

Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

Navel Orange Blemish Identification for Quality Grading System

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Computer Science
at Massey University, Albany, New Zealand

MingHui Liu

2009

Abstract

Each year, the world's top orange producers output millions of oranges for human consumption. This production is projected to grow by as much as 64 million in 2010 and so the demand for fast, low-cost and precise automated orange fruit grading systems is only deemed to become more increasingly important.

There is however an underlying limit to most orange blemish detection algorithms. Most existing statistical-based, structural-based, model-based and transform-based orange blemish detection algorithms are plagued by the following problem: any pixels in an image of an orange having about the same magnitudes for the red, green and blue channels will almost always be classified as belonging to the same category (either a blemish or not). This however presents a big problem as the RGB components of the pixels corresponding to blemishes are very similar to pixels near the boundary of an orange. In light of this problem, this research utilizes a priori knowledge of the local intensity variations observed on rounded convex objects to classify the ambiguous pixels correctly. The algorithm has the effect of peeling-off layers of the orange skin according to gradations of the intensity. Therefore, any abrupt discontinuities detected along successive layers would significantly help identifying skin blemishes more accurately. A commercial-grade fruit inspection and distribution system was used to collect 170 navel orange images. Of these images, 100 were manually classified as good oranges by human inspection and the rest are blemished ones. We demonstrate the efficacy of the algorithm using these images as the benchmarking test set. Our results show that the system garnered 96% correctly classified good oranges and 97% correctly classified blemished oranges. The proposed system is easily customizable as it does not require any training. The fruit quality bands can be adjusted to meet the requirements set by the market standards by specifying an agreeable percentage of blemishes for each band.

Acknowledgments

There are three persons I would like to thank for their contributions towards their works.

- **Dr. Napoleon Reyes (Supervisor)**
- **Dr. Andre Barczak (Co-Supervisor)**
- **Mr. Gadi Ben-Tal (Co-Supervisor), Senior Vision Engineer, Compac Sorting Equipment Limited, Auckland, New Zealand**

Thank you for introducing me this interesting topic. Thanks for giving me this opportunity to work on this project. Thanks for the time you spend with me. Many Thanks.

Finally, I will give many thanks to Massey University and all computer science staff. Thank you for providing me the opportunity and resources to work on my project.

Many thanks.

Table of Contents

Title page

Abstract

Acknowledgments

Chapter 1 Research Description	1
1.1 Overview of the Current State of Technology.....	1
1.2 Research Objectives.....	4
1.3 Research Methodology.....	4
1.4 Scope and Limitations of Research.....	5
1.5 Benchmarking Testbed.....	5
1.6 Structure of the Thesis Documentation.....	6
Chapter 2 Theoretical Framework	8
2.1 Advanced Image Processing Techniques.....	8
2.1.1 Otsu’s Method.....	8
2.1.1.1 Overview.....	8
2.1.1.2 General Algorithm.....	10
2.1.2 Canny Edge Detection.....	12
2.1.3 Morphological Operators.....	13
2.1.3.1 Overview.....	13
2.1.3.2 Convex Hull.....	15
2.2 Colour Space.....	16
2.2.1 Overview.....	16
2.2.2 RGB Colour Space.....	17
2.3 Parallel Image Processing System.....	18
2.3.1 Overview.....	18
2.3.2 System Architecture.....	19
Chapter 3 Review of Related Literature	21
3.1 High Speed Vision-Based Quality Grading of Oranges.....	21
3.1.1 Overview.....	21

3.1.2	Image Capture and Processing.....	22
3.1.3	General Algorithms.....	23
3.1.3.1	Histogram Analysis.....	23
3.1.3.2	Local Defect Search.....	24
3.1.3.3	Stem Detection.....	26
3.1.4	System Performance.....	26
3.2	Citrus Fruit External Defect Classification.....	28
3.2.1	Overview.....	28
3.2.2	Wavelet Packet Texture Analysis.....	29
3.2.3	Neural Network Classifier.....	31
3.2.4	System Performance.....	32
3.3	Intelligent Fruit Sorting System.....	34
3.3.1	Overview.....	34
3.3.2	Feature Extraction.....	36
3.3.3	Colour Ratio Judgment.....	37
3.3.4	Naive Bayes Classifier.....	39
3.3.5	System Performance.....	39
3.4	Neural Network-Based Apple Grading.....	40
3.4.1	Overview.....	40
3.4.2	Defect Segmentation.....	41
3.4.3	Neural Network Classifier.....	43
3.4.4	System Performance.....	44
Chapter 4	Preliminary Explorations.....	46
4.1	Otsu's Method.....	46
4.1.1	Experiment and Analysis.....	46
4.1.2	Algorithm Refinements.....	50
4.1.3	Summary.....	52
4.2	High Speed Vision-Based Quality Grading of Oranges.....	53
4.2.1	Histogram Analysis.....	53
4.2.1.1	Algorithm Details with Sample Computations.....	53

4.2.1.2	Neural Network Classifier.....	55
4.2.1.3	Performance.....	56
4.2.1.4	Algorithm Refinements.....	57
4.2.2	Local Defect Search.....	58
4.2.2.1	Algorithm Details with Sample Computations.....	58
4.2.2.2	Neural Network Classifier.....	59
4.2.2.3	Performance.....	59
4.2.3	Overall Quality Grading System Assessment.....	60
4.3	Citrus Fruit External Defect Classification.....	62
4.3.1	Algorithm Details with Sample Computations.....	62
4.3.2	Neural Network Classifier.....	63
4.3.3	Overall Citrus Fruit Defect Classification System Assessment.....	64
Chapter 5	Novel Algorithms on Orange Grading System.....	66
5.1	Central Thesis.....	66
5.2	System Architecture.....	67
5.3	Ripe/Unripe Orange Classification.....	68
5.4	Blemish Detection.....	69
5.4.1	Orange Colour Class.....	69
5.4.1.1	Colour Space Exploration.....	69
5.4.1.2	Derived Formula.....	71
5.4.1.3	General Algorithm.....	72
5.4.1.4	Missing Orange Colour Class.....	74
5.4.1.5	The Order of Orange Colour Classes.....	75
5.4.1.6	Summary.....	75
5.4.2	Orange Class Mean.....	76
5.4.2.1	Overview.....	76
5.4.2.2	General Algorithm.....	76
5.4.2.3	Algorithm Refinements.....	77
5.4.2.4	Effects of Illumination Intensity Variations on Ripe Orange Skin.....	78

5.4.2.5	Effects of Illumination Intensity Variations on Unripe Orange Skin.....	80
5.4.2.6	Summary.....	82
5.4.3	Orange Class Standard Deviation.....	82
5.4.3.1	Overview.....	82
5.4.3.2	General Algorithm.....	83
5.4.3.3	Summary.....	84
5.4.4	Between-Class Squared Mean Difference.....	84
5.4.4.1	Overview.....	84
5.4.4.2	General Algorithm.....	85
5.4.4.3	Summary.....	86
5.4.5	Closest Neighbor Class.....	87
5.4.5.1	Overview.....	87
5.4.5.2	Mean Selection for skewed Distributions.....	87
5.4.5.3	General Algorithm.....	90
5.4.5.4	Closest Neighbor for Ripe and Unripe Oranges.....	91
5.4.5.5	Summary.....	92
5.4.6	Class Reclassification.....	92
5.4.6.1	Overview.....	92
5.4.6.2	Colour Components and Brightness.....	93
5.4.6.3	General Algorithm.....	94
5.4.6.4	Summary.....	95
5.4.7	Pixel Reclassification.....	96
5.4.7.1	Overview.....	96
5.4.7.2	Data Analysis.....	96
5.4.7.3	General Algorithm.....	97
5.4.7.4	Summary.....	98
5.4.8	Blemish Identification.....	98
5.4.8.1	Overview.....	98
5.4.8.2	Topmost Layer Slicing.....	98

5.4.8.3	Blemish Segmentation.....	99
5.4.8.4	Refinement of Blemish Segmentation.....	99
5.4.8.5	Sample Execution of Blemish Segmentation.....	100
5.4.8.6	Experiment and Analysis on Ripe Oranges.....	102
5.4.8.7	Experiment and Analysis on Unripe Oranges.....	105
5.5	Stem Detection and Removal.....	107
5.6	Blemish Quantification.....	109
5.7	Grading.....	110
Chapter 6	Conclusion and Future Work.....	115
Appendix A.	Otsu’s Method.....	117
Appendix B.	Histogram Analysis.....	123
Appendix C.	Local Defect Search.....	125
Appendix D.	Pixel Reclassification.....	127
References.....		128

List of Figures

Fig. 1. Commercial fruit grading system.....	6
Fig. 2. Object segmentation using a constant threshold value.....	9
Fig. 3. Four and eight connectivity.....	14
Fig. 4. Expand, shrink, close and open	14
Fig. 5. Shrink a region to its skeleton.....	14
Fig. 6. Original image.....	15
Fig. 7. Convex image.....	15
Fig. 8. Sealed holes.....	16
Fig. 9. RGB and CMYK color model.....	16
Fig. 10. Example of RGB color space.....	17
Fig. 11. Colour channel splitting of a full RGB colour image.....	18
Fig. 12. Parallel image processing system.....	19
Fig. 13. Example of Zernike masks.....	26
Fig. 14. Processed orange images using high speed vision-based algorithm.....	27
Fig. 15. Citrus fruit external defect classification process.....	29
Fig. 16. Decomposition filter bank structure.....	30
Fig. 17. Neural network classifier for defect classification.....	31
Fig. 18. Colour image processing based intelligent fruit sorting system.....	34
Fig. 19. Vision inspection system in fruit sorting system.....	35
Fig. 20. Result of fruit segmentation using a constant threshold.....	36
Fig. 21. Result of noise removal using blob algorithm.....	37
Fig. 22. Interpolation-based contour detection.....	37
Fig. 23. Hue colour feature distribution.....	38
Fig. 24. Result of searching red pixels in an image.....	38
Fig. 25. BL, GR, RE, and IR filter images.....	40
Fig. 26. “Jonagold” apple samples.....	41
Fig. 27. Stem removal using neural network-based apple grading algorithm.....	42

Fig. 28. Example of segmentation using neural network.....	44
Fig. 29. Ripe orange sample for testing Otsu's method.....	46
Fig. 30. Isolated colour channels for ripe orange sample.....	46
Fig. 31. Example of ripe orange segmentation using Otsu's method.....	47
Fig. 32. Blemished ripe orange sample for testing Otsu's method.....	47
Fig. 33. Example of blemished ripe orange segmentation using Otsu's method.....	48
Fig. 34. Unripe orange sample for testing Otsu's method.....	48
Fig. 35. Isolated colour channels for unripe orange sample.....	48
Fig. 36. Example of unripe orange segmentation using Otsu's method.....	48
Fig. 37. Blemished unripe orange sample for testing Otsu's method.....	49
Fig. 38. Example of blemished unripe orange segmentation using Otsu's method.....	49
Fig. 39. Orange segmentation on the red channel using a constant threshold value fifty.....	50
Fig. 40. True and false positive rates using a constant threshold value fifty.....	51
Fig. 41. True and false positive rates using a constant threshold value seventy.....	51
Fig. 42. Example of scaled histogram.....	53
Fig. 43. Probability density at each gray level.....	54
Fig. 44. Example of image preprocessing.....	57
Fig. 45. Sample regions selected with a typical size of 30x30.....	58
Fig. 46. System workflow.....	61
Fig. 47. Sample image before and after preprocessing.....	63
Fig. 48. Example of decomposition resulting to 16 sub-windows.....	63
Fig. 49. Blemish detection based on a specified local intensity variation range.....	67
Fig. 50. Block schematic of the novel orange grading algorithm.....	67
Fig. 51. Ripe and unripe oranges.....	68
Fig. 52. Orange classification for ripe and unripe oranges.....	69
Fig. 53. A representation of additive colour mixing.....	70

Fig. 54. Colour component window in Paint.net.....	70
Fig. 55. Predefined orange colour classes.....	71
Fig. 56. Example of ripe orange image with two blue lines drawing across the center.....	78
Fig. 57. Intensity variations along the vertical blue line.....	79
Fig. 58. Two pixels selected from normal and blemished skin separately.....	79
Fig. 59. Example of unripe orange image with two red lines drawing across the center.....	80
Fig. 60. Intensity variations along the vertical red line.....	81
Fig. 61. Three pixels selected from the normal and blemished skin separately.....	81
Fig. 62. Similarity of the orange colour classes.....	87
Fig. 63. Demonstration of the closest neighbor class.....	87
Fig. 64. Clusters.....	94
Fig. 65. Image generated using the pixel reclassification matrix.....	98
Fig. 66. Segment the blemished area on the very top layer.....	99
Fig. 67. Spots in the segmented image.....	100
Fig. 68. Segmented image before and after the erode operation.....	100
Fig. 69. Blemish identification results for ripe oranges.....	100
Fig. 70. Blemish identification results for blemished ripe oranges.....	101
Fig. 71. Blemish identification results for unripe oranges.....	101
Fig. 72. Blemish identification results for blemished unripe oranges.....	102
Fig. 73. Result of image processing for a good ripe orange.....	102
Fig. 74. True positive rates computed based on one hundred good ripe oranges.....	103
Fig. 75. Result of image processing for a blemished ripe orange.....	103
Fig. 76. Blemish identification testing for blemished ripe orange.....	104
Fig. 77. True and false positive rates plotted for seventy blemished ripe oranges.....	104

Fig. 78. Blemishes identified outside of the topmost layer.....	105
Fig. 79. Some blemishes are hard to detect by human eye.....	105
Fig. 80. Example of skin colour variation on an unripe orange.....	106
Fig. 81. Result of merging the top two layers.....	106
Fig. 82. True positive rates computed based on sixty good oranges.....	106
Fig. 83. Intensity variation for blemished unripe oranges.....	107
Fig. 84. Blemish identification with different positions.....	107
Fig. 85. Intensity variation among three selected pixels.....	108
Fig. 86. Stem detection using Otsu's method.....	108
Fig. 87. Blemishes misclassified as the stem.....	108
Fig. 88. Depth of the blemish.....	108
Fig. 89. Classification results before smoothing the rough edges.....	110
Fig. 90. Classification results after smoothing the rough edges.....	111
Fig. 91. Classification results after removing the stem.....	112
Fig. 92. Classification results using the new formula.....	113
Fig. 93. Histogram analysis for Otsu's method implementation.....	118
Fig. 94. Probability analysis for Otsu's method implementation.....	118

List of Tables

Table 1. Vision-based algorithm classification result.....	27
Table 2. Citrus fruit external defect classification result.....	33
Table 3. Part of the citrus fruit classification results with images.....	34
Table 4. Classification result of intelligent fruit sorting system.....	40
Table 5. Result of classification using neural network-based grading algorithm.....	44
Table 6. Result of implementing histogram-based analysis.....	56
Table 7. Result of implementing modified histogram-based analysis.....	58
Table 8. Result of implementing modified local defect search.....	60
Table 9. Result of implementing improved vision-based grading algorithm.....	62
Table 10. Result of implementing external defect classification algorithm.....	65
Table 11. Orange colour class classification.....	71
Table 12. Orange class distribution matrix with detailed analysis.....	73
Table 13. Example of the orange class distribution matrix.....	74
Table 14. Orange colour class availability test for ripe oranges in the database.....	74
Table 15. Example of the orange class mean matrix.....	77
Table 16. Intensity variations along the horizontal blue line.....	78
Table 17. Intensity variations along the horizontal red line.....	80
Table 18. Example of the statistical analysis matrix.....	83
Table 19. Statistical analysis matrix with computed between-class variances.....	85
Table 20. Example of the covariance matrix.....	86
Table 21. Example of four different means.....	89
Table 22. Example of the computation for quadratic means.....	90
Table 23. Example of finding the closest neighbor class.....	91
Table 24. Example of the orange class mean matrix for a ripe orange.....	91
Table 25. Example of the orange class mean matrix for an unripe orange.....	91
Table 26. Example of the closest neighbor array with specified class numbers.....	92
Table 27. Class reclassification.....	93

Table 28. Example of the new class mean matrix for demonstration purpose.....	96
Table 29. Summary of classification results using the novel algorithm.....	114
Table 30. Sample data selected for Otsu's method implementation.....	117
Table 31. Sample data selected for image preprocessing.....	123
Table 32. New class mean matrix for demonstration purpose.....	127

Chapter 1

Research Description

1.1 Overview of the Current State of Technology

Orange is an important horticultural produce around the world amounting to millions of tons per annum (Thomas, 2009). Post-harvest diseases and mechanical damages greatly reduce the market value. Traditional inspection of fruits is performed by human experts, which is considered to be time-consuming and subjective (Brosnan & Sun, 2004). With the advent of fast and high-precision machine vision technologies, automation of the grading process is expected to reduce labour cost while significantly improving the efficiency, consistency and accuracy of this process (Du & Sun, 2004). Grading of fruits shares several common features with more classical automated inspection of manufactured goods. However there are not many robust and accurate grading systems targeting fruit defects comprehensively in the market. This defect scrutiny problem is significantly more difficult, because there is a wide range of colour and texture variations found in natural products (Chen, Chao, & Kim 2002).

