between the *ab initio* and proposed approaches are insignificant.

6.2.2 Lime maturity classification

Ripe limes are pure green in color, subsequently changing to yellow and brown. The sample images were fruits of roughly the same size but different ripenesses, with 40 images of appropriately-ripe, 40 images of highly-ripe and 44 images of overly-ripe fruit; sample pictures are shown in Figure 6.6. Exactly half the numbers of samples were selected randomly for training, the remaining samples being used for testing.

From the classification results presented in Table 6.3, the mean accuracy result of the proposed technique (91.0%) is sharply higher than that of the original

Table 6.1: Classification results of the experiment on small orange cultivar discrimination as regards adding new features

Experiment number	Classification accuracy results (%)		
	Original features	*Ab initio* evolution	Proposed method
1	62.5	93.1	94.4
2	66.7	97.2	94.4
3	66.7	91.7	90.3
4	70.8	97.2	95.8
5	72.2	97.2	94.4
6	59.7	97.2	94.4
7	66.7	93.1	91.7
8	68.1	90.3	90.3
9	68.1	95.8	94.4
10	63.9	100.0	98.6
11	72.2	95.8	93.1
12	72.2	95.8	100.0
13	72.2	91.7	90.3
14	69.4	90.3	93.1
15	70.8	94.4	95.8
16	68.1	90.3	90.3
17	69.4	100.0	98.6
18	65.3	93.1	94.4
19	75.0	94.4	91.7
20	75.0	95.8	91.7
Best	75.0	100.0	100.0
Mean	68.6	94.7	93.9
s.d.	4.0	3.0	2.9

Table 6.2: Z scores of the experiment on small orange cultivar discrimination as regards adding new features

Experiment	Original - *Ab initio*		Original - Proposed		*Ab initio* - Proposed	
number	Z	Superior	Z	Superior	Z	Superior
1	3.71	*Ab initio*	4.09	Proposed	0.00	-
2	4.12	*Ab initio*	3.59	Proposed	0.41	-
3	3.21	*Ab initio*	2.97	Proposed	0.00	-
4	3.75	*Ab initio*	3.47	Proposed	0.00	-
5	3.62	*Ab initio*	3.35	Proposed	0.41	-
6	3.88	*Ab initio*	3.33	Proposed	0.71	-
7	3.60	*Ab initio*	3.33	Proposed	0.00	-
8	2.83	*Ab initio*	2.83	Proposed	0.00	-
9	3.88	*Ab initio*	3.75	Proposed	0.00	-
10	4.90	*Ab initio*	4.80	Proposed	0.00	-
11	3.67	*Ab initio*	2.80	Proposed	0.41	-
12	3.49	*Ab initio*	4.25	Proposed	1.15	-
13	2.55	*Ab initio*	2.31	Proposed	0.00	-
14	3.06	*Ab initio*	3.20	Proposed	0.50	-
15	3.67	*Ab initio*	3.62	Proposed	0.00	-
16	2.94	*Ab initio*	3.88	Proposed	0.95	-
17	4.48	*Ab initio*	4.36	Proposed	0.00	-
18	3.47	*Ab initio*	4.00	Proposed	0.00	-
19	2.91	*Ab initio*	2.91	Proposed	0.00	-
20	3.06	*Ab initio*	3.06	Proposed	0.00	-

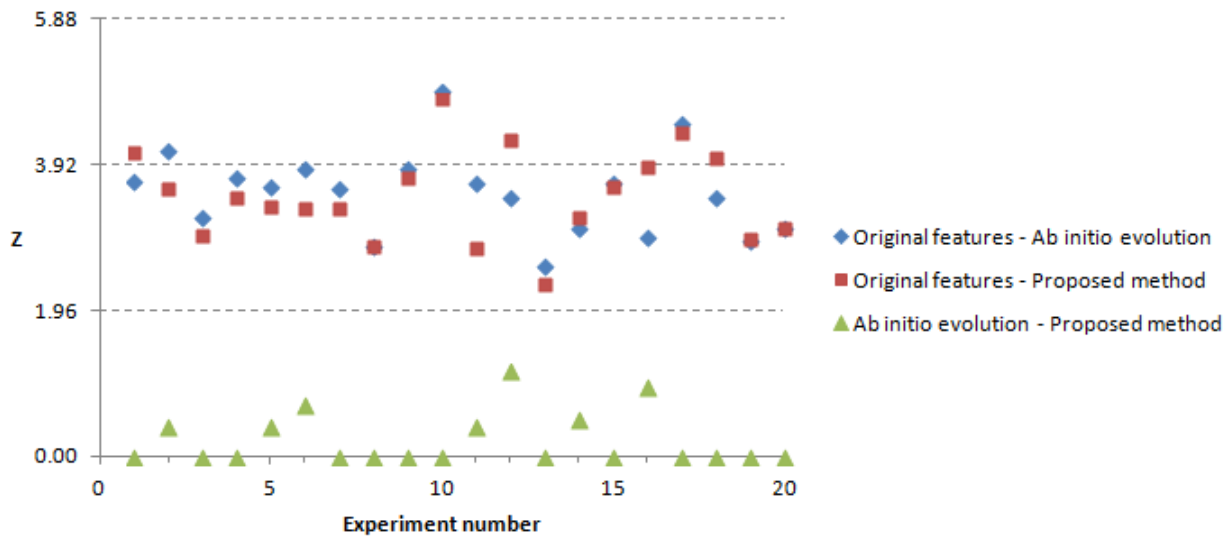


Figure 6.2: Illustration on Z scores of the experiment on small orange cultivar discrimination as regards adding new features

classifier (59.1%) but slightly lower than can be achieved using *ab initio* evolution (92.7%); this is insignificant, given the number of tests performed (i.e., images of only 60 fruits). Employing all features with *ab initio* evolution achieved a peak accuracy of 96.8% twice, and the proposed method achieved at best an accuracy of 95.2% twice. Both techniques returned their poorest accuracy at 85.5%, still higher than the best when using only the original features. The best result produced by using the old features is just about 75%, and the other classification results are less than 63%.

For all sub-experiments of the proposed approach, at least one sub-classifier of the binary classifier series generated by the proposed mutation technique needed no further evolution. This demonstrates that the system extended with the additional operators achieved more accuracy than performing by using only the original features, and the added operators were able to improve the performance of the existing solutions directly.

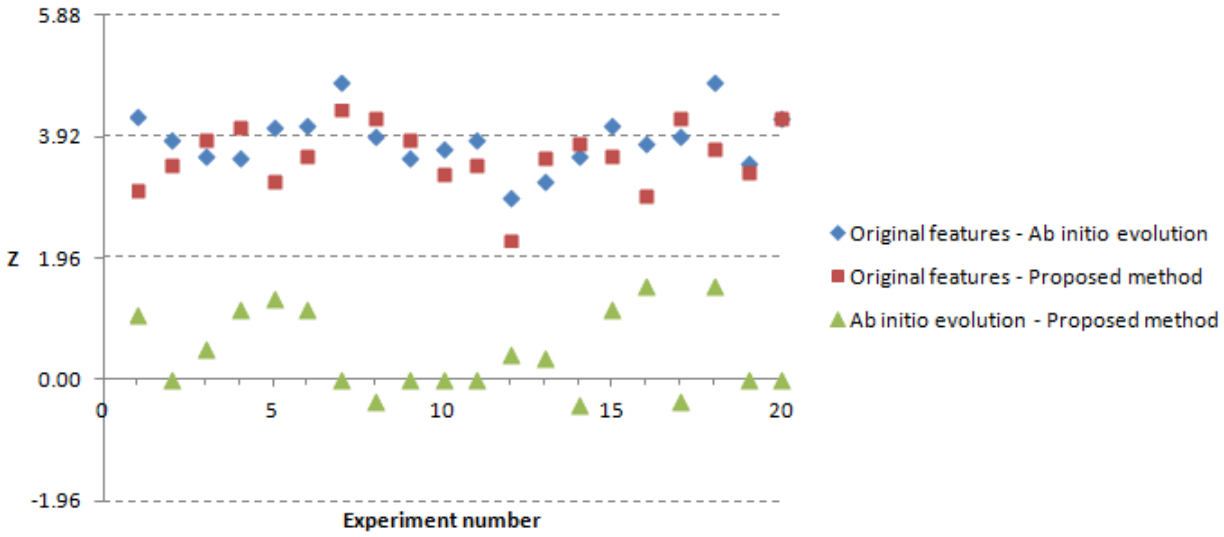


Figure 6.3: Illustration on Z scores of the experiment on lime maturity grading as regards adding new features

The Z scores are shown in Table 6.4 and illustrated in Figure 6.3. All Z values of the *ab initio* and the proposed techniques compared with the original classifiers are in the range 2.9–4.8, significantly higher than 1.96. In contrast, comparison between the *ab initio* and the proposed approaches generated all Z scores less than the critical value, which indicates that differences between the *ab initio* evolution and the proposed technique are insignificant while the performance of them compared with performing by the original program differed distinctly in a statistical sense.

6.2.3 Determining the maturity and size of cherry tomatoes

Cherry tomatoes can be harvested when they are green and have proper size. The colour then changes to orange and red with time. Cherry tomatoes are graded based on the maturity and size according to customer demands. Sample

Table 6.3: Classification results of the experiment on lime maturity grading as regards adding new features

Experiment	Classification accuracy results (%)		
number	Original features	*Ab initio* evolution	Proposed method
1	62.9	95.2	88.7
2	61.3	93.5	91.9
3	62.9	91.9	95.2
4	51.6	85.5	91.9
5	56.5	93.5	85.5
6	59.7	95.2	90.3
7	56.5	96.8	95.2
8	56.5	93.5	93.5
9	59.7	90.3	91.9
10	62.9	93.5	91.9
11	61.3	93.5	91.9
12	75.8	95.2	91.9
13	46.8	85.5	90.3
14	62.9	91.9	91.9
15	59.7	95.2	90.3
16	58.1	93.5	85.5
17	56.5	93.5	93.5
18	54.8	96.8	88.7
19	62.9	90.3	88.7
20	53.2	90.3	90.3
Best	75.8	96.8	95.2
Mean	59.1	92.7	91.0
s.d.	5.9	3.1	2.6

Table 6.4: Z scores of the experiment on lime maturity grading as regards adding new features

Experiment	Original - *Ab initio*		Original - Proposed		*Ab initio* - Proposed	
number	Z	Superior	Z	Superior	Z	Superior
1	4.25	*Ab initio*	3.06	Proposed	1.06	-
2	3.88	*Ab initio*	3.46	Proposed	0.00	-
3	3.62	*Ab initio*	3.88	Proposed	0.50	-
4	3.60	*Ab initio*	4.06	Proposed	1.15	-
5	4.09	*Ab initio*	3.21	Proposed	1.33	-
6	4.12	*Ab initio*	3.60	Proposed	1.15	-
7	4.80	*Ab initio*	4.35	Proposed	0.00	-
8	3.95	*Ab initio*	4.23	Proposed	-0.35	-
9	3.60	*Ab initio*	3.88	Proposed	0.00	-
10	3.75	*Ab initio*	3.33	Proposed	0.00	-
11	3.88	*Ab initio*	3.46	Proposed	0.00	-
12	2.94	*Ab initio*	2.25	Proposed	0.41	-
13	3.21	*Ab initio*	3.59	Proposed	0.35	-
14	3.62	*Ab initio*	3.80	Proposed	-0.41	-
15	4.12	*Ab initio*	3.60	Proposed	1.15	-
16	3.83	*Ab initio*	2.97	Proposed	1.51	-
17	3.95	*Ab initio*	4.23	Proposed	-0.35	-
18	4.80	*Ab initio*	3.73	Proposed	1.51	-
19	3.49	*Ab initio*	3.35	Proposed	0.00	-
20	4.23	*Ab initio*	4.23	Proposed	0.00	-

images were collected in four groups comprising 176 red-coloured and small, 82 red-coloured and large, 164 green-coloured and 104 orange-coloured; sample images are shown in Figure 6.7. Half the numbers of samples were randomly chosen for training and the remainder for testing.