Machine vision systems form their judgment based on specially designed image processing software (Bharati, Liu & John, 2004). Texture classification algorithms are grouped into four major categories based on the types of features they are associated with, such as statistical-based (Unay, 2005), structural-based (Recce, Taylor, Piebe & Tropiano, 1996), model-based (Chang, et al., 1994) and transform-based (Vijayarekha & Govindaraj, 2006) algorithms. However, most of the existing algorithms are not able to explicitly mark the pixels corresponding to the blemishes but could only provide a final answer (i.e. good or bad orange). Moreover, most of these systems are not easily customizable to meet the evolving requirements set by the market. For instance, a system trained to return 3 different possible fruit quality

bands will have to be retrained exhaustively by experts if the customer wishes to add more fruit quality bands. This tedious process is usually time-consuming and end-users of the system will not have the ability to refine the system on their own. Different fruit species and varieties will demand for a completely different training set, long and arduous process of system refinement, increase in labor and production cost, etc, and will not always guarantee to produce the desired performance.

Typically, texture classification algorithms utilize abstracted features from a spatial or frequency domain that cannot be cross-examined visually by humans (Johnson, 2008). It is extremely difficult to define any geometrical or spectral properties for the orange skin due to the wide-spectrum in variation found in organic produce. Texture classification algorithms are largely based on the use of neural networks as these systems are very good at finding useful correlations between features (Egmont-Petersen, Ridder & Handels, 2002). However, they require a vast amount of training exemplars, and are computationally intensive to train. Several researchers attempted to use a neural network-based classifier to achieve a more thorough analysis of the surface of fruits.

Here, we mention some of the prominent researches done on orange fruit grading. In (Recce, Taylor, Piebe & Tropiano, 1996), it is developed a high speed vision based orange grading system which is largely based on the use of neural networks to achieve a more thorough blemish analysis, including the detection of stems. Multiple views of an orange are analyzed and any views with low probability of containing defects or a stem would be excluded from further processing. This probability is estimated by a neural network algorithm that feeds on colour histograms (normalized red and green) extracted from the images. The resulting probability determines the goodness of Gaussian curve-fitting and that is used for grading oranges in the first stage of processing. It was reported that many detectors do not respond strongly to the defects on oranges. Therefore, five larger and smoother operators were invented for local defect detection purpose. Moreover, the orange pickers would have filtered some of the bad oranges already and so most oranges with good quality are delivered

to the packing house. Sorting the image on the pipeline can improve the throughput and overall grading performance. As a result, the time frame required for processing each orange is relatively smaller as blemished and relatively bigger oranges that take longer to process are placed at a lower priority in the queue. This algorithm however, is computationally expensive, and the intelligent time management has to be incorporated with state-of-the art hardware. Large amount of training samples are also required before any testing is done.

In (Vijayarekha & Govindaraj, 2006), it is suggested that features extracted from the images of fruit in either spatial or frequency domain can be used for defect classification. Wavelet-based texture classification methods use the wavelet sub-bands to extract textural features. These features are analyzed and extracted at different scales. The high frequency sub-band is decomposed further into a combination of high-frequency and low-frequency sub-windows. This is repeated successively until 16 sub-windows are extracted (or two levels with Daubechies). The algorithm is described to work similar to an advanced edge detector. It detects the blemishes by finding the intensity transition areas. However, the algorithm fails if the oranges are fully rotten and there are no textural differences on the fruit.

In (Unay, 2005), a novel artificial neural network-based segmentation and apple grading system was proposed. The background of the image is removed using a constant thresholding approach. The experiment results show that using a constant threshold will remove some of the low intensity areas as well, such as some very dark blemishes. Hence, a morphological filling operation is also used to recover these small holes. As for the inputs, statistical, textural and shape-based features are extracted from each of the four filter images, and are fed to support vector machines classifier. The regions identified as the stem are removed from the segmentation result. The algorithm was tested on five supervised classifiers, such as the Linear Discriminated Classifier (LDC), Nearest Neighbor Classifier (k-NN), Fuzzy Nearest Neighbor Classifier (fuzzy k-NN), Adaptive Boosting (AdaBoost) and Support Vector Machines (SVM). It was noted that the AdaBoost and SVM

classifiers perform the best with 90.3% overall recognition. As reported, the SVM is deemed more suitable to this system, because it does not require previous training set.

Upon examining the aforementioned algorithms, a novel system for grading oranges into different quality bands, according to their surface characteristics, is devised and presented in this paper. Both ripe and unripe oranges comprise the benchmarking dataset. It was observed that unripe oranges are more difficult to analyze for defect detection due to the colour transition areas. In addition, global intensity variation between pixels from the same orange is deemed not to be sufficient to classify defects correctly. Most of the existing algorithms are disregarding this significant issue. However, the novel algorithm takes full advantage of the global intensity variation for blemish detection purposes. We provide evidence of the merits of using this global intensity variation in our experiments.

1.2 Research Objectives

To devise an adaptive, intelligent grading system for oranges that allows for ripe/unripe classification, blemish detection, stem detection and removal, feature quantification and grading using the local intensity variation on RGB colour images.

1.3 Research Methodology

1. Study advanced image processing techniques, such as canny edge detection and Otsu's optimum thresholding method.
2. Study colour models in which way colours can be represented, such as RGB colour model.
3. Study advanced texture classification methods, such as model-based and transform-based methods.
4. Implement existing fruit grading algorithms.

5. Modify and improve existing fruit grading algorithms.
6. Explore the flaws and strengths of existing algorithms in order to devise a novel algorithm.
7. Gather and create a complete dataset for testing ripe and unripe oranges.
8. Design and develop a novel ripe/unripe orange classification algorithm.
9. Design and develop a novel blemish detection algorithm.
10. Design and develop a novel stem detection and removal algorithm.
11. Design and develop a novel blemish quantification algorithm.
12. Design and develop a novel grading algorithm.
13. Test the novel orange grading algorithm.
14. Modify and improve the novel orange grading algorithm.
15. Study advanced parallel image processing algorithms.
16. Explore the flaws and strengths of the parallel image processing algorithms.
17. Compare the performance of the system developed against the commercial available orange grading systems.

1.4 Scope and Limitations of Research

1. The novel algorithm is implemented and tested for ripe and unripe oranges only.
2. Database consists of 170 navel orange images.
3. Orange images are taken from the commercial machine on a standard operating condition.

1.5 Benchmarking Testbed

The proposed algorithms presented here were tested on a collection of 170 (blemished and unblemished, ripe and unripe) orange images captured using commercial-grade sorting equipment, operating at standard conditions.

As depicted in Figure 1, the commercial machine is comprised of a transport

system that moves and rotates the orange fruit, and a vision inspection system with controlled lighting that houses multiple digital cameras (AVT marlin 033C, 640x480 pixels) and mirrors. The imaging system captures 25 images per fruit at different angles and is interfaced with a computer through an IEEE 1394 connection. The computer can control the camera for adjusting any of the view settings, such as hue, brightness, contrast, lens focus and magnification.



Transport System



Vision Inspection System

Fig. 1. Commercial orange grading system.

1.6 Structure of the Thesis Documentation

This thesis consists of six chapters. Chapter 2 presents advanced image processing techniques, colour models and intelligent parallel distributed systems.

Chapter 3 presents a review of related literature, discussing advanced robust, intelligent fruit grading systems in detail, and analyzing the flaws and strengths of previous approaches.

Chapter 4 provides the implementation experience for some image processing and fruit grading algorithms. Most of them are largely based on the use of neural network to achieve a more thorough analysis of the surface of fruits. The performance is improved by modifying some of the algorithms.

Chapter 5 presents the novel orange grading algorithm developed. The algorithm is based on the local intensity variation between pixels from the same orange. Views that may contain defects are further analyzed using geometrical and spectral properties from the natural surface characteristics of oranges. Five key steps are introduced, such as ripe/unripe orange classification, blemish detection, stem

detection and removal, blemish quantification and grading. The algorithm is implemented and tested, and the experiment and analysis are presented in detail.

Chapter 6 reviews the contributions of this work, and identifies promising areas of research worthy of conducting future works.

Chapter 2

Theoretical Framework

2.1 Advanced Image Processing Techniques

Various intelligent image processing techniques are presented in this section, such as Otsu's method and Canny edge detector. Machine vision systems form their judgment based on specially designed image processing software. Image processing is the core part of the fruit grading system.

2.1.1 Otsu's Method

2.1.1.1 Overview

Thresholding is the simplest method of image segmentation. The core process in the thresholding is the choice of segmentation point. From a grayscale image, thresholding can be used to create binary images. Individual pixels in an image are marked as foreground pixels, for instance if their values are greater than the segmentation point, as background pixels otherwise. Logical matrix contains only "0" and "1" can be used to represent an image. Typically the foreground pixel is given a value of "1", and the background pixel is given a value of "0". The segmentation point may be selected manually by a user or computed automatically using a thresholding method. The mean or median value will work well to obtain a sufficient threshold value based on the most dominant pixel values in a noiseless image with uniform foreground and background pixel values, however this will generally not be the case (Johnson, 2008). An example of object segmentation using a constant threshold value is shown in Figure 2.



Fig. 2. Object segmentation using a constant threshold value.

In machine vision and image processing, Otsu's thresholding method is used to automatically perform histogram shape-based image thresholding, or converting a gray level image to a binary image (Zhang & Hu, 2008). The histogram method assumes that there is some average value for the foreground and background pixels, but the actual pixel values have some variations around the average value. Otsu's thresholding method minimizes the weighted within-class variance and this turns out to be the same as maximizing the between-class variance. Otsu's thresholding method operates directly on the grey level histogram and assumes that the image to be thresholded contains two classes of pixels (i.e. foreground and background). The difficult part during the process of clustering pixels is that two classes of pixels usually overlap, so minimizing the error of classifying a background pixel as a foreground becomes significant. Otsu's thresholding method calculates the optimum threshold by separating these two classes, so that their combined spread (intra-class variance) is minimal. Otsu's thresholding method makes each cluster as tight as possible and minimizing their overlap. Otsu's thresholding method does not require much specific knowledge of the image, and is robust against image noise.

The cost of Otsu's thresholding method is computationally cheap once the histogram is generated. However, the cost of Otsu's thresholding method would be very expensive when extended to a multi-level threshold due to the fact that a large number of iterations are required for computing the cumulative probability and the mean of a class (Cao, Shi & Cheng, 2002). In real time application, most of the methods suffer from time-consuming computations for multilevel thresholding. TSMO thresholding method is a two-state multi-threshold Otsu method which can significantly improve the efficiency with an accuracy equivalent to Otsu's method by greatly reducing the iterations required for computing the between-class variance in

a gray image.

The way to adjust the threshold is to increase the spread of one class and decrease the spread of the other. The goal then is to select the threshold that minimizes the combined spread. The within-class variance as the weighted sum of the variances of each cluster is defined as:

$$\sigma_{\omega}^2(t) = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t) \quad (1)$$

Otsu's thresholding method minimizes the weighted within-class variance and this turns out to be the same as maximizing the between-class variance which is defined as:

$$\sigma_b^2(t) = \sigma^2 - \sigma_{\omega}^2(t) = \omega_1(t)\omega_2(t)[\mu_1(t) - \mu_2(t)]^2 \quad (2)$$

2.1.1.2 General Algorithm

Input: Grey scale image.

Output: Threshold.

Data: I = grey scale image, t = Threshold, i = grey level in image I , $minGL$ = minimum grey level in image I , $maxGL$ = maximum grey level in image I .

```
1  foreach grey level  $i$  in image  $I$  do
    ComputeProbability()
  end
2  ComputeAverageIntensity()
3  foreach grey level  $i$  in image  $I$  do
    ComputeClassProbabilityForClassOne()
    ComputeClassProbabilityForClassTwo()
    ComputeClassMeanForClassOne()
    ComputeClassMeanForClassTwo()
    ComputeClassVarianceForClassOne()
```

```

    ComputeClassVarianceForClassTwo()
    ComputeWeightedSumVariance()
end
4   $t = \text{FindMaxWeightedSumVariance()} / \text{FindMinWeightedSumVariance}()$ 

```

1. Compute the probability p for each grey level.

$$P_i = \frac{n_i}{tn}$$

n is the number of pixels at grey level i . tn is the total number of pixels in image I .

2. Compute the average intensity m for image I .

$$m = \sum_{i=\text{minGL}}^{\text{maxGL}} P_i i$$

3. Step through all possible thresholds $t = \text{minGL} \dots \text{maxGL}$. The assumption is that pixels in image I are divided into two classes, class one and class two (i.e. foreground and background) by a threshold t . Class one denotes pixels with grey levels from l to t , and class two denotes pixels with grey levels from $t+1$ to maxGL .

1. Compute the class probabilities for class one and class two separated by a threshold t .

- Compute class probability for class one.

$$\omega_1 = \sum_{i=\text{minGL}}^t P_i$$

- Compute class probability for class two.

$$\omega_2 = \sum_{i=t+1}^{\text{maxGL}} P_i \quad \text{or} \quad \omega_2 = 1 - \omega_1$$

2. Compute the class means for class one and class two separated by a threshold t .

- Compute class mean for class one.

$$\mu_1 = \frac{\sum_{i=\text{minGL}}^{\text{maxGL}} P_i i}{\omega_1}$$

- Compute class mean for class two.

$$\mu_2 = \frac{\sum_{i=t+1}^{maxGL} P_i i}{\omega_2} \quad or \quad \mu_2 = \frac{(m - \sum_{i=minGL}^t P_i i)}{\omega_2}$$

3. Compute weighted sum of variances for class one and class two separated by a threshold t .

- Compute class variances for class one.

$$\sigma_1^2 = \frac{\sum_{i=minGL}^t (i - \mu_1)^2 P_i}{\omega_1}$$

- Compute class variances for class two.

$$\sigma_2^2 = \frac{\sum_{i=t+1}^{maxGL} (i - \mu_2)^2 P_i}{\omega_2}$$

- Compute weighted sum of variances of two classes.

$$\sigma_w^2 = \omega_1 \sigma_1^2 + \omega_2 \sigma_2^2 \quad or \quad \sigma_b^2 = \omega_1 \omega_2 (\mu_1 - \mu_2)^2$$

4. Desired threshold corresponds to the minimum σ_w^2 or maximum σ_b^2 .

For an illustration of the inner working of this algorithm with data samples, see appendix A.

2.1.2 Canny Edge Detection

Edge detection is the process of finding sharp contrasts in intensities. This process significantly reduces the amount of data in an image, while preserving the most important structural features. The majority of existing algorithms can be classified into two categories (Johnson, 2008), such as Laplacian of a Gaussian, first and second derivatives.

Canny edge detector is considered to be the ideal edge detection algorithm for images that are corrupted with white noises, and well known for its ability to

generate single-pixel thick continuous edges (Canny, 1986). Canny's idea and algorithm can be found in his paper, "*A Computational Approach to Edge Detection*". He followed a list of criteria to improve current methods of edge detection.

1. The first and most obvious criterion is the low error rate. It is important that edges occurring in an image should not be missed.
2. The second criterion is the edge points are well localized. In other words, the distance between the edge pixels as found by the detector and the actual edge is to be at a minimum.
3. A third criterion is to have only one response to a single edge. This was added because the first two steps were not substantial enough to completely eliminate the possibility of multiple responses to an edge.

2.1.3 Morphological Operators

2.1.3.1 Overview

Often it is necessary to process binary images. In a binary image, each pixel can only take two values, such as zero or one. The value is primarily used to denote the presence or absence of a feature which could be the result of previous processing, such as edge detection. The binary image also gives information about the shape of an object. Neighborhood operators for binary images are called morphological operators because they deal with the shape information (Johnson, 2008).

A pixel can be connected to another pixel in two ways, such as four connected and eight connected. A simple way to decide if a pixel is joined to one of its neighbors is to check all eight of the neighbors. Pixels are four connected, if they are joined to the left, right, above or below, but not diagonally. Pixels are eight connected, if they are joined to the central pixel. Eight connectedness correctly connects diagonal lines. However, it also connects the background across a diagonal line. The background must be treated differently from the foreground if eight

connectedness is used. Figure 3 shows a four connected and eight connected figures.

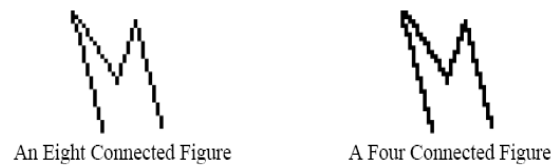


Fig. 3. Four and eight connectivity(Johnson, 2008).

Shrink and expand are two important morphological operators. Shrink will clear the pixels that have any non-class neighbors. Expand will set the pixels that have any class neighbors. An expand followed by a shrink is called a closed because it fills small holes between objects. A shrink followed by expand is classed an open because it keeps the small holes between objects open. Figure 4 shows the shrink, expand, open and close.

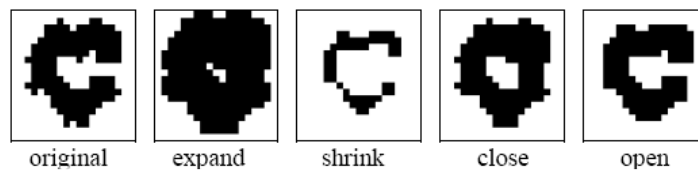


Fig. 4. Expand, shrink, close and open(Johnson, 2008).

The neighborhood operators, min and max work the same as shrink and expand. Taking the original image away from the expanded image works the same way as a contour detector. It is correspond to the outer four connected edge.

The shrink operator can be used to reduce the size of a region. If the contour of a region is important than shrinking to nothing, then an operator can be applied to shrink the region to its skeleton. The skeleton is a set of points which are equidistant from the two or more close edge points in the image (Johnson, 2008). Skeletonisation may be performed by repeatedly applying an operator which shrinks the image until the skeleton remains. Figure 5 shows an example of shrinking a region to its skeleton.



Fig. 5. Shrink a region to its skeleton(Johnson, 2008).

2.1.3.2 Convex Hull

In mathematics, the convex hull is defined as follows.

1. A set of data points P .
2. A real vector space V .
3. P in V is the smallest convex set containing P .

For a given non-empty finite set of n data points, the convex hull computation means find the boundary points which can form a simple closed polygonal chain (Preparata & Hong, 1977). The polygonal chain should contain all the data points P and its convex set is minimal. The number of points on the convex hull is defined as follows:

$$H_{convex}(X) = \{ \sum_{i=1}^k \alpha_i x_i \mid x_i \in X, \alpha_i \in \mathbb{R}, \alpha_i \geq 0, \sum_{i=1}^k \alpha_i = 1, k = 1, 2, \dots \} \quad (3)$$

Convex hull is a very useful image processing technique in computer vision. Figure 6 shows some holes within the red layer in a binary image.

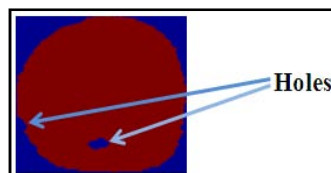


Fig. 6. Original image.

Figure 7 shows the convex image after the convex hull operation. The holes within the red layer are sealed.

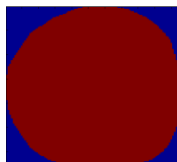


Fig. 7. Convex image.

The original image is defined as mask one, and the convex image is defined as mask two. The holes in the original image can be easily segmented by using mask two minus mask one. Figure 8 illustrates the sealed holes, which are referred to the

stem or blemishes in Section 5.4.8.



Fig. 8. Sealed holes.

2.2 Colour Space

2.2.1 Overview

Colour is the visual perceptual property corresponding in humans to the categories called red, yellow, blue and others. Spectrum of light is the distribution of light energy versus wavelength. Colour derives from the spectrum of light interacting in the eye with the spectral sensitivities of the light receptors (Werblin, Jacobs & Teeters, 1996). Colour can be represented as tuples of numbers, typically as three or four values or colour components (Deng et al., 2001). Figure 9 illustrates the RGB and CMYK colour model. The way of describing colours in such an abstract mathematical model is called colour space. RGB colour space is the most common way to encode colours in computing for sensing, representation, and display of images (Bumbaca & Smith, 1987).

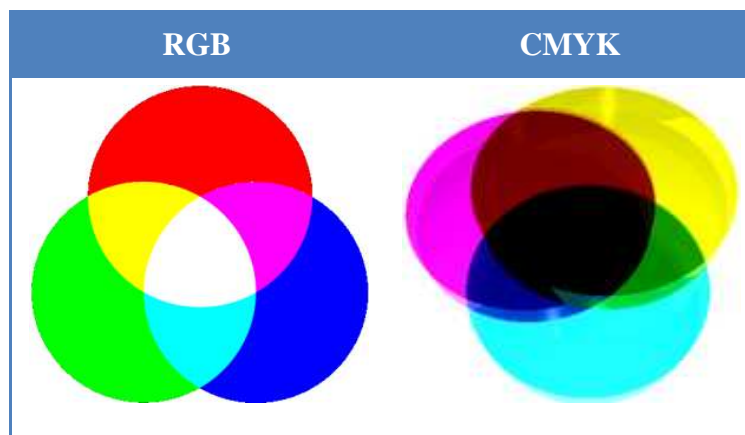


Fig. 9. RGB and CMYK colour model.

2.2.2 RGB Colour Space

Human eye is sensitive to three additive primary colours, red, green and blue. In machine vision, RGB (Red Green Blue) colour space is one of the most popular additive colour systems which are derived from human perception of colour (Rogowitz, 2001). It is called an additive colour system, since you could add light from the primary colours to make new colours. Human visual system is able to differentiate between 100-200 grey levels and 30,000-50,000 colours (Johnson, 2008). For this reason most vision systems use 256 grey levels (8 bits per pixel) or 256 levels of red, green and blue (24 bits per pixel). The 256 levels of red, green and blue usually do not represent equally spaced intensities, due to gamma correction (Farid, 2001). Figure 10 demonstrate a board array of colours (more than 16 million) can be displayed by using an appropriate combination of red, green and blue intensities.

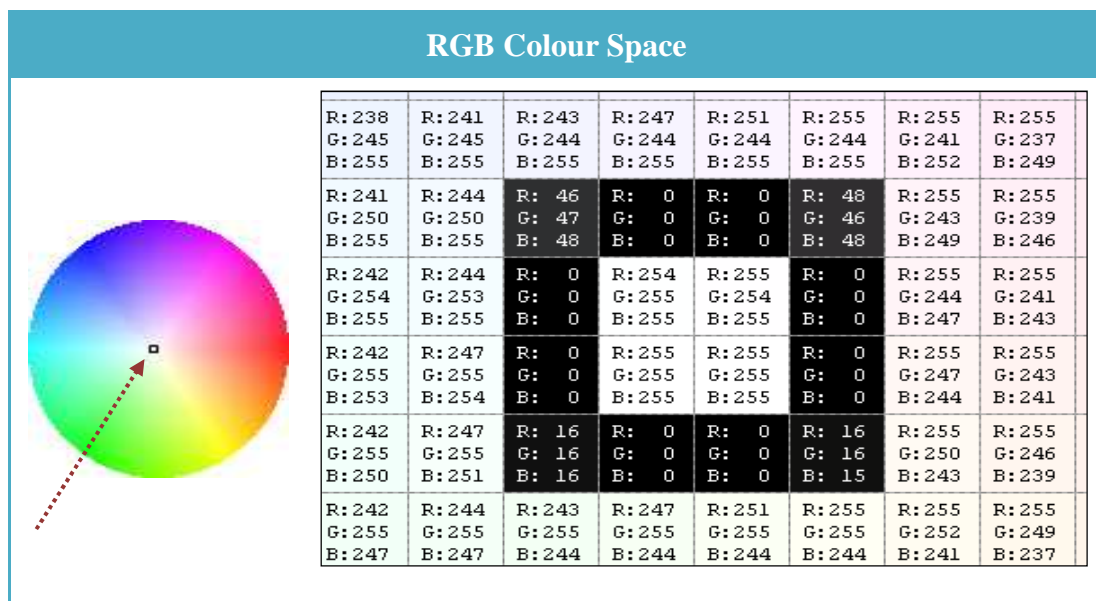


Fig. 10. Example of RGB colour space.

In RGB colour space, each primary colour is expressed as a channel which is used to refer to a certain component of an image in the conventional term (Bumbaca & Smith, 1987). A channel can be used to generate a grayscale image with the same size as the original image. In a grayscale image, each pixel only carries intensity

information. The intensity of a pixel is expressed within a given range between a minimum and maximum, inclusive. For instance, the weakest intensity is black(0), the strongest intensity is white(255) and many shades of gray in between. Figure 11 shows an example of the colour channel splitting of a full RGB colour image. The column at left shows the isolated colour channels in natural colours, while at right there are their grayscale equivalences:

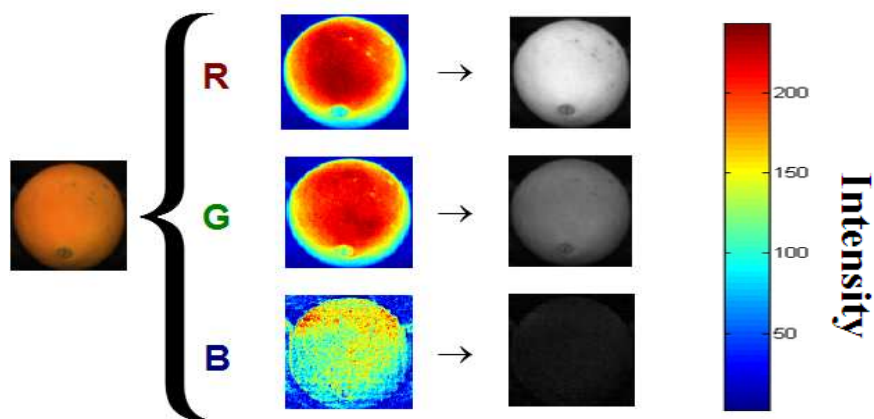


Fig. 11. Colour channel splitting of a full RGB colour image.

2.3 Parallel Image Processing System

2.3.1 Overview

Nowadays large amount of data are required for image processing, e.g., medical imaging, the traditional single processor is not able to complete complex tasks within a reasonable time frame. Parallel image processing (PIP) has been a topic of interest for many years. The basic idea is to use multi-processors to perform a single or multiple tasks at the same time (Messom, 2008). Large problems can almost always be divided into smaller ones. The maximum speedup is n with n processors. But in practice, it can't be achieved according to Amdal's Law (Onyuksel & Hosseini, 2002). The serial section is usually slow, such as reading data from a CD. The communication between processors is also considered to be time-consuming (Lekic, Mijanovic & Gobovic, 2002).

2.3.2 System Architecture

Palatin, Buzek, and Beran (2007) developed a parallel image processing system. Figure 12 is a block schematic of the system developed.

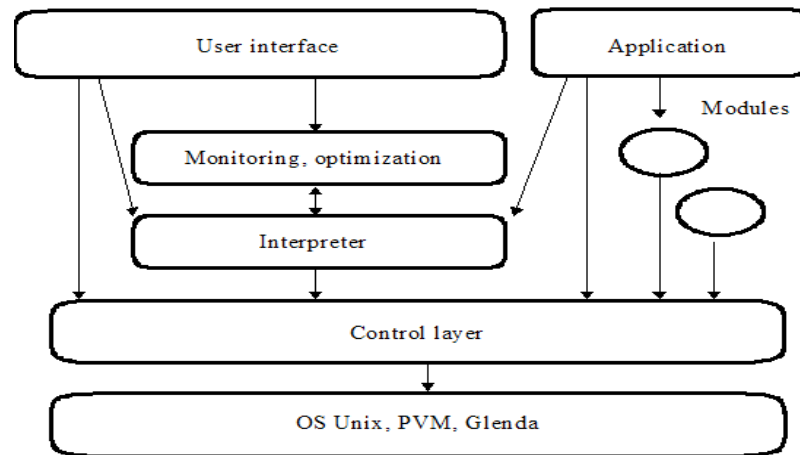


Fig. 12. Parallel image processing system(Palatin, Buzek, & Beran, 2007).

The system consists of six logical layers, such as system layer, control layer, functional layer, shell layer, optimization layer and application layer. Each layer performs a specific task.

1. System Layer: Parallel environment.
2. Control Layer: Communication between processors.
3. Functional Layer: Image processing algorithms.
4. Shell Layer: User interface.
5. Optimization Layer: Monitoring the system and optimize the performance.
6. Application Layer: Various image processing tasks.

The system can be divided into two parts, such as application and module. Functional modules can be implemented in parallel. The system uses the master-worker parallel programming to control modules by application:

- An application(master) sends commands to the system.
- Commands are stored in the shared memory.
- The functional modules(workers) take these commands, perform requested tasks and return the results back to the application.

- One application can send another command before getting the result from previous one. Each functional module may internally consist of many parallel processes.
- It would be very efficient if the functions provided by the module can be paralleled.
- A segmentation module can have several processes (sub modules). Each of them does the segmentation for one part of the image. Then the module joins the results of all the sub modules.

Two ways to reduce the unnecessary communication:

1. A shared database of data objects, not application.
2. Sending data directly from one module to the other (pipelining).

The shared memory is implemented as a shared associative memory, e.g.

- The data is accessed by a key value, not address.
- Provided by the coordination language Linda.
- The system is designed to use Linda implementation called Glenda as a basic parallel environment.

Chapter 3

Review of Related Literature

3.1 High Speed Vision-Based Quality Grading of Oranges

3.1.1 Overview

Recce, Taylor, Piebe and Tropiano (1996) developed a high speed vision-based orange grading system which is largely based on the use of neural networks to achieve a more thorough analysis, including the detection of stems. It is hard to define any geometrical or spectral properties for the orange skin, so the neural network classifier is fed by different feature extractors.

Most of the processing of fresh fruits in the packing house is highly automated, such as washing and packing. However the most important steps, e.g., inspection and grading in quality, are still performed manually throughout the world. This is because the automated quality control requires fast and complex image analysis.

Oranges are categorized according to surface textures, such as discolouration, bruising and other blemishes. The grading is based on the size of defects on the orange. If the stem can be successfully separated from the potential defects, then the accuracy of the resulted grading would be improved.

High speed is required in the real-time application, and a global inspection is applied by default to every view of each orange. In general, high quality oranges take shorter time to be categorized, and low quality oranges take longer time. The processing time is also affected by the orange size. Most of the oranges with good quality are delivered to the packing house. For this reason, the time frame between adjacent oranges can be much smaller than the time spent to process the most detailed analysis for a worst-case orange. Sorting the image on the pipeline can improve the throughput and overall grading performance. The intelligent time management has

to be incorporated with state-of-the art hardware, such as digital signal processor and specialized neural network parallel processor.