The classification results presented in Table 6.5 demonstrate the effectiveness of incorporating the new features, clearly out-performing the original classifiers and being almost equally effective to performing *ab initio* evolution. All classification results produced by the system added with new features are higher than 98% whereas using only the original features could generate accuracy between 58–71%. The performance of the proposed technique achieved perfect classification twice, and its mean result is 98.9%, nearly 35% better than that of using the original features alone. The *ab initio* evolution was able to reach a peak accuracy of 99.6% twice, and its mean result is close to that produced by the proposed technique. The extended operators still performed well with the existing classifiers. For all sub-experiments, one sub-classifier of a bundle of the classifiers was obtained directly from the mutation process without having to subject them to further evolution.

For the Z scores presented in Table 6.6, it is apparent that all Z values produced by the proposed method and *ab initio* evolution compared with employing the original features are over 8, substantially higher than the critical value. Figure 6.4 illustrates an interesting tendency of Z scores of using original features *vs.* *ab initio* evolution and original features vs. proposed method that most values are close to each other in each sub-experiment, and some of them are equal. For example, they produced the same Z value (9.17) for sub-experiment #1 and 8.68 for sub-experiment #5. The Z values of all sub-experiments prove that additional features could significantly enhance the system performance. All sub-experiments of the

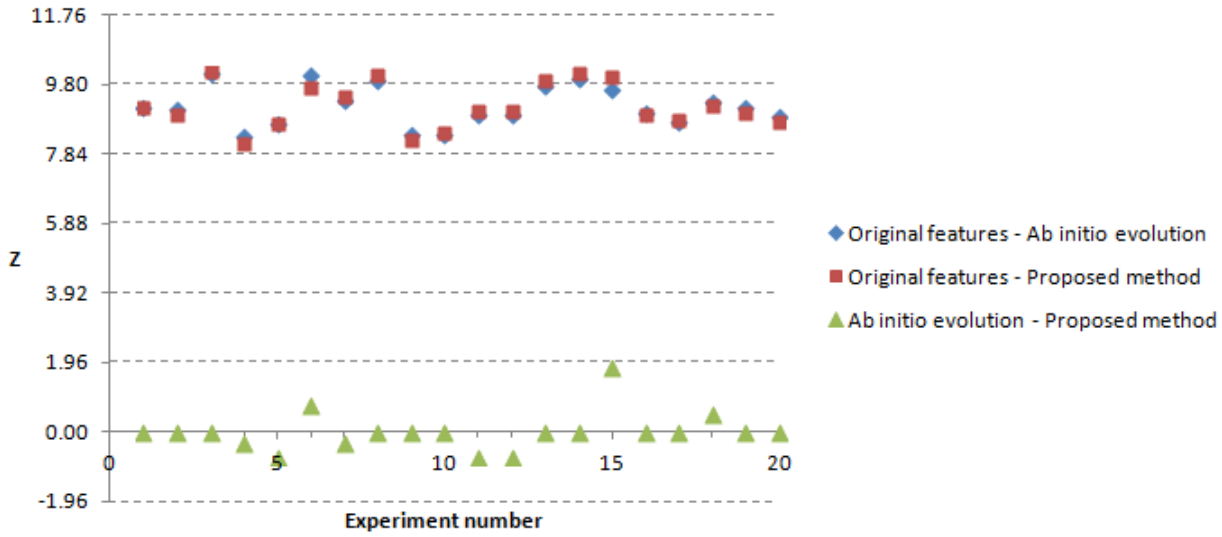


Figure 6.4: Illustration on Z scores of the experiment on cherry tomato grading as regards adding new features

proposed and *ab initio* approaches yielded Z values less than 1.96, so performance differences are insignificant.

6.2.4 Experimental conclusions of introducing new features using mutation

When further features of samples are found, a question comes up. Can new information be introduced to the system at low computational cost? Incorporating additional features using mutation was examined as a way of dealing with this issue, and three experiments were performed: identifying the variety of small oranges, classifying the maturity of limes, and determining the maturity and size of cherry tomatoes.

The experimental results demonstrate that additional features were able to enhance the system performance significantly. All classification results produced by

Table 6.5: Classification results of the experiment on cherry tomato grading as regards adding new features

Experiment	Classification accuracy results (%)		
number	Original features	*Ab initio* evolution	Proposed method
1	64.6	98.1	98.1
2	66.2	99.2	98.9
3	58.2	98.5	98.1
4	70.0	98.5	98.5
5	68.4	99.2	99.2
6	60.1	99.2	98.1
7	62.0	98.1	98.1
8	60.8	99.6	100.0
9	70.3	99.2	98.9
10	69.6	98.5	98.9
11	65.4	98.9	98.9
12	65.4	98.9	98.9
13	61.6	99.2	99.6
14	59.7	98.9	99.2
15	61.2	98.1	100.0
16	68.1	99.6	99.2
17	68.1	99.2	98.9
18	64.6	99.2	98.5
19	64.6	98.9	98.5
20	67.7	99.2	98.9
Best	70.3	99.6	100.0
Mean	64.8	99.0	98.9
s.d.	3.7	0.5	0.6

Table 6.6: Z scores of the experiment on cherry tomato grading as regards adding new features

Experiment	Original - *Ab initio*		Original - Proposed		*Ab initio* - Proposed	
number	Z	Superior	Z	Superior	Z	Superior
1	9.17	*Ab initio*	9.17	Proposed	0.00	-
2	9.12	*Ab initio*	8.96	Proposed	0.00	-
3	10.10	*Ab initio*	10.15	Proposed	0.00	-
4	8.33	*Ab initio*	8.12	Proposed	-0.35	-
5	8.68	*Ab initio*	8.68	Proposed	-0.71	-
6	10.05	*Ab initio*	9.71	Proposed	0.76	-
7	9.35	*Ab initio*	9.45	Proposed	-0.35	-
8	9.90	*Ab initio*	10.05	Proposed	0.00	-
9	8.39	*Ab initio*	8.22	Proposed	0.00	-
10	8.39	*Ab initio*	8.44	Proposed	0.00	-
11	8.97	*Ab initio*	9.07	Proposed	-0.71	-
12	8.97	*Ab initio*	9.07	Proposed	-0.71	-
13	9.75	*Ab initio*	9.90	Proposed	0.00	-
14	9.95	*Ab initio*	10.10	Proposed	0.00	-
15	9.65	*Ab initio*	10.00	Proposed	1.79	-
16	9.00	*Ab initio*	8.94	Proposed	0.00	-
17	8.73	*Ab initio*	8.78	Proposed	0.00	-
18	9.33	*Ab initio*	9.22	Proposed	0.50	-
19	9.18	*Ab initio*	9.03	Proposed	0.00	-
20	8.89	*Ab initio*	8.73	Proposed	0.00	-

(a) Sadung cultivar (b) Masui cultivar

Figure 6.5: Sample images of small oranges

(a) Appropriate-ripe (b) High-ripe (c) Overly-ripe

Figure 6.6: Sample images of limes

(a) Red-coloured and small

(b) Red-coloured and big

(c) Orange-coloured

(d) Green-coloured

Figure 6.7: Sample images of cherry tomatoes

(a) Pansetong cultivar

(b) Kimju cultivar

(c) Seedless cultivar

Figure 6.8: Sample images of guavas

(a) Paddies

(b) Brownish purple kernels

(c) Fractions

(d) Premium-grade kernels

Figure 6.9: Sample images of different qualities of purple sticky rice

(a) Barley

(b) Oats

(c) Rye

(d) Wheat

Figure 6.10: Sample images of different grain types

the *ab initio* and proposed approaches using original morphological, original grey-scaled, additional colour and additional texture features are much higher than those obtained by using only the original operators. The Z scores compared between employing only the original features and using all features are substantially higher than 1.96 for all experiments, indicating that the performance between the original system and the system extended with new features differed significantly in the statistical sense. Conversely, the Z values between the *ab initio* and proposed techniques are less than the critical value for all experiments: the proposed approach yielded roughly equally accurate systems compared with the *ab initio* evolution, and the performance differences are insignificant. In most of the cases considered, the mutation process was able to deliver programs that performed well enough on the training data that subsequent evolution was unnecessary. Furthermore, the proposed technique did not need to reevaluate the effectiveness of the existing operations because it had been done from the previous training, but the *ab initio* method did. Therefore, overall the proposed technique was considerably faster than the *ab initio* evolution.

6.3 Adding New Classes

This section moves on to examine how new classes of samples may be accommodated. This research maintained the intelligent classification system (ICS) of the original Jasmine introduced in section 3.1. ICS generates a classifier based on a binary decomposition approach. A trained multi-class vision system comprises a sequence of binary classifiers. Rather than requiring each binary classifier to distinguish one class from all others, the approach is to use the class confusion matrix resulting from the application of kNN to order the classes in terms of dif-

ficulty. The binary classifiers are then evolved in terms of increasing difficulty and form a multi-class classifier as a chain of binary classifiers. The ICS principle is illustrated in Figure 6.11. After all classes are estimated in terms of difficulty, each sub-classifier is generated and assembled in order according to class difficulty, from easier classes to harder classes.

Clearly, if a new class of samples is introduced into any classification system, it needs to be re-trained. For most approaches, this means determining class boundaries or, for GP-trained classifiers, learning new programs *ab initio*. However, for the case described in the previous paragraph in which there is a sequence of binary classifiers, it is necessary only to be able to distinguish the new class from all others, a somewhat simpler task.

An example of this idea is shown in Figure 6.12, where the classifier for distinguishing the new class is prepended to the existing sequence. Assuming that a sequence of binary classifiers has been generated based on the original Jasmine, as illustrated in Figure 6.11. When a new class of the samples is added to the system, a new sub-classifier distinguishing the new class from the existing classes is produced and prepended to the existing classifier chain to form a new multi-class classifier. Thus, when it is employed for a test set, the sub-classifier involving the new class operates before the other sub-classifiers. Following that, if the system is extended with a new class of the samples again, repeatedly a sub-classifier discriminating between the new class and a group of the previous classes is generated and inserted to perform first in the existing classifier sequence.