Vision-based quality grading system is tested in section 4.2.

3.1.2 Image Capture and Processing

The hardware components are constructed as follows:

1. A master processor.
2. A colour frame grabber based on Texas Instruments TMS320C40 DSP.
3. A Philips prototype board based on a L-Neuro2 parallel neural engine.
4. An industrial digital interface board.

There are three main image processing stages, such as histogram analysis, local defect search and stem detection. A fourth process provides the global supervision and control of the other three processes. The controlling process keeps track of the locations of individual oranges that are currently within the machine. Oranges are rolled and moved under the camera. The entire surface of the orange is imaged by using six planar views normal to the axes of a Cartesian coordinate.

As soon as the image is captured, the initial image extraction is performed by the C40 DSP. Afterwards, each of the views is analyzed, and a view is excluded from further processing if the particular view has low probability to contain either defects or the stem. This probability is predicted by a neural network algorithm that classifies colour histograms (normalized red and green) of the pixels.

The DSP passes the remaining views to the L-Neuro board for a more detailed analysis on the local surface area. A region-based operator is applied to the whole image, and each region is classified as normal or defected using a second neural network.

Before treating an identified region as a defect, it is necessary to double check if the defect is a stem or not. If the defect is much bigger or smaller than the size of a

typical stem, then there is no need to perform this operation. A third neural network is used for the stem identification.

After classification, the oranges on the pipeline are deflected to appropriate bins using purpose built pneumatic valves.

3.1.3 General Algorithms

3.1.3.1 Histogram Analysis

The first stage in image processing is named histogram analysis. The colour distribution of each of the views of an orange is analyzed and turned into features, such as mean and standard deviation. Histogram analysis of the normalized pixel values is targeted on the red and green components only. The assumption is that a good orange has normally distributed red and green colour components, since the natural skin pigmentation is composed mostly of one colour. Defects tend to interrupt the smooth normal distribution on the red and green components. However this effect is small and the number of pixels on the defected area is not predictable. Therefore, a view can be computed by the fit of the frequency distribution of the pixel values on the red and green components. The result which fit well to a Gaussian will be classified as a good orange. The best fit Gaussian $g(x)$ is given by:

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (4)$$

where x is the pixel value, σ and μ are the standard deviation and mean computed from the distribution. The error is the summed difference between the best fit function and the histogram data. Defects produce characteristic ranges of errors in specific segments of the histogram. For this reason, the histogram data is divided into a set of segments I_i , where:

$$I_i = |\sum_x h(x) - g(x)|x \in X_i \quad (5)$$

The experiments show that a simple scoring rule based on the fit of a Gaussian to

the original histogram does not perform well compared with a neural network based classifier. The neural network based classifier is able to learn the characteristic differences in a measured distribution from a normal distribution. The neural network applied in this system is a modified form of the back-propagation training algorithm. The input layer has ten neurons and combines information from the red and green histograms. The output layer has two neurons. Two classes are predefined, one is for good oranges and the other one is for defected oranges. Oranges classified into the second class are passed to the second stage for a more detailed analysis.

In the first stage, a fraction of the top quality oranges may be classified as a lower quality band. From the commercial point of view, upgrading the quality is better than downgrading the quality. However, it is safer to downgrade the quality in the first stage and perform a more detailed analysis in the second stage.

The number of training samples should be at least ten times larger than the number of weights in a multilayer back-propagation network. More than three thousand of orange samples are used for training including a twenty percent testing set.

3.1.3.2 Local Defect Search

Defects can be segmented from the normal skin for a more detailed analysis. Local colour variation between pixels from the same orange should not be identified as defects. Various types of defects make the segmentation process even harder by only looking at the local structural or textural properties. All defect types contribute roughly equally to the final grading decision. The defect detector using a set of masks is applied to regions of the orange image. The defect is characterized by a discontinuity in the skin pigmentation. The extracted local features are fed into the second neural network classifier.

Many edge detectors do not respond strongly to the defects on oranges. For this

reason, five larger and smoother operators are invented for the local defect detection purpose. There are two key steps in this operation.

1. Divide the image in regions($N \times N$). N is the typical size of defects.
2. Apply five operators on red and green components separately. Each operator is an $N \times N$ matrix with integer elements, such as zero and one.

The first four operators are defined as follows:

$$\begin{aligned}
 m_{ij}^1 &= \begin{cases} 1, & i < \frac{n}{2} \\ -1, & \text{otherwise} \end{cases} & m_{ij}^2 &= \begin{cases} 1, & j < \frac{n}{2} \\ -1, & \text{otherwise} \end{cases} \\
 m_{ij}^3 &= \begin{cases} 1, & i > j \\ 0, & i = j \\ -1, & \text{otherwise} \end{cases} & m_{ij}^4 &= \begin{cases} 1, & i < (N - j) \\ 0, & i = (N - j) \\ -1, & \text{otherwise} \end{cases} \quad (6)
 \end{aligned}$$

The fifth operator is constructed based on the results of the third and fourth operators.

$$m_{ij}^5 = \begin{pmatrix} -m_{ij}^4 & -m_{ij}^3 \\ m_{ij}^3 & m_{ij}^4 \end{pmatrix} \quad (7)$$

Convolution is a simple mathematical operation which is fundamental to many common image processing operators. The following formula shows the convolution of $N \times N$ matrix and operator.

$$I_k^{r,g} = \sum_{i=1}^n \sum_{j=1}^n x_{ij}^{r,g} m_{ij}^k \quad (8)$$

The input layer of the second neural network contains ten neurons. Applying each of the five masks to the red and green components separately will generate ten results after computation. The output layer of the second neural network contains two neurons. Two classes are predefined, one is for normal skin, and the other one is for defected skin. The database used in the second stage is the same as the one in the first stage. Defects are extracted for training purposes. Some defects are the same, which differ only in the selection of the start point. The convolution is only applied to arbitrarily partitioned regions. This can help to increase the computational speed. In the second stage, the stem is treated the same as defects.

3.1.3.3 Stem Detection

Stem is hard to distinguish from defects. Histogram does not show significant differences between the stem and defects. Stem has a much more regular structure than defects. The high degree of radial symmetry can be used to aid the identification of stem. The family of Zernike moments is a powerful technique for stem detection. Zernike moments are very sensitive to circular symmetries and invariant under rotation. A two dimensional square mask based on a Zernike polynomial is taken to have a unit radius in polar coordinates. Each pixel is assigned a complex value.

$$Z_{nm}(\theta, r) = [\cos(m\theta) + j * \sin(m\theta)] * R_{nm}(r) \quad (9)$$

The Zernike polynomial R_{nm} is defined as follows:

$$R_{nm} = \sum_{s=0}^{\lfloor \frac{n-m}{2} \rfloor} \frac{(-1)^s (n-s)!}{s! \left(\frac{n+m}{2} - 2\right)! \left(\frac{n+m}{2} - 1\right)!} r^{(n-2s)} \quad (10)$$

where n is the major order of the Zernike polynomial, and m is the minor order.

Figure 13 shows three Zernike masks.

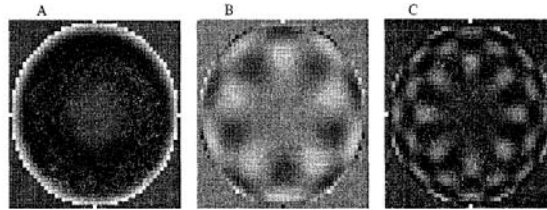


Fig. 13. Example of Zernike masks(Recce, Taylor, Piebe & Tropiano,1996).

The spatial features extracted from the suspected stems are fed into the third neural network classifier.

3.1.4 System Performance

This system is aimed directly for use in a commercial machine with stringent real-time requirements. The features of this system are cost-effective, robust,

simple and suited for implementation in parallel hardware. The experiment is based on four views of oranges.

1. The first view contains no defects.
2. The second and third views contain defects which have been detected.
3. The fourth view contains a detected stem.

Table 1 shows the errors given by the best of the neural networks tested. The error labeled “down” is the misclassification that potentially leads to a final downgrading of the orange.

Network	Training Set		Test Set	
	→ down	→ over	→ down	→ over
Histogram	0.00%	0.00%	13.73%	2.06%
Defect	0.00%	0.62%	0.80%	2.02%
Stem	0.11%	0.87%	0.44%	1.31%

Table 1. Vision-based algorithm classification result(Recce, Taylor, Piebe & Tropiano,1996).

The largest variance between the training and test errors is the histogram analysis. The histogram contains no spatial information, and unlikely to be refined significantly. Increasing the number of training samples may help to solve this problem. The evidence for the advantage provided by use of neural networks is shown in Figure 14. Part A is a good orange. Part B contains a detected stem. Part C and D contain identified defects.

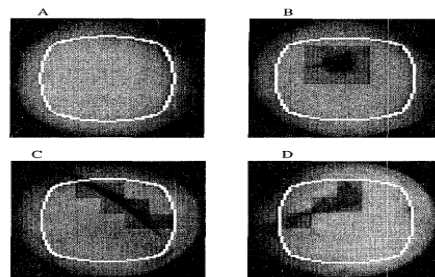


Fig. 14. Processed orange images using high speed vision-based algorithm(Recce, Taylor, Piebe & Tropiano,1996).

3.2 Citrus Fruit External Defect Classification

3.2.1 Overview

Vijayarekha and Govindaraj (2006) suggest that features extracted from the images of citrus fruit in either spatial or frequency domain can be used for defect classification. The external surface quality is directly related to the marketing and sales. The automatic grading system can significantly improve the accuracy and consistency while eliminating the subjectivity of manual inspection.

The defects on the external surface are caused by two reasons.

1. Pre-harvest and Post-harvest diseases. The major diseases are diplodia and phomopsis stem-end rot, splitting, pitting, green and blue mold, sour and brown rots, anthracnose, and alternaria rot, etc.
2. Mechanical damages during transportation.

The defects on the citrus fruit are characterized by different textures. Among various textures, three types of defects are categorized, such as pitting, splitting and stem-end rot.

1. Pitting is caused by mechanical damage or reduced gas exchange during transportation. Pits can coalesce to form irregular patches and brown to black blemishes.
2. Splitting is caused by the inability of the outer skin to hold the weight of the whole fruit. The outer skin of the citrus fruit splits and the inner pulp gets exposed. The defective region is usually brighter compared with the normal skin.
3. There are two types of stem-end rot, such as phomopsis stem-end rot and diplodia stem-end rot. Both types are very similar at the initial state, however, phomopsis stem-end rot soon will become tan to dark brown and a clear line of demarcation is formed at the junction between diseased and normal skin.

Wavelet packet transform (WPT) extends the wavelet transform. Wavelet analysis provides the spatial and frequency information which is useful for texture classification. The features, e.g., mean and standard deviation, are extracted from each of the detail as well as the approximation sub-windows, and then fed into the Artificial Neural Network (ANN) classifier for defect classification.

This system cannot classify the stem-end rot defect which appears on the entire surface of the fruit under inspection. If the edges are not clearly defined, then the features extracted from such images are not useful. The citrus fruit classification process is shown in Figure 15.

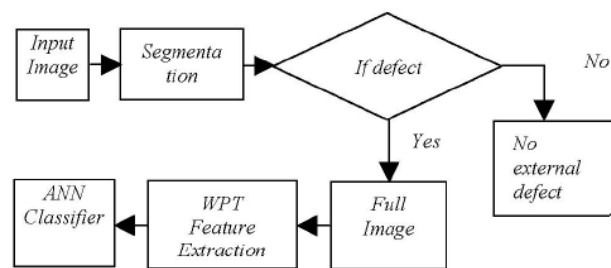


Fig. 15. Citrus fruit external defect classification process(Vijayarekha & Govindaraj, 2006).

The citrus fruit classification system is tested in section 4.3.

3.2.2 Wavelet Packet Texture Analysis

Wavelet-based texture classification methods extract textural features from the wavelet sub-bands. The advantages of using wavelet analysis are listed below:

1. Decompose the given input image into frequency sub-bands. Textures presented in the image can be analyzed and extracted at different scales.
2. Both low and high frequency sub windows can be analyzed. The high frequency sub window can be used as the next higher level of decomposition.
3. A high value of the wavelet packet transform coefficient represents more edges in the image. A low value of the wavelet packet transform coefficient represents a smooth texture.

The decomposition is performed by convolving the original signal separately with two half band pass wavelet filters.

1. The low pass decomposition filter (LD) removes all frequencies that are above half of the highest frequency and collects only the low frequency contents in the signal. The low pass filtering halves the resolution, but leave the scale unchanged.
2. A high pass decomposition filter (HD) removes all the frequencies that are below half of the highest frequency and collects only the high frequency contents.

The formula of the wavelet packet transform is defined as follows:

$$Y[n] = \sum h[k].x[2n - k] \quad (11)$$

where $Y[n]$ is the output sequence, and $X[n]$ is the input signal. Wavelet packet transform analyzes the signal at different frequency bands with different resolutions by decomposing the signal into a coarse approximation and detail information. Approximations and details are two orthogonal subspaces by splitting an individual discrete signal. Figure 16 shows the decomposition filter bank structure.

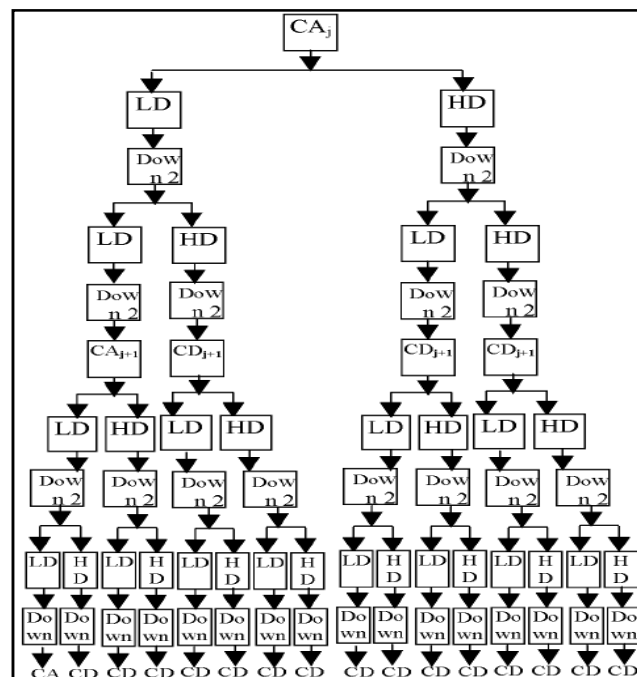


Fig. 16. Decomposition filter bank structure of WPT(Vijayarekha & Govindaraj, 2006).

The mean and standard deviation of 16 sub-windows are computed for each image using the following formulas:

$$\text{Mean}(m) = \frac{1}{N} \sum_{i,j=1}^N p_{ij} \quad \text{Standard deviation} = \left[\frac{1}{N} \sum_{i,j=1}^N [p_{ij} - m]^2 \right]^{0.5} \quad (12)$$

where P_{ij} is the coefficients of the wavelet packet transformed image at row i and column j , and N is the total number of pixels.

3.2.3 Neural Network Classifier

The neural network classifier is constructed as follows:

1. Thirty-two input neurons.
2. Three output neurons.
 - Pitting Defect: [1 -1 -1]
 - Splitting Defect: [-1 1 -1]
 - Stem-end Rot Defect: [-1 -1 1]
3. Ten hidden neurons.

The neural network architecture is shown in Figure 17.

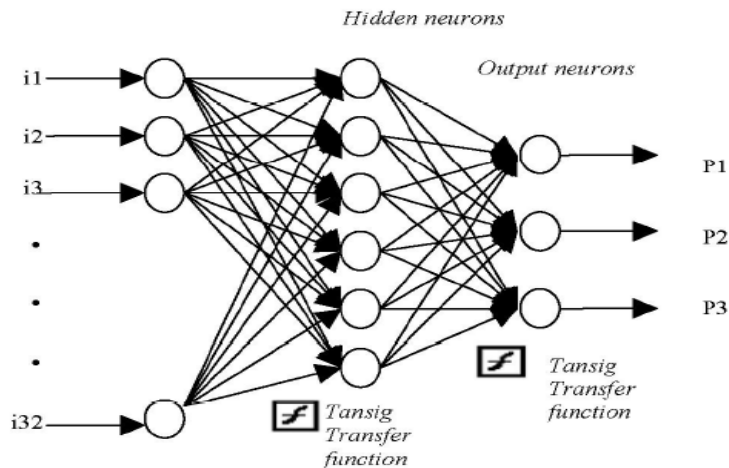


Fig. 17. Neural network classifier for defect classification(Vijayarekha & Govindaraj, 2006).

The error is considered to be the sum-squared error. Ten neurons in a single hidden layer are found on a trial, and the error basis is minimal. The training

method is the gradient descent back-propagation with momentum and adaptive learning rate. The momentum helps in faster convergence. Training stops at any of the following four conditions:

1. Maximum number of loops is reached.
2. Exceed the maximum time frame.
3. The predefined goal is reached.
4. Gradient falls below minimum gradient.

The transfer function between the hidden and output layer is the bipolar tangent sigmoid non-linear transfer function. Features are normalized and given as the input vector. The network cannot be over trained. The classification is defined as follows:

1. If the first output neuron $P1$ is higher than the others, then the fruit may contain pitting defects.
2. If the second output neuron $P2$ is higher than the others, then the fruit may contain splitting defects.
3. If the third output neuron $P3$ is higher than the others, then the fruit may contain stem-end rot defects.

3.2.4 System Performance

Twenty-three citrus fruits are collected from the market. They are manually categorized as follows:

1. Three citrus fruits are good.
2. Eight citrus fruits have pitting defects.
3. Five citrus fruits have splitting defects.
4. Seven citrus fruits have stem-end rot defects.

The image capture system is constructed as follows:

1. A colour CCD camera (Pulnix TMC-6700 CL).
2. A C-mount lens of focal length 6mm.

3. A camera link interface compatible frame grabber card (NI- 1428).
4. An illumination source.
5. A personal computer system (Intel Pentium IV @2.18 GHZ).

The citrus fruits are placed on the vertical stand. Three images are captured for each fruit with proper illumination at different positions. An image bank is generated for the image processing and classification purpose. The neural network classifier is trained using twenty-three citrus fruits. Forty-nine citrus fruits are selected as the testing dataset. They are manually categorized as follows:

1. Eight citrus fruits are good.
2. Nineteen citrus fruits have pitting defects.
3. Twelve citrus fruits have splitting defects.
4. Ten citrus fruits have stem-end rot defects.

Table 2 shows the classification result.

Fruit Image Types	No. of Images	No. of Correctly Classified Images	No. of Wrongly Classified Images	% of Correct Classification
Pitting	19	16	3	84.21%
Stem-end rot	10	5	5	50.00%
Splitting	12	12	0	100.00%

Table 2. Citrus fruit external defect classification result (Vijayarekha & Govindaraj, 2006).

The classification for the stem-end rot defects is not ideal. This is because some test samples are fully rotten and there are no textural differences on the fruit surface area. The intensity variation on such images is not obvious and the edges are not prominently defined. Increasing the number of training samples may help to identify the stem-end rot defects better. Some other images and artificial neural network classification results using wavelet packet transform features are shown in Table 3.










Image	Outputs obtained			Classification
	P1	P2	P3	
	0.9996	-0.9993	-0.9988	Pitting
	0.9993	-0.9965	-0.9982	Pitting
	0.9995	-0.9997	-0.9983	Pitting
	-0.9687	-0.9983	0.9558	Stem-end rot
	-0.9813	-0.6173	0.8965	Stem-end rot
	-0.9768	-0.7844	0.5761	Stem-end rot
	-0.9930	0.9989	-1.0000	Splitting
	-0.9259	0.9280	-0.9995	Splitting
	-0.9921	0.9920	-0.9999	Splitting

Table 3. Part of the citrus fruit classification results with images(Vijayarekha & Govindaraj, 2006).

3.3 Intelligent Fruit Sorting System

3.3.1 Overview

Guo and Cao (2004) developed an intelligent fruit sorting system based on colour image processing. The system is designed for any fruit with circular shape, such as apple and orange. A block schematic of the system is shown in Figure 18.

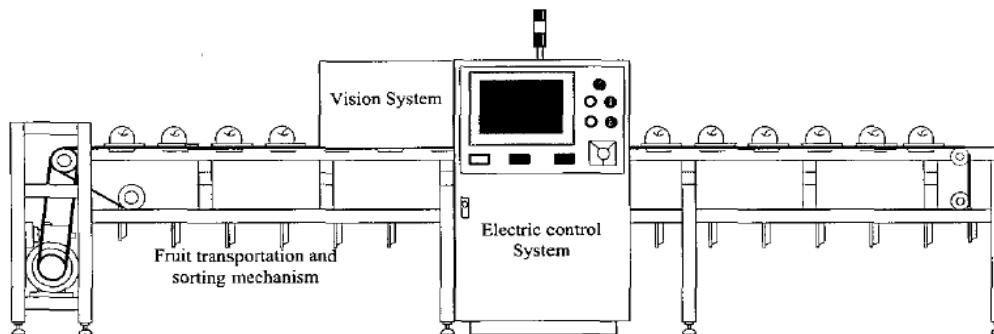


Fig. 18. Colour image processing based intelligent fruit sorting system(Guo & Cao, 2004).

The system consists of three parts.

1. Electric control system.
2. Vision inspection system.
3. Fruit transportation and sorting mechanism.

The vision inspection system is constructed as follows:

1. Two CCD cameras are installed on the top left and right corners with 120 degrees from each other.
2. One CCD camera is installed above the pans.

Majority of the fruit surface areas are imaged by three cameras. Some surface areas may not be inspected. However, the experiment result shows that this has little influence on the overall performance. A block schematic of the vision inspection system is shown in Figure 19.

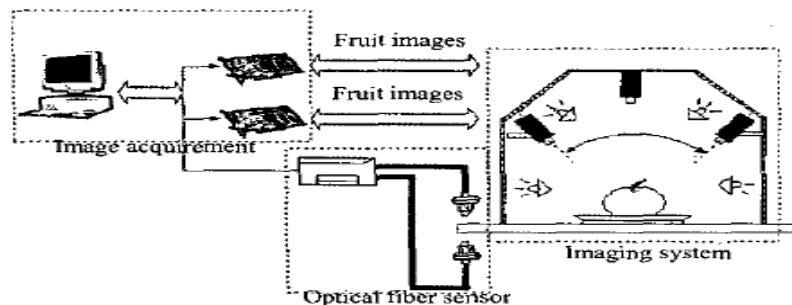


Fig. 19. Vision inspection system in fruit sorting system(Guo & Cao, 2004).

The system requirement is about five apples per second, so one apple has to be fully processed within two hundred milliseconds. The time allocation is designed as follows:

1. Image acquisition is limited within one hundred milliseconds. Two Matrox Meteor-II frame grabbers are chosen to speed up the process of image acquisition. The experiment shows that three images can be captured within one hundred milliseconds.
2. Image processing is limited within one hundred milliseconds. Three captured images have to be processed within one hundred milliseconds.

In order to improve the system performance, the following steps should be applied.

1. Optical fiber sensor is adopted to provide outside trigger signal for the

image system.

2. Multi-threads program structure is designed to improve the grabbing speed.
3. Single field grab mode is applied to resolve the blurred image problem caused by motion.

3.3.2 Feature Extraction

Fruit segmentation is an important step before image processing. The fruit segmentation has to be done within fifty milliseconds. The algorithm is required to maintain certain precision.

Ohta colour space is derived based on a set of orthogonal colour features. The conversion from RGB to Ohta colour space is liner and computationally cheap compared with HSI and HSV. The formula is shown below:

$$\left\{ \begin{array}{l} I_1 = \frac{(R+G+B)}{3} \\ I_2 = \frac{(R-B)}{2} \\ I_3 = \frac{(2G-R-B)}{4} \end{array} \right. \quad \left\{ \begin{array}{l} I'_2 = R - B \\ I'_3 = \frac{(2G-R-B)}{2} \end{array} \right. \quad (13)$$

A constant threshold value is selected for image segmentation purpose. The experiment result shows that the I_2 colour feature can mask out the image background. A sample segmentation result is shown in Figure 20.

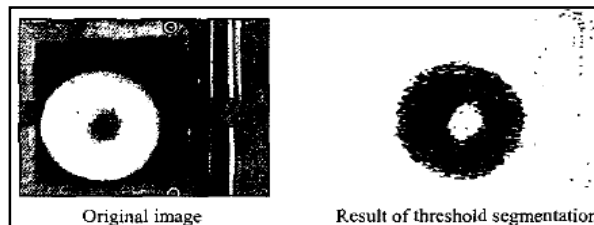


Fig. 20. Result of fruit segmentation using a constant threshold(Guo & Cao, 2004).

After the segmentation process, some noises are still on the background. Blob algorithm is selected as the noise removal tool. The procedure is defined as follows:

1. Select a constant value.
2. If the area is smaller than the constant value, then it will be removed from the image.

The result after implementing the blob algorithm is shown in Figure 21.

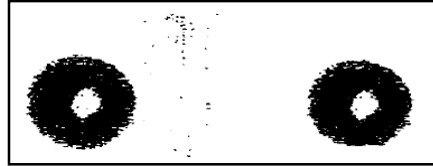


Fig. 21. Result of noise removal using blob algorithm(Guo & Cao, 2004).

The contour detection is crucial to the correct fruit shape feature extraction. The contour of apple blob is not smooth and traditional edge detector will mislead the feature extraction. The interpolation-based contour detection is applied in this system. There are four key steps in this process.

1. Compute the coordinates of geometrical center of apple blob.
2. Search twenty-four points on the contour fifteen degree each.
3. Compute the distances between the contour points and geometrical center.
4. The distances are used as the lengths of relating radius.

An example of the interpolation-based contour detection is show in Figure 22.



Fig. 22. Interpolation-based contour detection(Guo & Cao, 2004).

3.3.3 Colour Ratio Judgment

Uniform red colour distribution on the apple surface area is one of the criteria in quality inspection. High correlation coefficients have been obtained between the sugar content and the colour ratio of apple. The colour space is converted from

RGB to HSI. The HSI colour space consists of three components, such as hue, saturation and intensity. Hue is the pure colour, saturation is the contrast level, and intensity is the brightness. HSI colour space is considered to be more intuitive than RGB colour space. The conversion from RGB to HSI is defined as follow:

$$\begin{cases} H = \cos^{-1} \left\{ \frac{(R-G)+(R-B)}{2\sqrt{(R-G)^2+(R-B)(G-B)}} \right\} \\ S = 1 - \frac{3}{R+B+G} [\min(R, G, B)] \\ I = \frac{R+B+G}{3} \end{cases} \quad (14)$$

Figure 23 shows the hue colour feature distribution. Red colour is not uniform and restricted in certain areas.

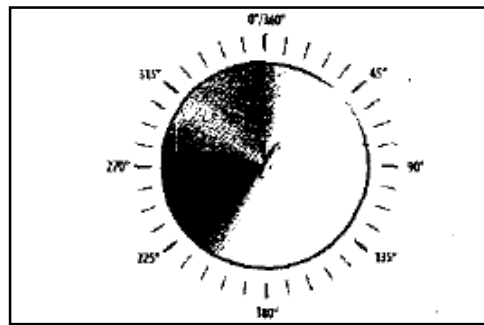


Fig. 23. Hue colour feature distribution(Guo & Cao, 2004).

Figure 24 shows the result of searching red pixels in an image.

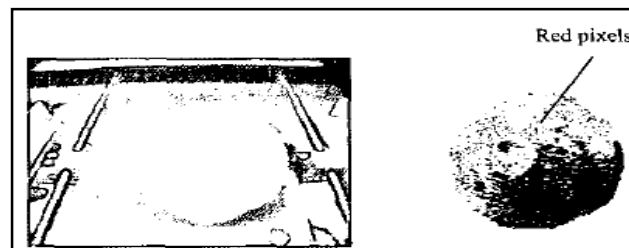


Fig. 24. Result of searching red pixels in an image(Guo & Cao, 2004).

The colour ratio of apple can be calculated using the following formula. C is the colour ratio. R_L , R_R and R_T are the areas of red pixels on the left, right and top images. A_L , A_R and A_T are the areas of apple on the left, right and top images.

$$C = \frac{R_L+R_R+R_T}{A_L+A_R+A_T} \quad (15)$$

3.3.4 Naive Bayes Classifier

A good classifier can learn standard patterns from different samples. The naive Bayes classifier is selected and trained for quality inspection purpose. The probability distribution of the apple image is considered to be a Gauss normal distribution. The estimated priori probability densities are computed using the following formula:

$$p(x|\omega_r) \quad p(x|\omega) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma}\right) \quad (16)$$

The estimated priori probabilities of classes are computed using the following formula. K is the total number of objects in the training set. K_r is the number of objects from the class r in the training set.

$$p(\omega_r) = \frac{K_r}{K} \quad (17)$$

The following formula shows how to classify all the patterns into class r :

$$\omega_r = \max_{i=1\dots s} [p(x|\omega_i)p(\omega_i)] \quad (18)$$

3.3.5 System Performance

The system is tested using Fuji apples. Three classes are predefined as follows:

1. Class A: More than seventy percent of the surface is deep red.
2. Class B: Forty to seventy percent of the surface is red.
3. Class C: Less than thirty percent of the surface is red.

Ten typical samples are selected from each class for training purpose. Sixty apples are selected for testing purpose. They are manually categorized as follows:

1. Twenty of them should belong to class A.
2. Twenty of them should belong to class B.
3. Twenty of them should belong to class C.

The classification result is shown in Table 4.

Samples	Number of apples classified into each group			Accuracy
	A	B	C	
A	18	2	0	90%
B	1	17	2	85%
C	0	1	19	95%

Table 4. Classification result of intelligent fruit sorting system(Guo & Cao, 2004).

The apple cannot be rotated on the conveying system, so some part of the surface is unchecked. This is the main reason why some of the apples are misclassified. If more images can be captured per image, then the performance of this system will be improved.

3.4 Neural Network-Based Apple Grading

3.4.1 Overview

Unay (2005) developed a novel artificial neural network-based segmentation and apple grading system. Database consists of one-view images of “Jonagold” apples. Images are captured from a diffusely illuminated environment. The high resolution monochrome digital camera has four interference band-pass filters centered at 450 nm(BL), 500 nm(GR), 750 nm(RE), and 800 nm(IR) with respective bandwidths of 80, 40, 80 and 50 nm. Each filter image is composed of 430x560 pixels with eight bits-per-pixel resolution. Figure 25 shows some examples of the filter images.

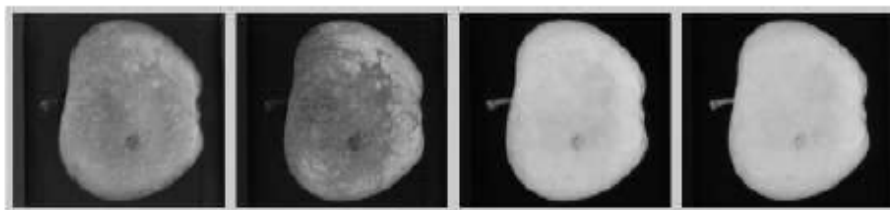


Fig. 25. BL, GR, RE, and IR filter images(Unay, 2005).

“Jonagold” variety is selected, because the bi-coloured skin is more difficult in

segmentation due to the colour transition areas. Figure 26 shows some examples of the “Jonagold” apples.