The proposed technique still retains the existing binary classifiers when the system is extended with a new class of samples. The system does not need to re-evolve solutions to replace the existing solutions. As every new sub-classifier of added new classes is prepended to the sequence, this idea performs differently

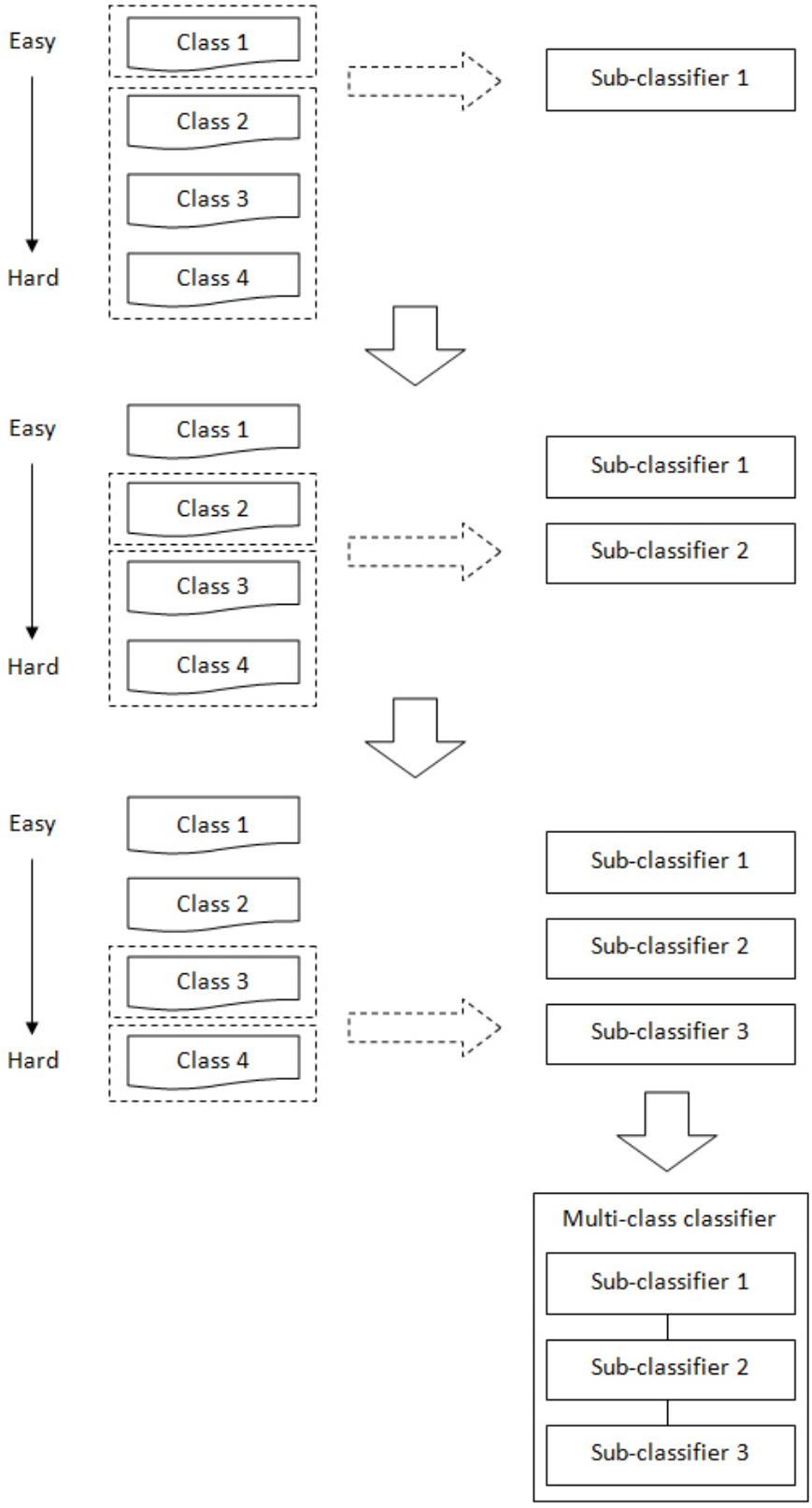


Figure 6.11: Illustration on the intelligent classification system technique

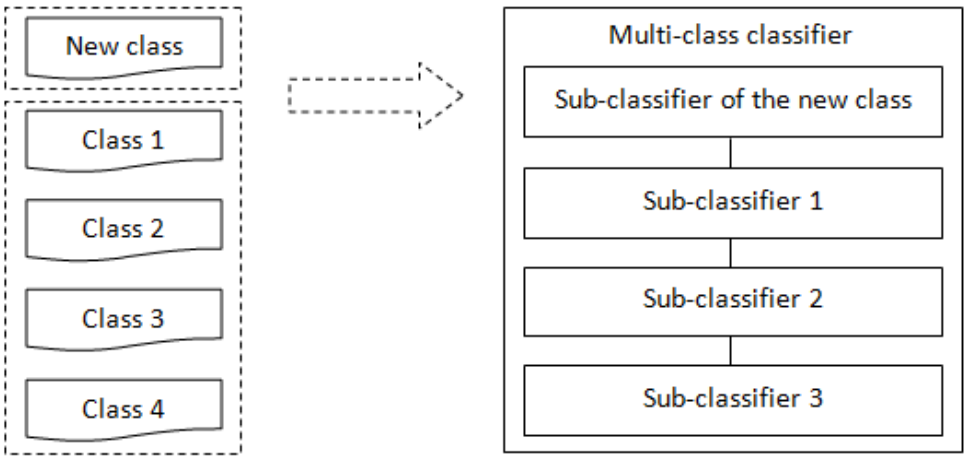


Figure 6.12: Adding a sub-classifier of a new class to a sequence of evolved binary classifiers

from ICS in which a sub-classifier of an easier class operates before one of a harder class.

6.4 Experiments on Adding a New Class of Samples

To explore the addition of a new class of samples, a binary classifier was first evolved to distinguish two classes, then a new class was added with its own binary classifier — and so on, until the last new class was introduced. The binary classifiers were then assembled to form the final multi-class classifier. Three experiments were performed: on guava cultivar classification, purple sticky rice grading and grain type identification. Twenty sub-experiments with different training data sets were carried out for each task, with half the samples selected randomly for

training processes and the remainder for testing. The accuracy results of the proposed approach are compared with the approach of evolving classifiers *ab initio*, and a comparison of the classification performance is done by using McNemar's test, and Z values are illustrated in graphs.

6.4.1 Guava type classification

This experiment aims to classify the cultivar of guavas. The samples included Pansetong (42 fruits), Kimju (48 fruits) and seedless (38 fruits) types; sample images are shown in Figure 6.8. The shape of the seedless type is not circular like the others, and all types have slightly different skin appearance. There were three groups of samples, so only one additional class was incorporated. The experiment first employed the samples of Pansetong and Kimju; then, the samples of seedless class were added.

The classification accuracy results are shown in Table 6.7. Although a classifier trained *ab initio* could reach perfect classification, its lowest accuracy (81.3%) is worse than that produced by the proposed method (82.8%). They returned the same accuracy result for sub-experiment #13. In thirteen sub-experiments, the proposed technique achieved better accuracy than performing by *ab initio* evolution. However, their mean results do not differ greatly. For the Z values shown in Figure 6.13 and Table 6.10, all values are less than 1.96, and it indicates that the performances of them do not differ in a statistical sense. For a comparison of time spent to generate a classifier (for sub-experiment #1), the *ab initio* evolution spent about 594 milliseconds, whilst the proposed technique took approximately 346 milliseconds (processing on Intel 2.27 GHz Core i5 with 8 GB of main memory under the Windows operating system). This demonstrates the potential of

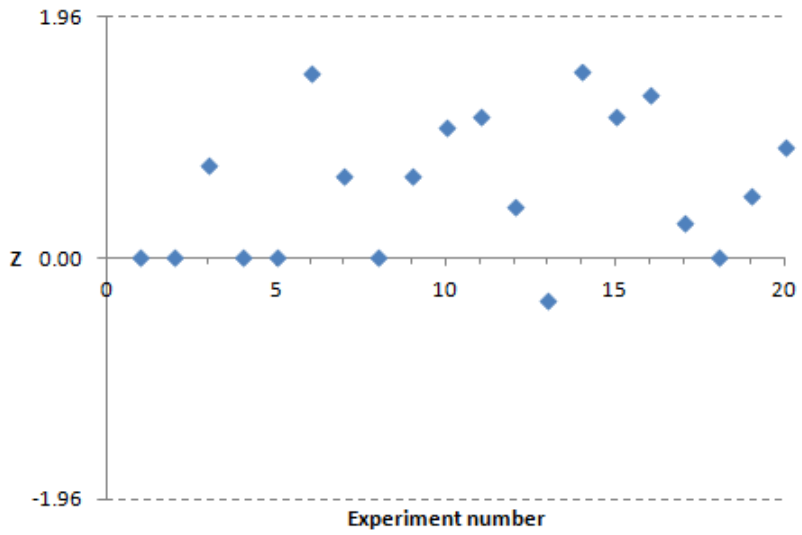


Figure 6.13: Illustration on Z scores of the experiment on guava type discrimination as regards adding new classes

the proposed technique to generate a new solution involving the existing classifier more speedily.

6.4.2 Purple sticky rice grading

After purple sticky rice is milled, they may consist of kernels with premium grade, discoloured, broken and paddies; sample pictures are shown in Figure 6.9. Before they are packed for sale, they have to be inspected (by humans) to take out the paddies. Commercially, packages of premium-grade kernels attract a higher price than the other, lower grades. However, rice grading is difficult for humans, so a computer vision solution is attractive. There were 88 images of paddies and 260 images of broken, brownish purple and pure purple kernels. A binary classifier was generated for two groups of the samples, and the two other classes were introduced one by one. For this experiment, broken kernels and paddies were first performed; after that, brownish purple and pure purple kernels were incorporated

Table 6.7: Classification results of the experiment on guava cultivar discrimination as regards adding new classes

Experiment	Classification accuracy results (%)	
number	*Ab initio* evolution	Proposed method
1	92.2	93.8
2	81.3	85.9
3	89.1	93.8
4	92.2	93.8
5	95.3	93.8
6	98.4	92.2
7	87.5	82.8
8	93.8	95.3
9	89.1	93.8
10	89.1	95.3
11	93.8	98.4
12	92.2	95.3
13	89.1	89.1
14	89.1	96.9
15	100.0	95.3
16	89.1	96.9
17	92.2	89.1
18	87.5	89.1
19	96.9	93.8
20	92.2	96.9
Best	100.0	98.4
Mean	91.5	93.1
s.d.	4.3	4.0

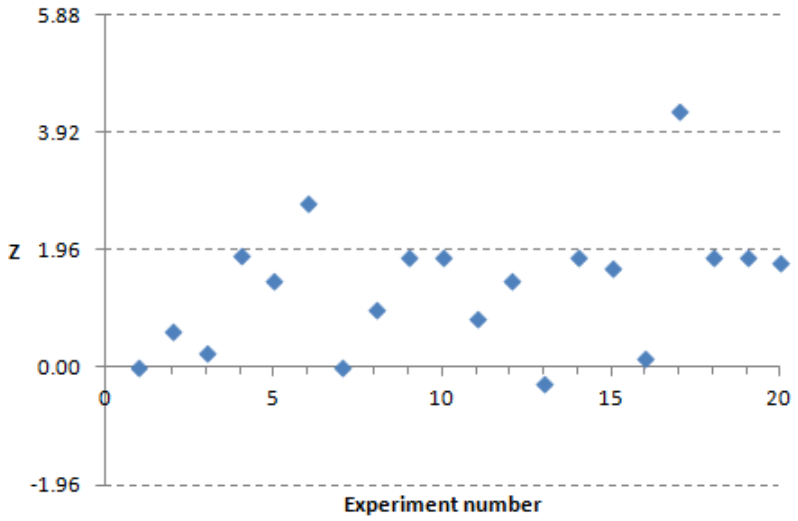


Figure 6.14: Illustration on Z scores of the experiment on purple sticky rice grading as regards adding new classes

respectively.

From the results presented in Table 6.8, the proposed method accomplished mean accuracy result of 97.3%, nearly 2% better than that achieved by *ab initio* evolution. The proposed technique was able to reach its highest accuracy of 99.1% twice, while the *ab initio* classifier reached its peak accuracy of 97.2% once. They produced the same result, 95.4%, once. The classification performance of the proposed approach exceeded the other for almost all other sub-experiments. Figure 6.14 and Table 6.10 shows that the Z scores of many sub-experiments are just under the critical value, and performance differences are significant twice.