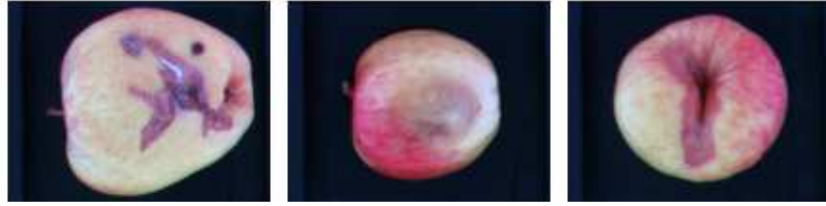


Fig. 26. “Jonagold” apple samples(Unay, 2005).

3.4.2 Defect Segmentation

Features are extracted from each of the four filter images using the following formulas:

$$\begin{array}{l}
 \text{statistical} \left\{ \begin{array}{l}
 \text{average}(\mu) = \frac{1}{N} \sum_{i=1}^N p_i \\
 \text{standard deviation}(\sigma) = \left(\frac{1}{N-1} \sum_{i=1}^N (p_i - \mu)^2 \right)^{\frac{1}{2}} \\
 \text{minimum}(\min) = \min(p_i) \text{ for } i = 1 \dots N \\
 \text{maximum}(\max) = \max(p_i) \text{ for } i = 1 \dots N \\
 \text{gradient}(\text{grad}) = \max - \min \\
 \text{skewness}(\text{skew}) = \frac{\sum_{i=1}^N (p_i - \mu)^3}{N \sigma^3} \\
 \text{kurtosis}(\text{kurt}) = \frac{\sum_{i=1}^N (p_i - \mu)^4}{N \sigma^4}
 \end{array} \right. \\
 \\
 \text{textural} \left\{ \begin{array}{l}
 \text{invariant moment}(\Phi_1) = \eta_{20} + \eta_{02} \text{ where } \eta_{xy} \text{ is the} \\
 \text{normalized central moment}
 \end{array} \right. \\
 \\
 \text{shape} \left\{ \begin{array}{l}
 \text{area}(S) = N \\
 \text{perimeter}(P) = N_p \\
 \text{circularity}(C) = \frac{P^2}{4\pi S}
 \end{array} \right. \quad (19)
 \end{array}$$

Image preprocessing is required before the defect segmentation. Images are captured on a dark, uniform coloured background. A constant threshold value is

used to segment the apple from the background. The experiment result shows that using a constant threshold will remove some low intensity areas as well, such as some very dark blemishes. Hence, a morphological filling operation is also used to recover these small holes. After background removal, fruit area is eroded by a rectangular structuring element with size adaptive to the fruit size.

Sometimes the neural network misclassifies healthy tissue closer to the fruit edges. An added new feature is inversely and linearly related to the distance of each pixel from the geometric center of the fruit. The neural network used for segmentation is the back-propagated network of perceptron neurons (BPNN), which makes binary decision between defected and healthy skin. The net has thirteen input neurons, five hidden neurons and two output neurons. The segmentation performance is independent of the number of hidden neurons above five. The net uses the adaptive learning rate and cross validation technique. The training and validation sets do not overlap.

Stem appears as dark blobs on images which is similar to some of the blemishes. BPNN does not consider the presence or absence of these regions. Figure 27 shows the stem before and after removal. The defected area is displayed in white on both images.

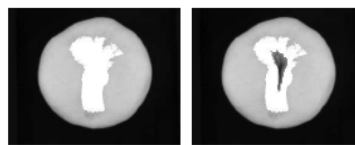


Fig. 27. Stem removal using neural network-based apple grading algorithm(Unay, 2005).

There are four key steps in the defect segmentation process.

1. Background removal and threshold-based object segmentation.
2. Statistical, textural and shape based features are extracted and introduced to support vector machines classifier.
3. The regions identified as the stem are removed from the segmentation result.

3.4.3 Neural Network Classifier

Average, standard deviation and median values are calculated over the segmented areas from all filter images. The ratio of defected pixels is also computed as one of the features. The features are normalized to have an average of zero and standard deviation of one before the classification. Five supervised classifiers are tested for the same algorithm.

1. Linear Discriminated Classifier (LDC), searches for a linear decision boundary that separates the feature space into two half-spaces by minimizing a criterion function.
2. Nearest Neighbor Classifier (k-NN), assigns an object to the most represented category among the k nearest samples of that object.
3. Fuzzy Nearest Neighbor Classifier (fuzzy k-NN), is the fuzzified version of K-NN. The formula is defined as follows:

$$u_i(x) = \frac{\sum_{j=1}^k u_{ij} (\|x-x_j\|)^{\frac{-2}{m-1}}}{\sum_{j=1}^k (\|x-x_j\|)^{\frac{-2}{m-1}}} \quad (20)$$

$u_i(x)$ is the predicted membership value of test sample x for class i , u_{ij} is the membership of j^{th} neighbor to the i^{th} class, and m is the fuzzifier parameter that determines how heavily the distance is weighted.

4. Adaptive Boosting (AdaBoost), tries to form a final strong classifier(g) from an ensemble of weak learners(h_t) by continuously adding these weak learners until the desired training error is reached. The formula for a test sample x is defined as follows:

$$g(x) = \text{sgn}[\sum_{t=1}^{t_{max}} \alpha_t h_t(x)] \quad (21)$$

5. Support Vector Machines (SVM), is a statistical learning method based on the structural risk minimization procedure. In the binary case, SVM tries to find the hyperplane that separates the classes with maximum margin. The formula for a test sample x is defined as follows:

$$g(x) = \text{sgn}[\sum_{i=1}^N \alpha_i y_i K(s_i, x)] \quad (22)$$

N is the number of training samples, y_i is the class label, and $K(s_i, x)$ is the kernel function.

Sample of the dataset are randomly ordered before training. This can help to prevent biased classification.

3.4.4 System Performance

Some examples of segmentation by neural network are shown in Figure 28. The defected regions are in gray colour and healthy ones in white colour.

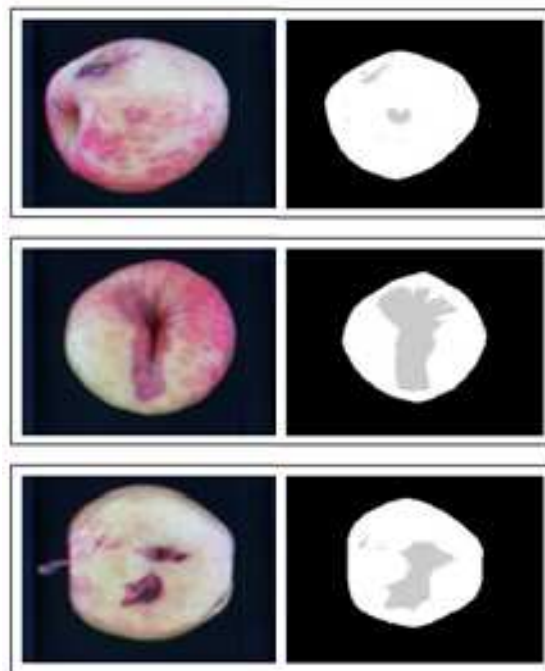


Fig. 28. Example of segmentation using neural network (Unay, 2005).

After the segmentation step, features are extracted from the segmented regions, and apples are classified into two predefined classes by several supervised classifiers. The experiment result is shown in Table 5. “D” refers to defected class, and “H” to healthy one.

classifiers		LDC		5-NN		Fuzzy 5-NN		AdaBoost		SVM	
		ground truth		ground truth		ground truth		ground truth		ground truth	
classes		D	H	D	H	D	H	D	H	D	H
confusion matrices	D	227	51	213	27	211	26	216	21	220	25
	H	19	229	33	253	35	254	30	259	26	255
class %		92.3	81.8	86.6	90.4	85.8	90.7	87.8	92.5	89.4	91.1
overall %		86.7		88.6		88.4		90.3		90.3	

Table 5. Result of classification using neural network-based grading algorithm(Unay, 2005).

The worst classifier in overall performance is the linear discriminant classifier (LDC). Nearest neighbor and fuzziness classifiers have similar results. AdaBoost and SVM classifiers perform best with 90.3% overall recognition. The SVM is more suitable to this system, because it does not require previous training sets.

Chapter 4

Preliminary Explorations

Intelligent image processing techniques and fruit grading algorithms are modified, implemented and tested in this chapter. The experiment and analysis are presented in detail. The flaws and strengths of previous approaches are explored in order to devise a novel algorithm.

4.1 Otsu's Method

4.1.1 Experiment and Analysis

In machine vision and image processing, Otsu's thresholding method (described in Section 2.1.1) is used to automatically perform histogram shape-based image thresholding. The histogram method assumes that there is an average value for the foreground and background pixels. In this case, the foreground is the orange, and the background is the conveying system. Four Examples are presented in this section for testing Otsu's method.

1. Example One: Testing Otsu's method on ripe oranges

Figure 29 shows a ripe orange image with no blemishes.



Fig. 29. Ripe orange sample for testing Otsu's method.

Figure 30 shows three isolated colour channels in natural colours.

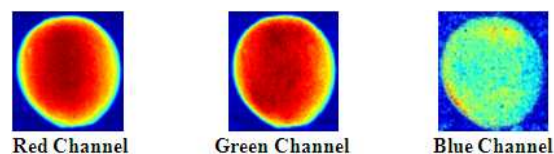


Fig. 30. Isolated colour channels for a ripe orange sample.

The threshold values are computed for three channels separately using Otsu's method.

- Red channel: 96
- Green channel: 46
- Blue channel: 9

Figure 31 shows the processed images after applying the thresholds on three channels separately.

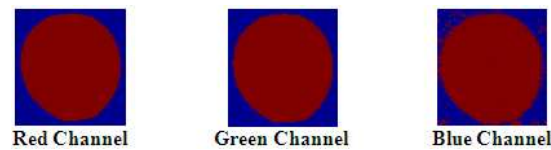


Fig. 31. Example of ripe orange segmentation using Otsu's method.

Otsu's method works perfectly on the isolated red and green channel. The average intensity on the blue channel is low due to the nature of the orange, so some background pixels appear on the right image.

2. Example Two: Testing Otsu's method on blemished ripe oranges

Figure 32 shows a ripe orange image with blemishes.



Fig. 32. Blemished ripe orange sample for testing Otsu's method.

The threshold values are computed for three channels separately using Otsu's method.

- Red channel: 94
- Green channel: 53
- Blue channel: 11

Figure 33 shows the processed images after applying the thresholds on three channels separately.

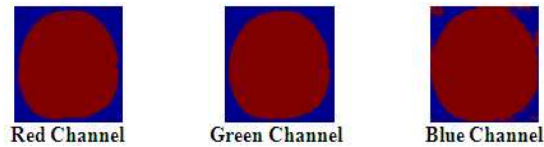


Fig. 33. Example of blemished ripe orange segmentation using Otsu's method.

Small holes appear on the left image which is caused by some low intensity areas on the red channel, such as blemishes and stem. It has no impact on the overall performance. In general, Otsu's method works fine for blemished ripe oranges.

3. Example Three: Testing Otsu's method on unripe oranges

Figure 34 shows an unripe orange image with no blemishes.

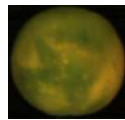


Fig. 34. Unripe orange sample for testing Otsu's method

Figure 35 shows three isolated colour channels in natural colours.

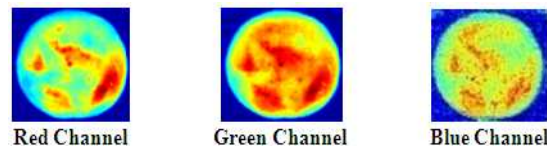


Fig. 35. Isolated colour channels for unripe orange sample

The threshold values are computed for three channels separately using Otsu's method.

- Red channel: 55
- Green channel: 50
- Blue channel: 11

Figure 36 shows the processed images after applying the thresholds on three channels separately.

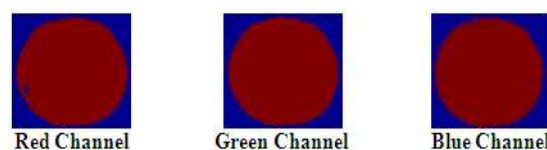


Fig. 36. Example of unripe orange segmentation using Otsu's method.

Small holes appear on the left and right images which are caused by different reasons. It is evident to see that Otsu's method is able to identify some low intensity areas on the orange skin, such as blemishes. It seems like the blemish is easier to be detected on the red channel. In general, Otsu's method works fine for unripe oranges.

4. Example Four: Testing Otsu's method on blemished unripe oranges

Figure 37 shows an unripe orange image with blemishes.

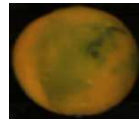


Fig. 37. Blemished unripe orange sample for testing Otsu's method.

The threshold values are computed for three channels separately using Otsu's method.

- Red channel: 67
- Green channel: 48
- Blue channel: 11

Figure 38 shows the processed images after applying the thresholds on three channels separately.

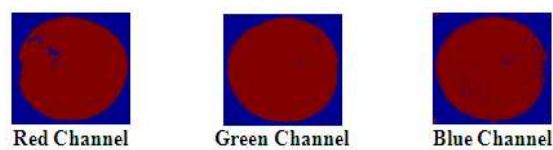


Fig. 38. Example of blemished unripe orange segmentation using Otsu's method.

Holes on the left image are caused by the colour transition areas on the red channel. Holes on the middle image are caused by the low intensity areas on the green channel, such as blemishes. Holes on the right image are not important in this case. It is evident to see that the intensity variation on the green channel is more important than the others for blemish identification.

4.1.2 Algorithm Refinements

Otsu's method works fine for the segmentation of orange and conveying system, however, the cost of Otsu's method is computationally expensive. Constant thresholds can be used to replace Otsu's method. The brightness of the conveying system is very low compared with the orange skin, and the best contrast appears on the red channel. Figure 39 shows the result of orange segmentation on the red channel using a constant threshold 50.

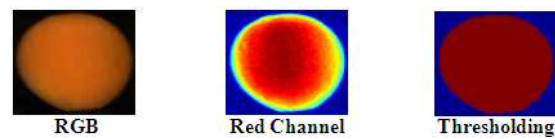


Fig. 39. Orange segmentation on the red channel using a constant threshold value fifty.

One hundred orange images are randomly selected from the database. The comparison is made between Otsu's method and constant thresholds. The assumption is that Otsu's method is one hundred percent accurate on object segmentation. Orange segmented by a constant threshold is compared with the one segmented using Otsu's method. False and true positive rate are selected as the statistical judgment.

1. False Positive Rate

False positive rate is the proportion of negative instances that were erroneously reported as being positive. If a background pixel is erroneously classified as a foreground pixel, then the pixel will be claimed as a false positive instance. All the background pixels are classified as negative instances.

$$fpr = \frac{\text{number of false positive instances}}{\text{total negative instances}} \quad (23)$$

2. True Positive Rate

True positive rate is the proportion of positive instances that reported as being positive. If a foreground pixel is reported as a foreground pixel, then the pixel will be claimed as a true positive instance. All the foreground pixels are

classified as positive instances.

$$tpr = \frac{\text{number of true positive instances}}{\text{total positive instances}} \quad (24)$$

Figure 40 shows the result of orange segmentation using a constant threshold value 50. Low false positive rates with high true positive rates indicate that the constant threshold 50 can achieve a similar result with Otsu's method.

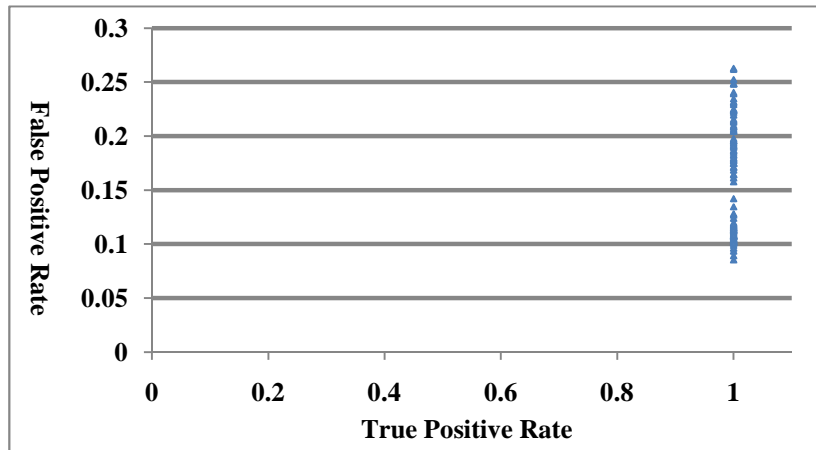


Fig. 40. True and false positive rates using a constant threshold value fifty.

Some background pixels are erroneously classified as foreground pixels. This means the threshold value can be increased. Figure 41 shows the result of orange segmentation using a constant threshold value 70.

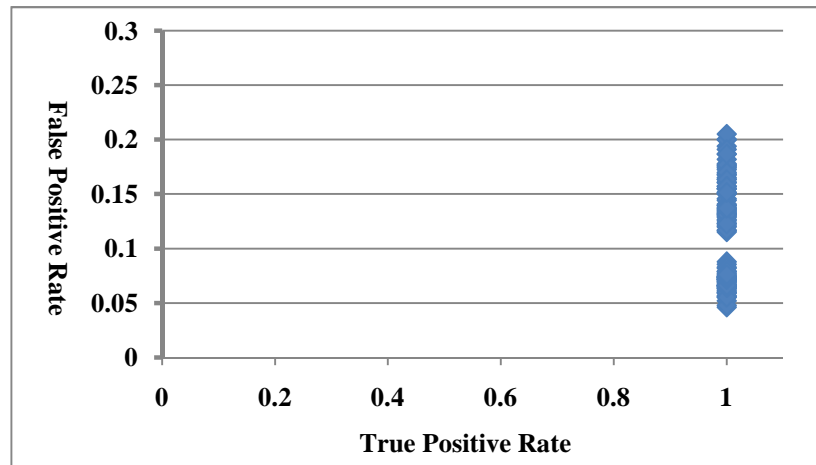


Fig. 41. True and false positive rates using a constant threshold value seventy.

It is evident to see that the false positive rates decrease and the true positive rates stay the same. Most of the background pixels are reclassified correctly without

affecting the foreground pixels. In conclusion, constant thresholds can be used as a replacement of Otsu's method.

4.1.3 Summary

Otsu's method works fine for the segmentation of orange and conveying system, however, it is computational expensive. Constant thresholds can be used as a replacement of Otsu's method on the red channel. The statistical analysis of true and false positive rates shows that most of the pixels belong to the orange are classified correctly, and a few pixels belong to the conveying system are misclassified. The threshold value has to be changed accordingly when the lighting condition is changed. The following steps have to be followed in order to find the best threshold value.

1. Adjust the camera and light source.
2. Select one hundred orange samples randomly.
3. Apply a constant threshold on the red channel.
4. Compare the result with Otsu's method using statistical measurements, such as true and false positive rates.
5. The best threshold appears when the true positive rates are high and the false positive rates are low.

Once the orange is extracted from the conveying system on the red channel, then a mask can be generated and applied to the green and blue channel separately for orange segmentation. In this way, it is more accurate than applying Otsu's method on the blue channel. Because the average intensity on the blue channel is very low, the foreground and background pixels can be easily mixed up.

4.2 High Speed Vision-Based Quality Grading of Oranges

The algorithms presented by (Recce, Taylor, Piebe & Tropiano, 1996) are tested in this section for getting some experience only. The algorithms implemented are slightly different from the original work (described in Section 3.1) due to the limited resources. The first (histogram analysis) and second (local defect search) stages of processing are tested separately for demonstrate purpose. The combined result of implementing all three stages (described in Section 4.2.3) is also presented.

4.2.1 Histogram Analysis

4.2.1.1 Algorithm Details with Sample Computations

The histogram analysis of normalized pixel values is targeted on the red and green channels only. The assumption is that a good orange has normally distributed red and green colour components. A constant threshold value seventy is applied on the red channel for orange segmentation purpose. State-of-the art histogram-based features are extracted from the isolated red and green channels separately, such as mean, variance, skewness, kurtosis, energy and entropy. The pixel values on both red and green channels are normalized between one and nine. The histogram is calculated based on the scaled grey levels. Figure 42 shows an example of the scaled histogram on the red channel.

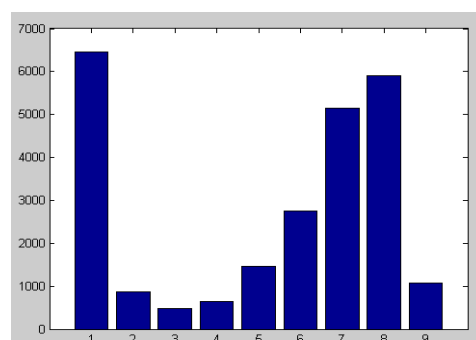


Fig. 42. Example of scaled histogram.

The approximate probability density P for each scaled grey level i is computed

using the following formula:

$$P_i = \frac{\text{number of pixels at each grey level}}{\text{total number of pixels}} \quad (25)$$

Figure 43 shows an example of computed probability densities for each scaled grey level.

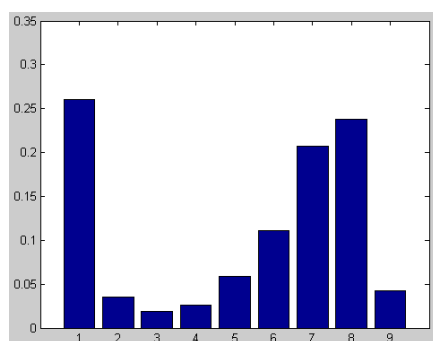


Fig. 43. Probability densities for each scaled grey level.

Six features are extracted based on the histogram obtained from the red and green channels separately.

1. Mean

$$\mu = \sum_{i=1}^9 P_i i \quad (26)$$

2. Variance

$$\sigma = \sum_{i=1}^9 P_i (i - \mu)^2 \quad (27)$$

3. Skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable.

$$\text{skewness} = \sqrt{\sigma}^{-3} \sum_{i=1}^9 P_i (i - \mu)^3 \quad (28)$$

4. Kurtosis is a measure of the “peakedness” of the probability distribution of a real-valued random variable.

$$\text{kurtosis} = \sqrt{\sigma}^{-4} \sum_{i=1}^9 P_i (i - \mu)^4 \quad (29)$$

5. Energy

$$\text{energy} = \sum_{i=1}^9 P_i^2 \quad (30)$$

6. Entropy is a statistical measure of randomness that can be used to characterize the texture of the input image.

$$entropy = \sum_{i=1}^9 P_i \ln P_i \quad (31)$$

4.2.1.2 Neural Network Classifier

The network neurons are constructed as follow:

1. Twelve input neurons.
2. Two output neurons: $P1$ and $P2$.
 - Good Orange Class: [1 -1]
 - Blemished Orange Class: [-1 1]
3. Five hidden neurons.

The input layer has twelve neurons and combines information from the red and green histograms. Two output classes are predefined, such as good orange class and blemished orange class. Oranges classified into the blemished orange class are passed to the second stage (local defect search) for a more detailed analysis. The hidden layer has five neurons. This is to ensure the number of training samples is at least ten times larger than the number of weights. Features are randomly shuffled six hundred times before feeding the neural network. This can help to improve the performance of the neural network classifier. A detailed description of the neural network is presented below:

1. Network Type: Feed-forward back-propagation
2. Performance Function: Sum squared error(SSE)
3. Training Function: Gradient descent with momentum and adaptive learning rate(TRAINGDX)
4. Transfer Function: Log-sigmoid transfer function(LOGSIG)

The orange classification is predefined as follows:

1. If the first output neuron $P1$ is positive and higher than $P2$, then the orange

will be classified as a good orange.

2. If the second output neuron $P2$ is positive and higher than $P1$, then the orange will be classified as a blemished orange.

4.2.1.3 Performance

Seventy oranges are selected as the training dataset. They are manually categorized as follows:

1. Fifty of them are good oranges.
2. Twenty of them are blemished oranges.

One hundred oranges are selected as the testing dataset. They are manually categorized as follows:

1. Fifty of them are good oranges.
2. Fifty of them are blemished oranges.

Histogram based features are extracted from the red and green components separately. Twelve features per image are fed into the trained neural network for classification. The result is shown in Table 6.

Fruit Image Types	No. of Images	No. of Correctly Classified Images	No. of Wrongly Classified Images	% of Correct Classification
Good Orange	50	19	31	38%
Blemished Orange	50	31	19	62%

Table 6. Result of implementing histogram-based analysis.

In the first stage of processing (histogram analysis), a fraction of the top quality oranges might be classified as a lower quality band. It is safer to downgrade the quality in the first stage and perform a more detailed analysis in the second stage (local defect search). However the number of correct classifications for blemished

oranges is very low. This means many blemished oranges are incorrectly classified into the good orange class. If that is the case, the algorithm needs to be refined.

4.2.1.4 Algorithm Refinements

Image preprocessing is performed before extracting the histogram-based features from the red and green components. Each pixel value p on the red and green components is replaced by the maximum variance among four directions using the following formula:

$$p_{ij} = \max \left[\frac{p_{(i-3)j} + p_{(i+3)j}}{2}, \frac{p_{i(j-3)} + p_{i(j+3)}}{2}, \frac{p_{(i-3)(j-3)} + p_{(i+3)(j+3)}}{2}, \frac{p_{(i-3)(j+3)} + p_{(i+3)(j-3)}}{2} \right] \quad (32)$$

For an illustration of the inner working of this algorithm with data samples, see Appendix B. Figure 44 shows the result after replacing the pixel values on the red channel. The algorithm works the same as an advanced edge detector. All the noises are eliminated and potential features are remained.

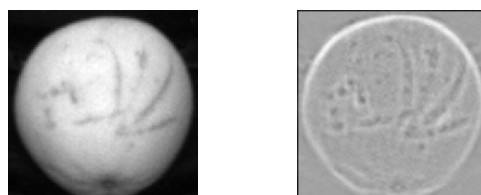


Fig. 44. Example of image preprocessing.

The training algorithm is changed to fuzzy training. A fuzzy training algorithm (Zhou, Li & Jin, 2002) is proposed to improve the pattern recognition performance of neural network as an alternative to the conventional back-propagation training algorithm with hard-decision supervision(HDS). Table 7 shows the classification result.

Fruit Image Types	No. of Images	No. of Correctly Classified Images	No. of Wrongly Classified Images	% of Correct Classification
Good Orange	50	30	20	60%
Blemished Orange	50	34	16	68%

Table 7. Result of implementing modified histogram-based analysis.

The system performance significantly improved after image preprocessing. However the number of correct classifications for both good and blemished oranges is still low.

4.2.2 Local Defect Search

4.2.2.1 Algorithm Details with Sample Computations

The defect is characterized by a discontinuity in skin pigmentation. The typical size of defects N is set to be 30×30 . Five operators (described in Section 3.1.3.2) are applied to the regions on the red and green component separately. The convolution is only applied to arbitrarily partitioned regions, which is aimed to reduce the computational cost. Figure 45 shows two partitioned normal and defected regions with a typical size of 30×30 . Some regions selected for training are the same, which differ only in the selection of the start point.



Fig. 45. Sample regions selected with a typical size of 30×30 .

Block-wise features are extracted from the partitioned regions as additional features, which are considered to be useful for local texture classification. There are two types of block-wise features, such as mean block variance and squared block variance. The mean block variance mbv is defined as follows:

$$mbv = \frac{\sum_{i=0}^{NN} \sum_{j=0}^{NN} m_{ij}}{NN} \quad m_{ij} = \frac{\sum_{b=0}^7 (p_{ij} - P_b)^2}{8} \quad (33)$$

where p_{ij} is the pixel value within the region($N \times N$), P_b is the eight neighborhood-pixels of p_{ij} which is defined as follows:

$$P_b = [p_{(i-1)(j-1)} \quad p_{(i-1)j} \quad p_{(i-1)(j+1)} \quad p_{i(j+1)} \quad p_{(i+1)(j+1)} \quad p_{(i+1)j} \quad p_{(i+1)(j-1)} \quad p_{i(j-1)}]$$

The squared block variance sbv is defined as follows:

$$sbv = \frac{\sum_{i=0}^{NN} \sum_{j=0}^{NN} (m_{ij} - mbv)^2}{mbv} \quad (34)$$

For an illustration of the inner working of this algorithm with data samples, see Appendix C.

4.2.2.2 Neural Network Classifier

The neural network constructed is the same as the one described in histogram analysis (described in Section 4.2.1.2). Oranges classified into the blemished orange class may be passed to the third stage (described in Section 3.1.3.3) for a more detailed analysis. In the second stage (local defect search), the stem is traded as defects. If the defect has a similar size with a typical stem, then the defect definitely will be analyzed in the third stage (described in Section 3.1.3.3).

4.2.2.3 Performance

Five operators (described in Section 3.1.3.2) are applied to the partitioned regions on the red and green channels separately. Block-wise features are extracted as additional features from the red and green channels separately. The result after classification is shown in Table 8.

Fruit Image Types	No. of Images	No. of Correctly Classified Images	No. of Wrongly Classified Images	% of Correct Classification
Good Orange	50	33	17	66%
Blemished Orange	50	20	30	40%

Table 8. Result of implementing modified local defect search.

The number of correct classifications for blemished oranges is low. Many blemished oranges are misclassified into the good orange class. There are a few possible explorations.

1. Increasing the number of training samples may help to improve the performance.
2. The typical size of defects N might be too big or too small.
3. The convolution is only applied to arbitrarily partitioned regions. The defected area might be missed.

4.2.3 Overall Quality Grading System Assessment

Oranges are examined in the following three stages:

1. Histogram Analysis

In the first stage of processing, a fraction of top quality oranges may be classified as a lower quality band. It is safer to downgrade the quality in the first stage and perform a more detailed analysis in the second stage.

2. Local Defect Search

The defect is characterized by a discontinuity in the skin pigmentation. All defect types contribute roughly equally to the final grading decision. The convolution is only applied to arbitrarily partitioned regions. In the second stage, the stem is traded as defects as well.

3. Stem Detection

The stem has a much more regular spatial structure than the defects. The family of Zernike moments is a powerful technique for stem detection. Zernike moments are very sensitive to circular symmetries and invariant under rotation.

Three neural network classifiers are trained for each stage separately. The workflow is described in Figure 46.

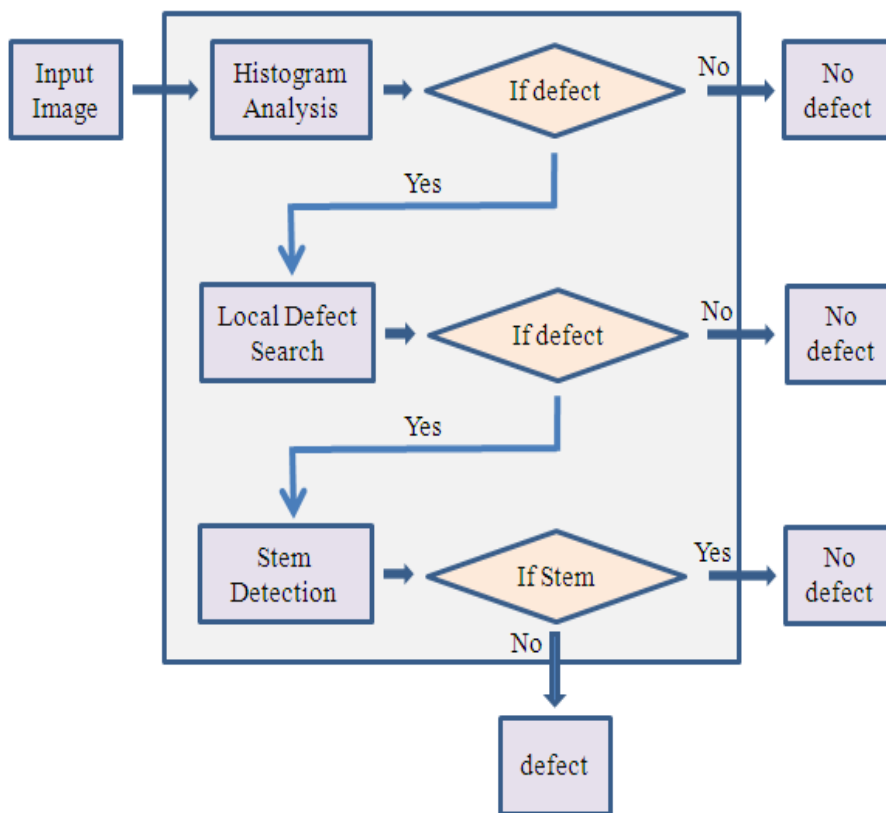


Fig. 46. System workflow.