6.4.3 Grain type identification

This experiment aims to identify types of grain including barley, oats, rye and wheat. 120 pictures of each group were employed, and they contained front and back sides of kernels. Different types of grain used in this experiment are presented

Table 6.8: Classification results of the experiment on purple sticky rice grading as regards adding new classes

Experiment	Classification accuracy results (%)	
number	*Ab initio* evolution	Proposed method
1	96.5	96.8
2	96.1	97.0
3	96.8	97.2
4	97.0	98.8
5	96.3	97.9
6	96.3	99.1
7	94.7	94.5
8	94.9	96.1
9	95.9	97.9
10	95.2	97.2
11	95.2	96.3
12	96.5	98.2
13	95.4	95.4
14	95.9	97.9
15	96.5	98.4
16	95.4	94.9
17	91.2	97.5
18	95.6	97.9
19	96.3	98.4
20	97.2	99.1
Best	97.2	99.1
Mean	95.8	97.3
s.d.	1.3	1.3

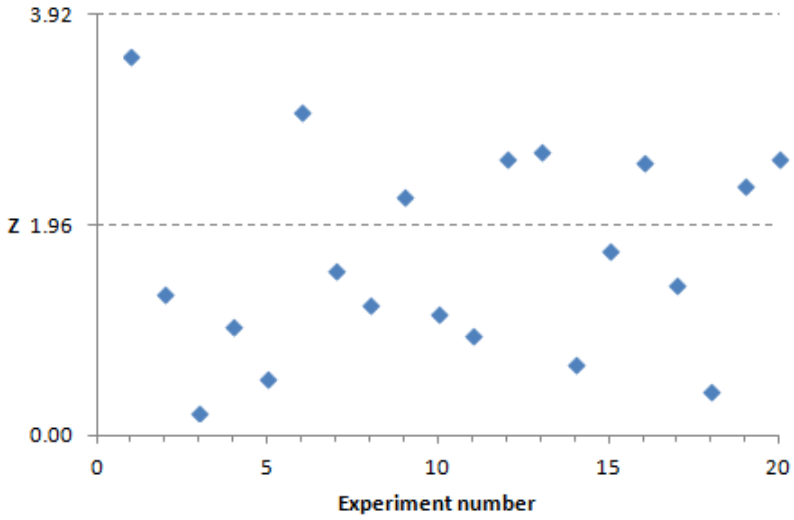


Figure 6.15: Illustration on Z scores of the experiment on grain type identification as regards adding new classes

in Figure 6.10. There were four types, so a new class was added twice. The samples of barley and oats were first employed; afterwards, rye and wheat were added in that order.

The classification accuracy results are shown in Table 6.9. Most results of the proposed method are higher than 80% whereas the *ab initio* performance was able to achieve over 80% accuracy twice, and the rest of results are in the range 72–78% accuracy. The mean result of the proposed method is 81.9%, approximately 5.5% larger than that of the *ab initio* approach. Figure 6.15 and Table 6.10 presents their Z scores. Eight sub-experiments demonstrate that the performance of the proposed method significantly differed from that performing by the *ab initio* because the Z scores are greater than 1.96. However, the other sub-experiments obtained Z scores lower than the critical value. Hence, over half the numbers of experiments indicate that their performances did not differ significantly.

Table 6.9: Classification results of the experiment on grain type identification as regards adding new classes

Experiment	Classification accuracy results (%)	
number	*Ab initio evolution*	Proposed method
1	76.7	87.5
2	74.6	79.2
3	76.7	77.9
4	76.3	80.4
5	82.1	80.0
6	72.5	82.9
7	74.2	80.0
8	77.9	82.5
9	73.3	81.7
10	77.1	81.7
11	77.1	80.4
12	74.2	83.3
13	74.6	83.3
14	84.6	82.1
15	74.6	80.4
16	76.3	85.0
17	72.5	77.9
18	78.3	80.0
19	77.9	85.8
20	76.7	85.0
Best	84.6	87.5
Mean	76.4	81.9
s.d.	3.0	2.6

Table 6.10: Z scores of all experiments as regards adding new classes

Experiment	Guava types		Purple sticky rice qualities		Grain types	
number	Z	Superior	Z	Superior	Z	Superior
1	0.00	-	0.00	-	3.54	Proposed
2	0.00	-	0.61	-	1.32	-
3	0.76	-	0.24	-	0.23	-
4	0.00	-	1.87	-	1.02	-
5	0.00	-	1.46	-	0.54	-
6	1.50	-	2.75	Proposed	3.02	Proposed
7	0.67	-	0.00	-	1.55	-
8	0.00	-	0.97	-	1.22	-
9	0.67	-	1.84	-	2.24	Proposed
10	1.06	-	1.84	-	1.14	-
11	1.15	-	0.83	-	0.94	-
12	0.41	-	1.46	-	2.58	Proposed
13	-0.35	-	-0.27	-	2.65	Proposed
14	1.51	-	1.84	-	0.68	-
15	1.15	-	1.65	-	1.74	-
16	1.33	-	0.17	-	2.56	Proposed
17	0.29	-	4.27	Proposed	1.40	-
18	0.00	-	1.84	-	0.42	-
19	0.50	-	1.84	-	2.34	Proposed
20	0.89	-	1.75	-	2.59	Proposed

6.4.4 Experimental conclusions of adding a new class of samples

In order to assess whether an already-trained system can have new classes of objects introduced, three experiments (guava cultivar classification, purple sticky rice grading, and grain type identification) were performed. Adding new classes of samples was achieved by evolving a single binary classifier, executed at the beginning of the sequence of binary classifiers, to yield overall multi-class classification.

Most sub-experiments show that the proposed technique achieved better accuracy than that of *ab initio* trained classifiers; nevertheless, their performances did not differ greatly overall. For the first experiment, on guava variety discrimination, the proposed method yielded approximately equally accurate systems compared with *ab initio* evolution, and all Z values were less than 1.96. For the experiment on purple sticky rice grading, although two sub-experiments show that the performance differences are significant and many Z scores are nearly at the critical value, about half the numbers of experiments obtained Z scores lower than the critical value. Eight times in the last experiment the performance of the proposed technique significantly exceeded that of the *ab initio* evolution while the other showed insignificant differences in performance.

Although the ICS technique that underlies Jasmine generates a binary classifier chain order according to class difficulty, from easier to harder, the proposed technique prepends a new binary classifier to the beginning of the chain; this classifier is not necessarily easier to evolve than the easiest in the existing chain. The experiments performed show that adding a new classifier by the proposed method is generally faster than *ab initio* evolution, yet yields classifiers for the complete problem that are at least commensurate in performance and occasionally better.

The author has found that, in practice, the proposed method is easier to control for producing a binary classifier of every new added class. It is like focusing on one class at a time; in contrast, ICS generates a sequence of binary classifiers at the same time. All binary classifiers are reproduced in every run of Jasmine, and an appropriate sub-classifier of each class may not be generated in the same run. The system cannot keep some sub-classifiers that perform well for another run to integrate with other sub-classifiers. The result is that using the presented technique appears to work reasonably well; thus, the answer of the question is inclined to a positive agreement. When a new class of objects is found, it can be involved effectively in the previous classifier without training from scratch.

This chapter demonstrated the ability of the GP system solving various tasks of agricultural produce grading: inspections on the maturity, qualities, varieties and size. Some tasks can be easily performed by humans, such as defect discrimination on guavas or maturity classification on limes and tomatoes. However, some inspections are difficult for humans; for example, purple sticky rice grading is a sophisticated task because of the small size, huge amount, and similarity of appearances of rice. It requires high accuracy evaluation and consistent operating — this is an important reason that grading machines are in high demand in the agricultural industry.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

This thesis aimed to develop a vision system to solve classification problems. The research extends a vision system involving genetic programming, and assesses its effectiveness in a substantial series of experiments. The author intended to make use of the program in a challenge of agricultural produce grading. Accordingly, the thesis contributions can be divided into three phases. Firstly, the system was modified to be able to process large datasets more automatically, and further feature operations extended the system to support a wide range of classification tasks. Secondly, new techniques were proposed to reconstruct 3D objects based on using a small number of cameras set in arbitrary locations; and, once reconstructed, extract properties of them. Finally, new ideas were presented to incorporate a GP solution with new features and new classes of objects. Both of these aimed to extend the capability of the system with less computation than generating a new solution from scratch.

Conclusions from these contributions are presented in section 7.1, while further research is suggested in section 7.2.

7.1 Review of Contributions

The review of the main contributions of this research is summarized in three parts, as follows.

7.1.1 Modification on Jasmine

Many stages of the original Jasmine needed interaction with the user in order to identify input or retrieve output. Some of these did not operate automatically enough to use for large-scale evaluations. Therefore, it was modified to be able to manage huge datasets and return results automatically and efficiently.

The original Jasmine provided only shape, location-based and grey-scale feature extraction for object classification. It was found that they were not sufficient to inspect agricultural produce. Consequently, the system was extended with more operations for morphological, colour and texture feature extraction so as to improve its ability to perform object classification. The details of these modifications were described in chapter 3. Experiments were done to verify the performance of the system extended with these additional features. A general type of object and some types of agricultural produce were employed for the experiments. The experimental results showed that the extended features were able to enhance the classification performance of the system significantly.

7.1.2 3D reconstruction and feature extraction

As inspection from only one view is not adequate for some types of agricutural produce, an idea was presented to determine the shape of an object in 3D. The VCFA technique was devised to estimate both the shape and appearance of a reconstructed 3D object. The reconstruction is based on employing 2D images taken by a few cameras in arbitrary locations. The generated 3D voxels were used to extract shape features. In order to analyse colour and texture, some simple rules were defined to remove duplicated areas appearing in multiple images and present object regions in only one corresponding viewpoint. VCA and OCA were proposed for colour and texture feature extraction. These contributions were presented in chapter 4.

In chapter 5, experiments were done to examine the effectiveness of these additional operators for agricultural produce grading. The results proved that using only one viewpoint, or 2D properties, was insufficient to inspect some types of agricultural produce. Characteristics extracted from 3D objects could be employed efficiently for this type of task. The performance of GP was compared with other machine learning techniques: kNN, NN and SVM, working on the same datasets. The results indicated that their performances did not differ greatly.

7.1.3 Incorporating a classifier with new features and classes

This research considered the need to re-train a multi-class classifier that has initially been evolved using genetic programming to accommodate new features or new classes. The proposed ideas were described in chapter 6. For the first approach, additional features are incorporated into a classifier by mutation. If the fitness of a mutated individual is enough to solve the task, it will be returned as

a new classifier immediately; otherwise, normal genetic programming proceeds.

Experiments indicated that most new classifiers generated by the mutation process required no further evolution; however, this is likely to be a consequence of the advantage of employing the additional operators relative to the original operators — the author does not expect this to be achievable in more general vision tasks. Nevertheless, the proposed technique yielded equally effective systems compared with *ab initio* evolution using all the features, and it proceeded faster than producing a new classifier from scratch.

For the second idea, a multi-class classifier is generated based on a binary decomposition approach. A binary classifier is evolved that is able to distinguish a new class of samples from all existing classes, and it is executed before the existing classification programs.

Most experimental results showed that performance differences between the proposed technique and the *ab initio* evolution were insignificant; however, some results indicated that the performance of the proposed technique significantly differed from that produced by *ab initio* evolution. Therefore, this idea can be used to incorporate an existing classifier with a solution of a new class of samples added in the system without reproducing a solution for existing classes.

7.2 Future Work

The contributions paved the way for the field of agricultural produce grading in order to develop a vision system to support a wide range of classification problems. Some issues should be considered for future research, and they can be listed as:

- Some types of agricultural produce do not have simple shape, so 3D volume measurement can be used for sorting, as explored in this thesis. Nonetheless,

(a) Big size (b) Medium size (c) Small size

Figure 7.1: Dragon fruit of different size

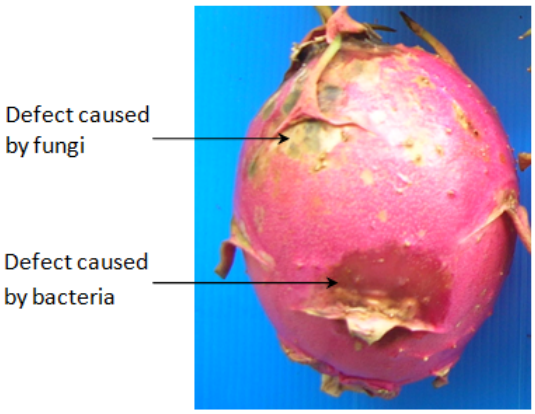


Figure 7.2: Dragon fruit skin affected by different causes

(a) Jumbo size

(b) Big size

(c) Medium size

(d) Small size

Figure 7.3: Longan bunches classified in different sizes

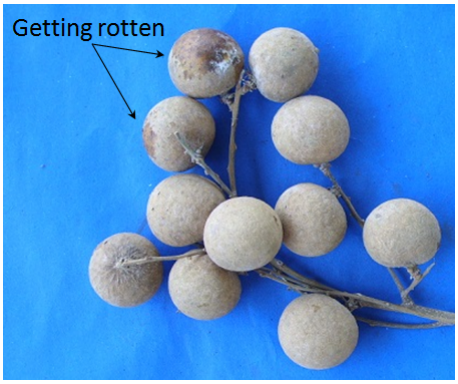


Figure 7.4: Longan bunch containing rotten fruits

some types, such as dragon fruit, have parts extended from a core body that are useless for size evaluation. Only the main body is considered for size estimation. Figure 7.1 presents dragon fruit of different size. Solving this problem will be concerned in a future task.

- Diseases have an effect on any agricultural produce, and some of these will result in characteristic appearances, even in the same fruit. Sample defective skin caused by fungi and bacteria on a dragon fruit is shown in Figure 7.2. Discrimination of diseases appearing on the same surface would be a valuable capability.

- Some types of agricultural produce are bunched, such as longans and grapes. Some of them are graded by size. Although each fruit in a bunch is not exactly the same size, in reality farmers classify the size by considering the overall appearance of the bunch. Sample images of longan bunches graded in different sizes are shown in Figure 7.3. For analysis based on images, when a bunch is shown as a picture, it would not be straightforward to discriminate the size by using only one view because fruits closer to the camera seem bigger than further ones. Some fruits overlap others or are occluded by a stem. These issues may lead to research based on 3D analysis.

- For grading by humans, each fruit has to be inspected individually, and defective or rotten fruits will be cut out. An example of a longan bunch containing rotten fruits is presented in Figure 7.4. The feature operations used in this thesis were employed to estimate defects effectively. The system can be extended to be able to identify defective fruit of bunches.

This research described in this thesis attempted to ascertain whether a single, generic vision system could be directed towards different inspection tasks in the agricultural domain automatically. The findings from all the experiments suggest that this is indeed possible — and this has great potential for the manufacturers of grading machinery.

Of course, the enhancements made to Jasmine have application well beyond agricultural grading. Furthermore, the mechanisms for incrementally extending the capability of an already-trained vision system to accommodate new features or classes may well be general enough to mimic the adaptability that the human visual system exhibits. This would be an exciting topic to explore.

- The End -

Appendices

APPENDIX A

APPENDIX

Table A.1: Standard normal distribution table (area of the right of the Z score)

Z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.50000	0.50399	0.50798	0.51197	0.51595	0.51994	0.52392	0.52790	0.53188	0.53586
0.1	0.53983	0.54380	0.54776	0.55172	0.55567	0.55962	0.56356	0.56749	0.57142	0.57535
0.2	0.57926	0.58317	0.58706	0.59095	0.59483	0.59871	0.60257	0.60642	0.61026	0.61409
0.3	0.61791	0.62172	0.62552	0.62930	0.63307	0.63683	0.64058	0.64431	0.64803	0.65173
0.4	0.65542	0.65910	0.66276	0.66640	0.67003	0.67364	0.67724	0.68082	0.68439	0.68793
0.5	0.69146	0.69497	0.69847	0.70194	0.70540	0.70884	0.71226	0.71566	0.71904	0.72240
0.6	0.72575	0.72907	0.73237	0.73565	0.73891	0.74215	0.74537	0.74857	0.75175	0.75490
0.7	0.75804	0.76115	0.76424	0.76730	0.77035	0.77337	0.77637	0.77935	0.78230	0.78524
0.8	0.78814	0.79103	0.79389	0.79673	0.79955	0.80234	0.80511	0.80785	0.81057	0.81327
0.9	0.81594	0.81859	0.82121	0.82381	0.82639	0.82894	0.83147	0.83398	0.83646	0.83891
1.0	0.84134	0.84375	0.84614	0.84849	0.85083	0.85314	0.85543	0.85769	0.85993	0.86214
1.1	0.86433	0.86650	0.86864	0.87076	0.87286	0.87493	0.87698	0.87900	0.88100	0.88298
1.2	0.88493	0.88686	0.88877	0.89065	0.89251	0.89435	0.89617	0.89796	0.89973	0.90147
1.3	0.90320	0.90490	0.90658	0.90824	0.90988	0.91149	0.91309	0.91466	0.91621	0.91774
1.4	0.91924	0.92073	0.92220	0.92364	0.92507	0.92647	0.92785	0.92922	0.93056	0.93189
1.5	0.93319	0.93448	0.93574	0.93699	0.93822	0.93943	0.94062	0.94179	0.94295	0.94408
1.6	0.94520	0.94630	0.94738	0.94845	0.94950	0.95053	0.95154	0.95254	0.95352	0.95449
1.7	0.95543	0.95637	0.95728	0.95818	0.95907	0.95994	0.96080	0.96164	0.96246	0.96327
1.8	0.96407	0.96485	0.96562	0.96638	0.96712	0.96784	0.96856	0.96926	0.96995	0.97062
1.9	0.97128	0.97193	0.97257	0.97320	0.97381	0.97441	0.97500	0.97558	0.97615	0.97670
2.0	0.97725	0.97778	0.97831	0.97882	0.97932	0.97982	0.98030	0.98077	0.98124	0.98169
2.1	0.98214	0.98257	0.98300	0.98341	0.98382	0.98422	0.98461	0.98500	0.98537	0.98574
2.2	0.98610	0.98645	0.98679	0.98713	0.98745	0.98778	0.98809	0.98840	0.98870	0.98899
2.3	0.98928	0.98956	0.98983	0.99010	0.99036	0.99061	0.99086	0.99111	0.99134	0.99158
2.4	0.99180	0.99202	0.99224	0.99245	0.99266	0.99286	0.99305	0.99324	0.99343	0.99361
2.5	0.99379	0.99396	0.99413	0.99430	0.99446	0.99461	0.99477	0.99492	0.99506	0.99520
2.6	0.99534	0.99547	0.99560	0.99573	0.99585	0.99598	0.99609	0.99621	0.99632	0.99643
2.7	0.99653	0.99664	0.99674	0.99683	0.99693	0.99702	0.99711	0.99720	0.99728	0.99736
2.8	0.99744	0.99752	0.99760	0.99767	0.99774	0.99781	0.99788	0.99795	0.99801	0.99807
2.9	0.99813	0.99819	0.99825	0.99831	0.99836	0.99841	0.99846	0.99851	0.99856	0.99861
3.0	0.99865	0.99869	0.99874	0.99878	0.99882	0.99886	0.99889	0.99893	0.99896	0.99900
3.1	0.99903	0.99906	0.99910	0.99913	0.99916	0.99918	0.99921	0.99924	0.99926	0.99929
3.2	0.99931	0.99934	0.99936	0.99938	0.99940	0.99942	0.99944	0.99946	0.99948	0.99950
3.3	0.99952	0.99953	0.99955	0.99957	0.99958	0.99960	0.99961	0.99962	0.99964	0.99965
3.4	0.99966	0.99968	0.99969	0.99970	0.99971	0.99972	0.99973	0.99974	0.99975	0.99976
3.5	0.99977	0.99978	0.99978	0.99979	0.99980	0.99981	0.99981	0.99982	0.99983	0.99983
3.6	0.99984	0.99985	0.99985	0.99986	0.99986	0.99987	0.99987	0.99988	0.99988	0.99989
3.7	0.99989	0.99990	0.99990	0.99990	0.99991	0.99991	0.99992	0.99992	0.99992	0.99992
3.8	0.99993	0.99993	0.99993	0.99994	0.99994	0.99994	0.99994	0.99995	0.99995	0.99995
3.9	0.99995	0.99995	0.99996	0.99996	0.99996	0.99996	0.99996	0.99996	0.99997	0.99997

BIBLIOGRAPHY

[1] V. S. Nalwa, *A Guided Tour of Computer Vision*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1993.

[2] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Boston. MA: Pearson, 3^{rd} ed., 2010.

[3] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. New York: John Wiley & Sons, 2nd ed., 2001.

[4] A. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," *Advances in neural information processing systems*, vol. 14, p. 841, 2002.

[5] O. Oechsle, *Towards the Automatic Construction of Machine Vision Systems using Genetic Programming*. PhD thesis, University of Essex, 2009.

[6] V. H. Milan Sonka and R. Boyle, *Image Processing, Analysis and Machine Vision*. Chapman & Hall Computing, 1993.

[7] T. Gevers and A. W. Smeulders, "Color-based object recognition," *Pattern recognition*, vol. 32, no. 3, pp. 453–464, 1999.

[8] J. C. Russ, *The Image Processing Handbook (3rd Ed.)*. Boca Raton, FL, USA: CRC Press, Inc., 1999.

[9] K. R. Castleman, *Digital Image Processing*. Upper Saddle River, NJ, USA: Prentice Hall Press, 1996.

[10] "Color conversion." `http://www.rapidtables.com/convert/color/index.htm`. Accessed, Jan. 2013.

[11] X.-L. Li, Y. He, and Z.-J. Qiu, "Textural feature extraction and optimization in wavelet sub-bands for discrimination of green tea brands," in *Machine Learning and Cybernetics, 2008 International Conference on*, vol. 3, pp. 1461–1466, IEEE, 2008.

[12] F. Zhou, J.-F. Feng, and Q.-y. Shi, "Texture feature based on local fourier transform," in *Image Processing, 2001. Proceedings. 2001 International Conference on*, vol. 2, pp. 610–613, IEEE, 2001.

[13] R. M. Haralick and L. G. Shapiro, *Computer and robot vision*. Reading: Addison-Wesley, 1992.

[14] M. M. Galloway, "Texture analysis using gray level run lengths," *Computer graphics and image processing*, vol. 4, no. 2, pp. 172–179, 1975.

[15] G. A. Baxes, *Digital Image Processing: Principles and Applications*. New York, NY, USA: John Wiley & Sons, Inc., 1994.

[16] P. Jackman and D.-W. Sun, "Recent advances in image processing using image texture features for food quality assessment," *Trends in Food Science & Technology*, vol. 29, no. 1, pp. 35–43, 2013.

[17] V. Narendra and K. Hareesha, "Prospects of computer vision automated grading and sorting systems in agricultural and food products for quality evaluation," *International Journal of Computer Applications*, vol. 1, no. 4, pp. 1–9, 2010.

[18] V. Narendra and K. Hareesha, "Quality inspection and grading of agricultural and food products by computer vision-a review," *International Journal of Computer Applications*, vol. 2, no. 1, pp. 43–65, 2010.

[19] J. Blasco, S. Cubero, J. Gómez-Sanchís, P. Mira, and E. Moltó, "Development of a machine for the automatic sorting of pomegranate (punica granatum) arils based on computer vision," *Journal of Food Engineering*, vol. 90, no. 1, pp. 27–34, 2009.

[20] M. S. M. Alfatni, A. R. M. Shariff, H. Z. M. Shafri, O. M. B. Saaed, O. M. Eshanta, *et al.*, "Oil palm fruit bunch grading system using red, green and blue digital number," *Journal of Applied Sciences*, vol. 8, no. 8, pp. 1444–1452, 2008.

[21] N. Jamil, A. Mohamed, and S. Abdullah, "Automated grading of palm oil fresh fruit bunches (ffb) using neuro-fuzzy technique," in *Soft Computing and Pattern Recognition, 2009. SOCPAR'09. International Conference of*, pp. 245–249, IEEE, 2009.

[22] A. Jaffar, R. Jaafar, N. Jamil, C. Y. Low, and B. Abdullah, "Photogrammetric grading of oil palm fresh fruit bunches," *Int. J. Mech. Mechatron. Eng*, vol. 9, pp. 18–24, 2009.

[23] J. Jin, J. Li, G. Liao, X. Yu, and L. C. C. Viray, "Methodology for potatoes defects detection with computer vision," in *International Symposium on Information Processing, Huangshan*, pp. 346–351, 2009.

[24] M. Omid, M. Abbasgolipour, A. Keyhani, and S. Mohtasebi, "Implementation of an efficient image processing algorithm for grading raisins," *International Journal of Signal and Image Processing*, vol. 1, no. 1, pp. 31–34, 2010.

[25] M. Abbasgholipour, M. Omid, A. Keyhani, and S. Mohtasebi, "Color image segmentation with genetic algorithm in a raisin sorting system based on machine vision in variable conditions," *Expert Systems with Applications*, vol. 38, no. 4, pp. 3671–3678, 2011.

[26] I. S. Ahmad, J. F. Reid, M. R. Paulsen, and J. B. Sinclair, "Color classifier for symptomatic soybean seeds using image processing," *Plant disease*, vol. 83, no. 4, pp. 320–327, 1999.

[27] A. C. L. Lino, J. Sanches, and I. M. D. Fabbro, "Image processing techniques for lemons and tomatoes classification," *Bragantia*, vol. 67, pp. 785 – 789, 00 2008.

[28] Y. Gejima, H. Zhang, and M. Nagata, "Judgment on level of maturity for tomato quality using l*a*b* color image processing," in *Advanced Intelligent Mechatronics, 2003. AIM 2003. Proceedings. 2003 IEEE/ASME International Conference on*, vol. 2, pp. 1355–1359 vol.2, 2003.

[29] T. Morimoto, T. Takeuchi, H. Miyata, and Y. Hashimoto, "Pattern recognition of fruit shape based on the concept of chaos and neural networks," *Computers and Electronics in Agriculture*, vol. 26, no. 2, pp. 171–186, 2000.

[30] G. Jahns, H. Møller Nielsen, and W. Paul, "Measuring image analysis attributes and modelling fuzzy consumer aspects for tomato quality grading," *Computers and Electronics in Agriculture*, vol. 31, no. 1, pp. 17–29, 2001.

[31] M. Rashidi and K. Seyfi, "Classification of fruit shape in kiwifruit applying the analysis of outer dimensions," *Int J Agric Biol*, vol. 5, pp. 759–762, 2007.

[32] M. Rashidi, M. Gholami, *et al.*, "Determination of kiwifruit volume using ellipsoid approximation and image-processing methods," *Int. J. Agric. Biol*, vol. 10, pp. 375–380, 2008.

[33] C. Sun, M. Berman, D. Coward, and B. Osborne, "Thickness measurement and crease detection of wheat grains using stereo vision," *Pattern recognition letters*, vol. 28, no. 12, pp. 1501–1508, 2007.

[34] F. Hahn and S. Sanchez, "Carrot volume evaluation using imaging algorithms," *Journal of agricultural engineering research*, vol. 75, no. 3, pp. 243–249, 2000.

[35] S. Kerdpiboon, W. L. Kerr, and S. Devahastin, "Neural network prediction of physical property changes of dried carrot as a function of fractal dimension and moisture content," *Food research international*, vol. 39, no. 10, pp. 1110–1118, 2006.

[36] S. Riyadi, A. A. A. Rahni, M. M. Mustafa, and A. Hussain, "Shape characteristics analysis for papaya size classification," in *Research and Development, 2007. SCOReD 2007. 5th Student Conference on*, pp. 1–5, IEEE, 2007.

[37] P. Poonnoy and A. Tunsakool, "Machine for mango quality classification," *Journal of research and development KMUTT*, vol. 28, pp. 43–58, Oct. 2005.

[38] P. Atencio, T. G. Sanchez, and J. W. Branch, "Automatic visual model for classification and measurement of quality of fruit: Case mangifera indical," *Dyna rev.fac.nac.minas*, vol. 76, no. 160, pp. 317–326, 2009.

[39] P. Yimyam, S. Jaitrong, and P. Sirisomboon, "Mango maturity classification by using physical properties," *Agricultural Engineering*, pp. 12–15, Apr. 2011.

[40] M. Khojastehnazhand, M. Omid, A. Tabatabaeefar, *et al.*, "Determination of orange volume and surface area using image processing technique," *International Agrophysics*, vol. 23, no. 3, pp. 237–242, 2009.

[41] M. Beyer, R. Hahn, S. Peschel, M. Harz, and M. Knoche, "Analysing fruit shape in sweet cherry (prunus avium l.)," *Scientia Horticulturae*, vol. 96, no. 1, pp. 139–150, 2002.

[42] T. Pun, M. Lefebvre, S. Gil, D. Brunet, J.-D. Dessimoz, and P. Guegerli, "Potato operation: computer vision for agricultural robotics," in *Robotics-DL tentative*, pp. 320–331, International Society for Optics and Photonics, 1992.

[43] J. Fang, C. Zhang, and S. Wang, "Application of genetic algorithm (ga) trained artificial neural network to identify tomatoes with physiological diseases," in *Computer And Computing Technologies In Agriculture, Volume*

II (D. Li, ed.), vol. 259 of *The International Federation for Information Processing*, pp. 1103–1111, Springer US, 2008.

[44] G. Feng and C. Qixin, "Study on color image processing based intelligent fruit sorting system," in *Proc. of the 5th World Congress on Intelligent Control and Automation*, pp. 4802–4805, 2004.

[45] K. Nakano, "Application of neural networks to the color grading of apples," *Computers and electronics in agriculture*, vol. 18, no. 2, pp. 105–116, 1997.

[46] P. Vaysse, G. Grenier, O. Lavialle, G. Henry, S. Khay-Ibbat, C. Germain, and J.-P. Da Costa, "Image processing as a tool for quality assessment of fruits in bulk shipping bins," *Information and Technology for Sustainable Fruit and Vegetable Production, Frutic*, 2005.

[47] P. S. Rao, A. Gopal, R. Revathy, and K. Meenakshi, "Colour analysis of fruits using machine vision system for automatic sorting and grading," *J. Instru. Soc. India*, vol. 34, no. 4, pp. 284–291, 2004.

[48] Z. Xiaobo, Z. Jiewen, and L. Yanxiao, "Apple color grading based on organization feature parameters," *Pattern Recognition Letters*, vol. 28, pp. 2046–2053, Nov. 2007.

[49] I. Kavdir and D. E. Guyer, "Comparison of artificial neural networks and statistical classifiers in apple sorting using textural features," *Biosystems Engineering*, vol. 89, pp. 331–344, Nov. 2004.

[50] Q. Li, M. Wang, and W. Gu, "Computer vision based system for apple surface defect detection," *Computers and Electronics in Agriculture*, vol. 36, no. 2, pp. 215–223, 2002.

[51] V. Leemans and M.-F. Destain, "A real-time grading method of apples based on features extracted from defects," *Journal of Food Engineering*, vol. 61, pp. 83–89, 2004.

[52] D. Unay and B. Gosselin, "Stem and calyx recognition on jonagold apples by pattern recognition," *Journal of food Engineering*, vol. 78, no. 2, pp. 597–605, 2007.

[53] Q. Xul, X. Zou, and J. Zhao, "On-line detection of defects on fruit by machinevision systems based on three-color-cameras systems," in *Computer and Computing Technologies in Agriculture II, Volume 3*, pp. 2231–2238, Springer, 2009.

[54] Z. yuan Wen, L. ming Shen, H. ping Jing, and K. Fang, "Color and shape grading of citrus fruit based on machine vision with fractal dimension," in *Image and Signal Processing (CISP), 2010 3rd International Congress on*, vol. 2, pp. 898–903, 2010.

[55] F. Lopez-Garcia, G. Andreu-Garcia, J. Blasco, N. Aleixos, and J.-M. Valiente, "Automatic detection of skin defects in citrus fruits using a multivariate image analysis approach," *Computers and Electronics in Agriculture*, vol. 71, no. 2, pp. 189 – 197, 2010.

[56] N. Kondo, U. Ahmad, M. Monta, and H. Murase, "Machine vision based quality evaluation of iyokan orange fruit using neural networks," *Computers and Electronics in Agriculture*, vol. 29, no. 1, pp. 135–147, 2000.

[57] L. Xu, "Strawberry maturity neural network detectng system based on genetic algorithm," in *Computer and Computing Technologies in Agriculture II, Volume 2*, pp. 1201–1208, Springer, 2009.

[58] D. G. Kim, T. F. Burks, J. Qin, and D. M. Bulanon, "Classification of grapefruit peel diseases using color texture feature analysis," *International Journal of Agricultural and Biological Engineering*, vol. 2, no. 3, pp. 41–50, 2009.

[59] Z. Liu, F. Cheng, Y. Ying, and X. Rao, "Identification of rice seed varieties using neural network," *Journal of Zhejiang University SCIENCE*, vol. 6B, pp. 1095–1100, Nov. 2005.

[60] M. Khojastehnazhand, M. Omid, and A. Tabatabaeefar, "Development of a lemon sorting system based on color and size," *African Journal of Plant Science*, vol. 4, no. 4, pp. 122–127, 2010.

[61] T. Chalidabhongse, P. Yimyam, and P. Sirisomboon, "2d/3d vision-based mango's feature extraction and sorting," in *Control, Automation, Robotics and Vision, 2006. ICARCV '06. 9th International Conference on*, pp. 1–6, 2006.

[62] M. Shahin and S. Symons, "Lentil type identification using machine vision," *Canadian Biosystems Engineering*, vol. 45, pp. 3–5, 2003.

[63] P. Menesatti, C. Costa, G. Paglia, F. Pallottino, S. D'Andrea, V. Rimatori, and J. Aguzzi, "Shape-based methodology for multivariate discrimination among italian hazelnut cultivars," *Biosystems Engineering*, vol. 101, no. 4, pp. 417–424, 2008.

[64] J. Paliwal, N. S. Visen, and D. S. Jayas, "Evaluation of neural network architectures for cereal grain classification," *Journal of Agricultural Engineering Research*, vol. 79, pp. 361–370, Aug. 2001.

[65] J. Paliwal, N. S. Visen, D. S. Jayas, and N. D. G. White, "Cereal grain and dockage identification using machine vision," *Biosystems Engineering*, vol. 85, pp. 51–57, May 2003.

[66] J. Paliwal, N. S. Visen, D. S. Jayas, and N. D. G. White, "Comparison of a neural network and a non-parametric classifier for grain kernel identification," *Biosystems Engineering*, vol. 85, pp. 405–413, Aug. 2003.

[67] N. S. Visen, J. Paliwal, D. S. Jayas, and N. D. G. White, "Image analysis of bulk grain samples using neural networks," *Canadian Biosystems Engineering*, vol. 46, pp. 11–15, 2004.

[68] S. Shantaiya and U. Ansari, "Identification of food grains and its quality using pattern classification," in *International Conference [ICCT-2010], Special Issue of IJCCT*, vol. 2, (Rochester, N.Y.), pp. 70–74, 2010.

[69] A. Douik and M. Abdellaoui, "Cereal varieties classification using wavelet techniques combined to multi-layer neural networks," in *16th Mediterranean Conference on Control and Automation Congress Centre*, pp. 1822–1827, 2008.

[70] I. Y. Zayas, C. Martin, J. Steele, and A. Katsevich, "Wheat classification using image analysis and crush-force parameters," *Transactions of the ASAE-American Society of Agricultural Engineers*, vol. 39, no. 6, pp. 2199–2204, 1996.

[71] A. Douik and M. Abdellaoui, "Cereal grain classification by optimal features and intelligent classifiers cereal image acquisition system classification features," *Int. J. of Computers, Communications & Control*, vol. V, no. 4, pp. 506–516, 2010.

[72] S. Majurndar and D. S. Jayas, "Classification of cereal grains using machine vision," Tech. Rep. 6, American Society of Agricultural Engineers, 2000.

[73] A. Mehrez, D. Ali, and D. G. Electrique, "Hybrid method for cereal grain identification using morphological and color features," in *13th IEEE International Conference on Electronics, Circuits and Systems*, pp. 870–873, 2006.

[74] T. Pearson, D. Brabec, and S. Haley, "Color image based sorter for separating red and white wheat," *Sensing and Instrumentation for Food Quality and Safety*, vol. 2, pp. 280–288, Oct. 2008.

[75] S. Majumdar, *Classification of cereal grains using machine vision*. PhD thesis, University of Manitoba, 1997.

[76] D. Guyer and X. Yang, "Use of genetic artificial neural networks and spectral imaging for defect detection on cherries," *Computers and Electronics in Agriculture*, vol. 29, no. 3, pp. 179–194, 2000.

[77] M. Shahin and S. Symons, "A machine vision system for grading lentils," *Canadian Biosystems Engineering*, vol. 43, pp. 7–7, 2001.

[78] A. Mahmoudi, M. Omid, A. Aghagolzadeh, and A. Borgayee, "Grading of iranian's export pistachio nuts based on artificial neural networks," *International Journal of Agriculture and Biology (Pakistan)*, 2006.

[79] H. Zheng, H. Lu, Y. Zheng, H. Lou, and C. Chen, "Automatic sorting of chinese jujube (zizyphus jujuba mill. cv. hongxing) using chlorophyll fluorescence and support vector machine," *Journal of food engineering*, vol. 101, no. 4, pp. 402–408, 2010.

[80] B. K. Yadav and V. K. Jindal, "Monitoring milling quality of rice by image analysis," *Computers and Electronics in Agriculture*, vol. 33, pp. 19–33, Dec. 2001.

[81] F. Cheng and Y. Ying, "Machine vision inspection of rice seed based on hough transform," *Journal of Zhejiang University SCIENCE*, vol. 5, pp. 663–667, June 2004.

[82] J. D. Guzman and E. K. Peralta, "Classification of philippine rice grains using machine vision and artificial neural networks," in *World conference on agricultural information and IT*, 2008.

[83] H. Gao, Y. Wang, and P. Ge, "Rice shape parameter detection based on image processing," in *CCTA'07*, pp. 287–294, 2008.

[84] A. Ouyang, R. Gao, Y. Liu, and X. Dong, "An automatic method for identifying different variety of rice seeds using machine vision technology," in *2010 Sixth International Conference on Natural Computation (ICNC 2010)*, pp. 84–88, 2010.

[85] I. Paulus and E. Schrevens, "Shape characterization of new apple cultivars by fourier expansion of digitized images," *Journal of Agricultural Engineering Research*, vol. 72, pp. 113–118, 1999.

[86] A. J. Currie, S. Ganeshanandam, D. A. Noiton, D. Garrick, C. J. A. Shelbourne, and N. Oraguzie, "Quantitative evaluation of apple (malus x domestica borkh.) fruit shape by principal component analysis of fourier descriptors," *Euphytica*, vol. 111, pp. 219–227, 2000.

[87] D. Zhang, G. Lu, *et al.*, "A comparative study on shape retrieval using fourier descriptors with different shape signatures," in *Proc. of international conference on intelligent multimedia and distance education (ICIMADE01)*, pp. 1–9, 2001.

[88] S. Osowski and D. D. Nghia, "Fourier and wavelet descriptors for shape recognition using neural networks: a comparative study," *Pattern Recognition*, vol. 35, no. 9, pp. 1949–1957, 2002.

[89] J. R. Koza, F. Bennett, D. Andre, and M. A. Keane, "Genetic programming iii: darwinian invention and problem solving [book review]," *Evolutionary Computation, IEEE Transactions on*, vol. 3, no. 3, pp. 251–253, 1999.

[90] S. Luke, *Essentials of Metaheuristics*. Lulu, 2^{nd} ed., 2013. Available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/.

[91] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*, vol. 1. MIT press, 1992.

[92] R. Poli, W. W. B. Langdon, N. F. McPhee, and J. R. Koza, *A Field Guide to Genetic Programming*. Lulu. com, 2008.

[93] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," *Foundations of genetic algorithms*, vol. 1, pp. 69–93, 1991.

[94] S. Silva, S. Dignum, and L. Vanneschi, "Operator equalisation for bloat free genetic programming and a survey of bloat control methods," *Genetic Programming and Evolvable Machines*, vol. 13, no. 2, pp. 197–238, 2012.

[95] P. G. Espejo, S. Ventura, and F. Herrera, "A survey on the application of genetic programming to classification," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 40, no. 2, pp. 121–144, 2010.

[96] O. Oechsle and A. F. Clark, "Feature extraction and classification by genetic programming," in *Proceedings of the International Conference on Vision Systems*, pp. 131–140, 2008.

[97] H. Guo and A. Nandi, "Breast cancer diagnosis using genetic programming generated feature," in *Machine Learning for Signal Processing, 2005 IEEE Workshop on*, pp. 215–220, 2005.

[98] J.-H. Hong and S.-B. Cho, "Lymphoma cancer classification using genetic programming with snr features," in *Genetic Programming*, pp. 78–88, Springer, 2004.

[99] S. Hengpraprohm and P. Chongstitvatana, "A genetic programming ensemble approach to cancer microarray data classification," in *Innovative Computing Information and Control, 2008. ICICIC '08. 3rd International Conference on*, pp. 340–340, 2008.

[100] R. Poli, "Genetic programming for image analysis," in *Proceedings of the First Annual Conference on Genetic Programming*, pp. 363–368, MIT Press, 1996.

[101] K.-H. Liu and C.-G. Xu, "A genetic programming-based approach to the classification of multiclass microarray datasets," *Bioinformatics*, vol. 25, no. 3, pp. 331–337, 2009.

[102] Y. Zhang and P. I. Rockett, "Feature extraction using multi-objective genetic programming," in *Multi-objective machine learning*, pp. 75–99, Springer, 2006.

[103] Y. Zhang and P. I. Rockett, "Evolving optimal feature extraction using multi-objective genetic programming: a methodology and preliminary study on edge detection," in *Proceedings of the 7th annual conference on Genetic and Evolutionary Computation*, pp. 795–802, ACM, 2005.

[104] C. Harris and B. Buxton, "Evolving edge detectors with genetic programming," in *Proceedings of the First Annual Conference on Genetic Programming*, pp. 309–314, MIT Press, 1996.

[105] S. Cagnoni, E. Lutton, and G. Olague, *Genetic and evolutionary computation for image processing and analysis*, vol. 8. Hindawi Publishing Corporation, 2007.

[106] B. Lam and V. Ciesielski, "Discovery of human-competitive image texture feature extraction programs using genetic programming," in *Genetic and Evolutionary Computation–GECCO 2004*, pp. 1114–1125, Springer, 2004.

[107] P. K. Spivak, "Discovery of optical character recognition algorithms using genetic programming," *Genetic Algorithms and Genetic Programming at Stanford*, pp. 223–232, 2002.

[108] F. A. Alsulaiman, N. Sakr, J. Valde, A. El Saddik, and N. D. Georganas, "Feature selection and classification in genetic programming: Application to haptic-based biometric data," in *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, pp. 1–7, IEEE, 2009.

[109] J. Lin, H. Ke, B. Chien, and W. Yang, "Classifier design with feature selection and feature extraction using layered genetic programming," *Expert Systems with Applications*, vol. 34, pp. 1384–1393, Feb. 2008.

[110] M. G. Smith and L. Bull, "Feature construction and selection using genetic programming and a genetic algorithm," in *Genetic Programming*, pp. 229–237, Springer, 2003.

[111] M. G. Smith and L. Bull, "Genetic programming with a genetic algorithm for feature construction and selection," *Genetic Programming and Evolvable Machines*, vol. 6, no. 3, pp. 265–281, 2005.

[112] J. J. Szymanski, S. P. Brumby, P. A. Pope, D. R. Eads, D. M. Esch-Mosher, M. C. Galassi, N. R. Harvey, H. D. McCulloch, S. J. Perkins, R. B. Porter, *et al.*, "Feature extraction from multiple data sources using genetic programming," in *AeroSense 2002*, pp. 338–345, International Society for Optics and Photonics, 2002.

[113] H. Guo, L. B. Jack, and A. K. Nandi, "Feature generation using genetic programming with application to fault classification," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 35, no. 1, pp. 89–99, 2005.

[114] Y. Zhang and P. I. Rockett, "A generic optimising feature extraction method using multiobjective genetic programming," *Applied Soft Computing*, vol. 11, no. 1, pp. 1087–1097, 2011.

[115] R. K. Rao, K. Tun, and S. Lakshminarayanan, "Genetic programming based variable interaction models for classification of process and biological systems," *Industrial & Engineering Chemistry Research*, vol. 48, no. 10, pp. 4899–4907, 2009.

[116] W. Smart and M. Zhang, "Using genetic programming for multiclass classification by simultaneously solving component binary classification problems," in *Genetic Programming*, pp. 227–239, Springer, 2005.

[117] M. Zhang and M. Lett, "Genetic programming for object detection: Improving fitness functions and optimising training data," *The IEEE Intelligent Informatics Bulletin*, vol. 7, pp. 12–21, Dec. 2006.

[118] J. Li, X. Li, and X. Yao, "Cost-sensitive classification with genetic programming," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 3, pp. 2114–2121 Vol. 3, 2005.

[119] C. Fogelberg and M. Zhang, "Linear genetic programming for multi-class object classification," in *AI 2005: Advances in artificial intelligence*, pp. 369–379, Springer, 2005.

[120] K. Badran and P. I. Rockett, "The roles of diversity preservation and mutation in preventing population collapse in multiobjective genetic programming," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pp. 1551–1558, ACM, 2007.

[121] N. Al-Madi and S. A. Ludwig, "Improving genetic programming classification for binary and multiclass datasets," in *Computational Intelligence and Data Mining (CIDM), 2013 IEEE Symposium on*, pp. 166–173, IEEE, 2013.

[122] M. C. Bot, "Feature extraction for the k-nearest neighbour classifier with genetic programming," in *Genetic Programming*, pp. 256–267, Springer, 2001.

[123] L. Liu, L. Shao, and P. Rockett, "Genetic programming-evolved spatio-temporal descriptor for human action recognition.," in *BMVC*, pp. 1–12, 2012.

[124] W. A. Tackett, "Genetic programming for feature discovery and image discrimination," in *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 303–309, 1993.

[125] A. Song, T. Loveard, and V. Ciesielski, "Towards genetic programming for texture classification," in *AI 2001: Advances in Artificial Intelligence*, vol. 2256, pp. 557–573, 2001.

[126] T. Loveard and V. Ciesielski, "Representing classification problems in genetic programming," *Evolutionary Computation*, vol. 2, pp. 1070–1077, 2001.

[127] C. Blake and C. J. Merz, "UCI repository of machine learning databases." `https://archive.ics.uci.edu/ml/datasets.html`, 1998.

[128] W. R. Smart and M. Zhang, "Classification strategies for image classification in genetic programming," *Image and Vision Computing NZ*, pp. 402–407, 2003.

[129] M. Zhang and W. Smart, "Multiclass object classification using genetic programming," in *Applications of Evolutionary Computing*, pp. 369–378, Springer, 2004.

[130] D. P. Muni, N. R. Pal, and J. Das, "A novel approach to design classifiers using genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 8, pp. 183–196, Apr. 2004.

[131] L. Zhang. and A. K. Nandi, "Fault classification using genetic programming," *Mechanical Systems and Signal Processing*, vol. 21, no. 3, pp. 1273 – 1284, 2007.

[132] Y. Zhang and P. Rockett, "Domain-independent feature extraction for multi-classification using multi-objective genetic programming," *Pattern Analysis & Applications*, vol. 13, no. 3, pp. 273–288, 2010.

[133] N. Kanwal, *Low-level image features and navigation systems for visually impaired people*. PhD thesis, University of Essex, 2013.

[134] N. Kanwal, S. Ehsan, E. Bostanci, and A. Clark, "A statistical approach for comparing the performances of corner detectors," in *Communications, Computers and Signal Processing (PacRim), 2011 IEEE Pacific Rim Conference on*, pp. 321–326, Aug 2011.

[135] N. Kanwal, S. Ehsan, E. Bostanci, and A. Clark, "Evaluating the angular sensitivity of corner detectors," in *Virtual Environments Human-Computer Interfaces and Measurement Systems (VECIMS), 2011 IEEE International Conference on*, pp. 1–4, Sept 2011.

[136] M. R. Spiegel and J. J. Stephens, *Statistics*. McGraw-Hill, 3rd ed., 1999.

[137] T. D. Miethe and J. F. Gauthier, *Simple Statistics*. Oxford University, 2008.

[138] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[139] M. Evans, N. Hastings, and B. Peacock, *Statistical Distributions*. New York: Wiley, 3rd ed., 2000.

[140] M. Goldman, "Statistics for bioinformatics." `http://www.stat.berkeley.edu/~mgoldman/Section0402.pdf`. Accessed, Jan. 2014.

[141] M. A. Napierala, "What is the bonferroni correction." `http://www.aaos.org/news/aaosnow/apr12/research7.asp`. Accessed, Jan. 2014.

[142] P. S. Mann, *Introductory statistics.* John Wiley & Sons, 4th ed., 2001.

[143] J. P. Shaffer, "Multiple hypothesis testing," *Annual review of psychology*, vol. 46, no. 1, pp. 561–584, 1995.

[144] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian journal of statistics*, pp. 65–70, 1979.

[145] Y. Hochberg, "A sharper bonferroni procedure for multiple tests of significance," *Biometrika*, vol. 75, no. 4, pp. 800–802, 1988.

[146] Y. Benjamini and Y. Hochberg, "Controlling the false discovery rate: a practical and powerful approach to multiple testing," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 289–300, 1995.

[147] R. Kohavi and F. Provost, "Glossary of terms," *Machine Learning*, vol. 30, no. 2-3, pp. 271–274, 1998.

[148] S. S. Klanarong Silanam, Narongsak Sanlamun and W. Kaensup, "Mango sorter machine." `http://www.kmutt.ac.th/rippc/best35e.htm`. Accessed, Jan. 2014.

[149] R. Jain, R. Kasturi, and B. G. Schunck, *Machine Vision*, vol. 5. McGraw-Hill New York, 1995.

[150] G. Babis, "Computer vision." `http://www.cse.unr.edu/~bebis/CS791E/`. Accessed, Sep. 2012.

[151] S. M. Seitz and C. R. Dyer, "Photorealistic scene reconstruction by voxel coloring," in *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, CVPR '97, (Washington, DC, USA), pp. 1067–, IEEE Computer Society, 1997.

[152] S. Seitz and C. Dyer, "Photorealistic scene reconstruction by voxel coloring," *International Journal of Computer Vision*, vol. 35, no. 2, pp. 151–173, 1999.

[153] A. W. Fitzgibbon, G. Cross, and A. Zisserman, "Automatic 3d model construction for turn-table sequences," in *3D Structure from Multiple Images of Large-Scale Environments*, pp. 155–170, Springer, 1998.

[154] A. Ladikos, S. Benhimane, and N. Navab, "Efficient visual hull computation for real-time 3d reconstruction using cuda," in *Proceedings of the 2008 Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1–8, 2008.

[155] T. Wada, X. Wu, S. Tokai, and T. Matsuyama, "Homography based parallel volume intersection: toward real-time volume reconstruction using active cameras," in *Computer Architectures for Machine Perception, 2000. Proceedings. Fifth IEEE International Workshop on*, pp. 331–339, 2000.

[156] G. Cheung, T. Kanade, J. Y. Bouguet, and M. Holler, "A real time system for robust 3d voxel reconstruction of human motions," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 2, pp. 714–720, 2000.

[157] R. Kehl, M. Bray, and L. Van Gool, "Full body tracking from multiple views using stochastic sampling," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, pp. 129–136, IEEE, 2005.

[158] J. C. Noordam, G. W. Otten, T. J. Timmermans, and B. H. van Zwol, "High-speed potato grading and quality inspection based on a color vision system," in *Electronic imaging*, pp. 206–217, International Society for Optics and Photonics, 2000.

[159] J.-Y. Bouguet, "Camera calibration toolbox for Matlab." `http://www.vision.caltech.edu/bouguetj/calib_doc/`. Accessed, Sep. 2012.

[160] J.-Y. Bouguet, "Camera calibration toolbox for Matlab." `http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/parameters.html`. Accessed, Sep. 2012.

[161] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *Computer vision and pattern recognition, 2006 IEEE Computer Society Conference on*, vol. 1, pp. 519–528, IEEE, 2006.

[162] A. Laurentini, "The visual hull concept for silhouette-based image understanding," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 16, no. 2, pp. 150–162, 1994.

[163] W. N. Martin and J. Aggarwal, "Volumetric descriptions of objects from multiple views," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-5, pp. 150–158, March 1983.

[164] W. Matusik, C. Buehler, and L. McMillan, *Polyhedral visual hulls for real-time rendering.* Springer, 2001.

[165] S. K. Srivastava and N. Ahuja, "Octree generation from object silhouettes in perspective views," *Computer Vision, Graphics, and Image Processing*, vol. 49, no. 1, pp. 68–84, 1990.

[166] E. Boyer, "Object models from contour sequences," in *Computer Vision: ECCV'96*, pp. 109–118, Springer, 1996.

[167] Y. Kuzu and V. Rodehorst, "Volumetric modeling using shape from silhouette," *Proc. the 4th Turkish-German Joint Geodetic Days*, pp. 469–476, 2001.

[168] K. N. Kutulakos and S. M. Seitz, "What do photographs tell us about 3D shape?," tech. rep., Technical Report TR692, Computer Science Dept., U. Rochester, 1998.

[169] K. N. Kutulakos and S. M. Seitz, "A theory of shape by space carving," *Int. J. Comput. Vision*, vol. 38, pp. 199–218, July 2000.

[170] W. B. Culbertson, T. Malzbender, and G. Slabaugh, "Generalized voxel coloring," in *Vision Algorithms: Theory and Practice*, pp. 100–115, Springer, 2000.

[171] C. Leung, B. Appleton, and C. Sun, "Embedded voxel colouring," in *Digital Image Computing - Techniques and Applications*, pp. 623–632, 2003.

[172] P. de Oude, "Two color consistency checks for voxel coloring." `http://staff.science.uva.nl/~bredeweg/pdf/BSc/20032004/deOude.pdf`, 2004.

[173] O. Ozüna, U. Yılmazb, and V. Atalaya, "Comparison of photoconsistency measures used in voxel coloring," *XXXVI*, p. 51, 2005.

[174] C. Leung, B. Appleton, M. Buckley, and C. Sun, "Embedded voxel colouring with adaptive threshold selection using globally minimal surfaces," *Int. J. Comput. Vision*, vol. 99, pp. 215–231, Sept. 2012.

[175] O. Batchelor, R. Mukundan, and R. Green, "Incremental voxel colouring by ray traversal," in *Computer Graphics, Imaging and Visualisation, 2006 International Conference on*, pp. 396–401, IEEE, 2006.

[176] A. O. Balan, "Voxel carving and coloring-constructing a 3D model of an object from 2D images." `http://cs.brown.edu/~alb/papers/balan03voxel.pdf`. Accessed, Sep. 2012.

[177] B. Goldluecke and M. Magnor, "Space-time isosurface evolution for temporally coherent 3d reconstruction," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1, pp. I–350, IEEE, 2004.

[178] K. SusheelKumar, V. B. Semwal, S. Prasad, and R. Tripathi, "Generating 3d model using 2d images of an object," *International Journal of Engineering Science*, 2011.

[179] P. Yimyam and A. F. Clark, "Adding new features and classes to classifiers evolved using genetic programming," in *Electronics, Computer and Computation (ICECCO), 2013 International Conference on*, pp. 224–227, IEEE, 2013.

[180] S. Luke, *Issues in Scaling Genetic Programming: Breeding Strategies, Tree Generation, and Code Bloat*. PhD thesis, Department of Computer Science, University of Maryland, A. V. Williams Building, University of Maryland, College Park, MD 20742 USA, 2000.

[181] W. B. Langdon and W. Banzhaf, "Repeated patterns in tree genetic programming," in *Proceedings of the 8th European Conference on Genetic Programming, volume 3447 of LNCS*, pp. 190–202, Springer, 2005.

[182] G. Slabaugh, T. Malzbender, B. Culbertson, and R. Schafer, "A survey of methods for volumetric scene reconstruction from photographs." `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.127.9206`, 2001.

[183] G. G. Slabaugh, W. B. Culbertson, T. Malzbender, M. R. Stevens, and R. W. Schafer, "Methods for volumetric reconstruction of visual scenes," *International Journal of Computer Vision*, vol. 57, no. 3, pp. 179–199, 2004.

[184] E. Töppe, M. R. Oswald, D. Cremers, and C. Rother, "Silhouette-based variational methods for single view reconstruction," in *Video Processing and Computational Video*, pp. 104–123, Springer, 2011.

[185] D. Vernon, *Machine Vision: Automated Visual Inspection and Robot Vision*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1991.

[186] N. Zuech and R. K. Miller, *Machine Vision*. London, UK, UK: Chapman & Hall, Ltd., 1989.

[187] W. B. Langdon and R. Poli, *Foundations of Genetic Programming*. Springer Publishing Company, Incorporated, 1st ed., 2010.

[188] F. Rothlauf, *Representations for Genetic and Evolutionary Algorithms*. Springer Publishing Company, Incorporated, 2nd ed., 2010.

[189] W. Banzhaf, F. D. Francone, R. E. Keller, and P. Nordin, *Genetic Programming: An Introduction: on the Automatic Evolution of Computer Programs and Its Applications*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998.

[190] P. M. Mather, *Computer Processing of Remotely-sensed Images: An Introduction*. New York, NY, USA: John Wiley & Sons, Inc., 1988.

[191] T. V. Perneger, "What's wrong with bonferroni adjustments," *BMJ: British Medical Journal*, vol. 316, no. 7139, p. 1236, 1998.

[192] J. P. A. Ioannidis, "Why most published research findings are false," *PLoS Medicine*, vol. 2, p. e124, August 2005.