One hundred oranges are selected as the testing dataset. They are manually categorized as follows:

1. Fifty of them are good oranges.
2. Fifty of them are blemished oranges.

The final result after classification is shown in Table 9.

Fruit Image Types	No. of Images	No. of Correctly Classified Images	No. of Wrongly Classified Images	% of Correct Classification
Good Orange	50	32	18	64%
Blemished Orange	50	30	20	60%

Table 9. Result of implementing improved vision-based grading algorithm.

The classification result is poor. There are a few suggestions which may help to improve the performance.

1. Increasing the number of training samples for each stage.
2. Add some new features.
3. The convolution may apply to more regions on each image.
4. The typical size of defect N can be readjusted.

4.3 Citrus Fruit External Defect Classification

The algorithms introduced by (Vijayarekha & Govindaraj, 2006) are tested in this section. The algorithms implemented might be slightly different from the original work (described in Section 3.2) due to the limited resources.

4.3.1 Algorithm Details with Sample Computations

Image preprocessing is performed in the following order.

1. Convert colour image into grey scale image.
2. Grey scale image is de-noised using the median filter.
3. Orange is cropped to its size.

Figure 47 shows a sample orange image before and after the image preprocessing.

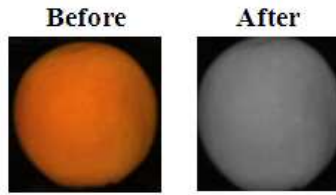


Fig. 47. Sample image before and after preprocessing.

The decomposition (two levels with Daubechies ten wavelet) splits both the approximation and detail windows of the first level of decomposition resulting to sixteen sub-windows. Figure 48 is an example of the derived sixteen sub-windows after decomposition. Mean and standard deviation of the wavelet coefficients are computed for each of the sixteen sub-windows (described in Section 3.2.2).

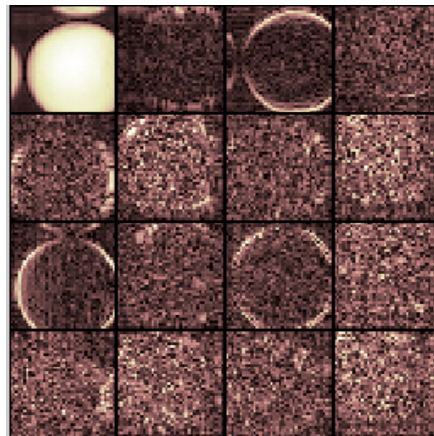


Fig. 48. Example of decomposition resulting to 16 sub-windows.

4.3.2 Neural Network Classifier

The network neurons are constructed as follows:

1. Thirty-two input neurons.
2. Two output neurons: $P1$ and $P2$.
 - Good Orange Class: $[1 \ -1]$
 - Blemished Orange Class: $[-1 \ 1]$
3. Ten hidden neurons.

Features f extracted from the sub-windows are normalized between zero and one using the following formula:

$$nf = \frac{f - \underset{1 \leq F \leq n}{\text{mix } F}}{\underset{1 \leq F \leq n}{\text{max } F} - \underset{1 \leq F \leq n}{\text{mix } F}} \quad (35)$$

where F is a set of features, and n is the total number of features. Normalized features nf are randomly shuffled six hundred times before feeding to the neural network.

A detailed description of the neural network is presented below:

1. Network Type: Feed-forward back-propagation
2. Performance Function: Sum squared error(SSE)
3. Training Function: Gradient descent with momentum and adaptive learning rate(TRAINGDX)
4. Transfer Function: Bipolar tangent sigmoid non-linear transfer function(TANSIG)

The orange classification is predefined as follows:

1. If the first output neuron $P1$ is positive and higher than $P2$, then the orange will be classified as a good orange.
2. If the second output neuron $P2$ is positive and higher than $P1$, then the orange will be classified as a blemished orange.

4.3.3 Overall Citrus Fruit Defect Classification System Assessment

Seventy oranges are selected as the training dataset. They are manually categorized as follows:

1. Fifty of them are good oranges.
2. Twenty of them are blemished oranges.

One hundred oranges are selected as the testing dataset. They are manually categorized as follows:

1. Fifty of them are good oranges.
2. Fifty of them are blemished oranges.

Wavelet-based features are extracted from the images and normalized. Features are fed into the trained neural network for classification. Table 10 shows the result after classification.

Fruit Image Types	No. of Images	No. of Correctly Classified Images	No. of Wrongly Classified Images	% of Correct Classification
Good Orange	50	16	34	32%
Blemished Orange	50	41	9	82%

Table 10. Result of implementing external defect classification algorithm.

The number of correct classifications for blemished oranges is high. However, a lot of good oranges are actually misclassified into the blemished orange class. There are a few suggestions which may help to improve this system.

1. Increasing the number of training samples.
2. Reconsider the features extracted from the sixteen sub-windows. Kurtosis, energy and entropy are useful features which can be extracted as well.

Chapter 5

Novel Algorithms on Orange Grading System

5.1 Central Thesis

There is a limit to most existing statistical-based (Unay, 2005), structural-based (Recce, Taylor, Piebe & Tropiano, 1996), model-based (Chang, et al., 1994) and transform-based (Vijayarekha & Govindaraj, 2006) orange blemish detection algorithms. Any two pixels in an orange image having about the same magnitudes for the red, green and blue channels will almost always be classified as belonging to the same category (either a blemish or not). This however presents a big problem, as depicted in Figure 49, it is possible to have several pixels depicting more or less the same colour channel values, but should belong to different categories. In the figure, pixel A reflects $R=134$, $G=86$, $B=24$ and should be classified as a normal skin. On the other hand, Pixel B is described to have colour channel values very close to Pixel A, but should be classified as belonging to a blemish. In light of this problem, this research utilizes a priori knowledge of the local intensity variation observed on rounded convex objects to classify the aforementioned pixels correctly.

For any rounded convex object, the intensity gradually increases from the edges to the center in a two-dimensional image. The proposed algorithm partitions the given image into eight orange colour classes (described in Section 5.4.1). This in turn would generate different layers/classes using average intensities for a given image (illustrated in Figure 49). These layers are then refined further to eliminate extraneous layers (described from Section 5.4.2 to 5.4.7). Finally, the blemishes are detected by employing a convex hull approach on the topmost layer. Any discontinuities between successive layers/classes will lead to the identification of blemishes (described in Section 5.4.8).

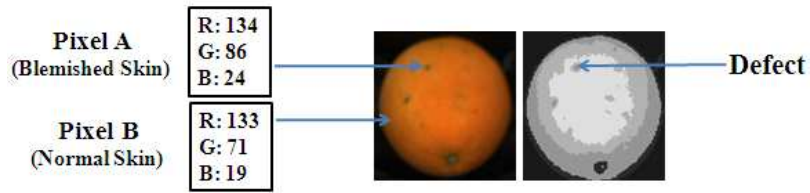


Fig. 49. Blemish detection based on a specified local intensity variation range.

5.2 System Architecture

A block schematic of the orange grading system is shown in Figure 50.

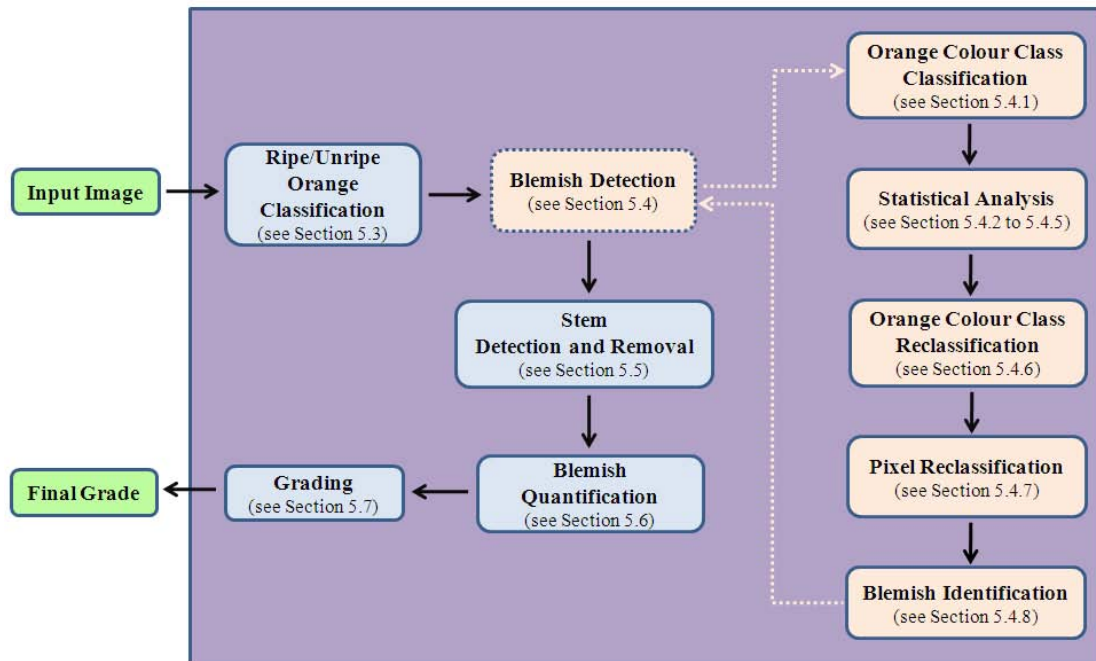


Fig. 50. Block schematic of the novel orange grading algorithm.

In the first stage of processing, oranges are classified into two categories according to the natural skin colour, such as ripe and unripe oranges. This is followed by the blemish detection process which is the core part of the orange grading system. The novel blemish detection algorithm simulates how humans make observations of the local intensity variations phenomenon. Humans do not judge the colours of the orange skin by using absolute pixel values per se (i.e. RGB) but instead consider the neighboring orange surface characteristics. Detecting the stem on the other hand is a bit tricky as it closely resembles blemishes. Therefore, in this research, stem

detection is performed only after identifying all possible blemishes.

Quantifying blemishes is a necessary precursor to grading the oranges. Here, the percentage of blemishes over the whole orange is computed as the main grading feature. In addition, the different quality bands can be adjusted easily according to the requirement set by the market.

5.3 Ripe/Unripe Orange Classification

Oranges can be classified into two categories according to the natural skin colour, such as ripe and unripe oranges. Figure 51 shows two examples of the ripe and unripe oranges.

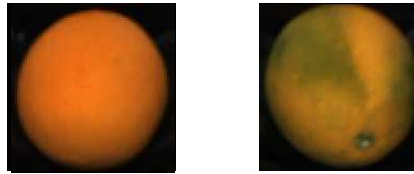


Fig. 51. Ripe and unripe oranges.

Roughly speaking, if more than one quarter of an orange has green colour, then it should be classified as an unripe orange. The blemish detection algorithms for ripe and unripe oranges are slightly different due to the skin colour variation. The formula for orange classification oc is defined as follows:

$$oc = Otsu_r - Otsu_g \quad (36)$$

Otsu's method (described in Section 2.1.1) is applied on the isolated red and green channels separately. The value of oc increases when the skin colour turns to more orange. The threshold t was empirically found to be 40, and the results of applying the classification rules below completely adhere to human visual inspection.

- If oc is greater than t , then the orange will be classified as a ripe orange
- If oc is less than or equal to t , then the orange will be classified as an unripe orange.

A subset of the test set (100 hundred oranges) was selected as the testing dataset. Fifty of them are ripe oranges and fifty are unripe oranges. The computed values for oc are plotted in Figure 52 for all 100 oranges. It is evident that there is a clear boundary between ripe and unripe oranges. All 100 classification results all agree with human visual inspection.

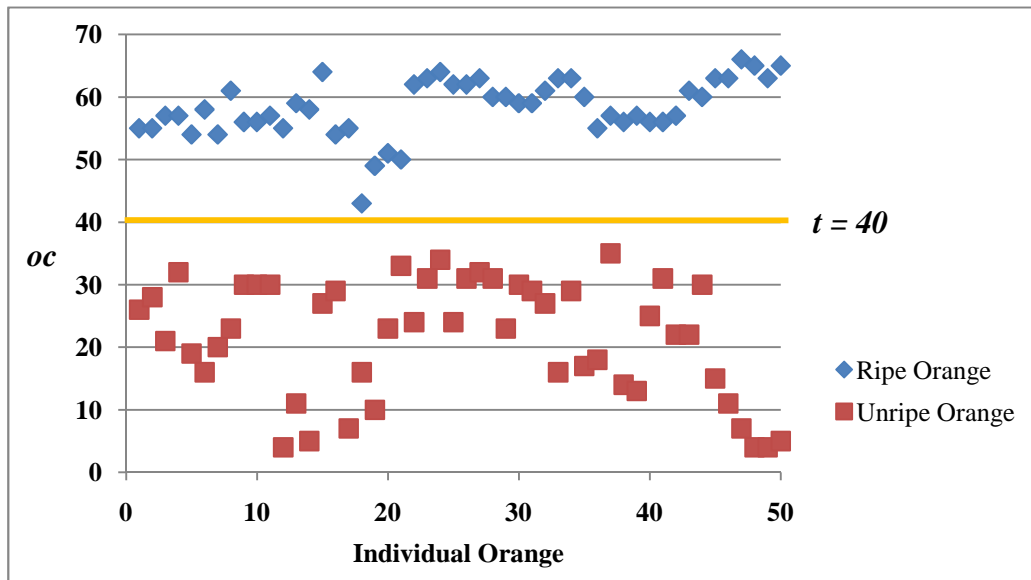


Fig. 52. Orange classification for ripe and unripe oranges.

5.4 Blemish Detection

Oranges are assessed according to the surface texture, such as discoloration, bruising and other blemishes. All blemish types contribute roughly equally to the final grading decision. Blemish detection algorithm is the core part of the orange grading system.

5.4.1 Orange Colour Class

5.4.1.1 Colour Space Exploration

All possible colours can be made from three primary colours red, green and blue. The following example demonstrates that a broad array of colours can be displayed

by using an appropriate combination of red, green and blue intensities. There are three coloured light beams with dimmer switches, one red light, one green light, and one blue light. Three coloured light beams are used to shine three primary colours onto a black wall and dimmer switches are used to adjust the intensity of each primary colour. A representation of the additive colour mixing is shown in Figure 53.



Fig. 53. A representation of the additive colour mixing(Wikipedia, 2008).

A similar result could be achieved using the Paint.net which is a famous tool for image processing. Three coloured light beams are simulated by using the red, green, and blue components separately. The dimmer switch on each light beam is simulated by adjusting the intensity of each component. Figure 54 shows the colour component window in Paint.net.



Fig. 54. Colour component window in Paint.net.

The orange colour class is derived from the combinations of primary colours. Figure 55 shows eight predefined orange colour classes which produce different results in colour and brightness. For instance, pure red colour is classified as an orange red class, and the combination of red and green colour is classified as an orange yellow class. Each of these classes may contain many intensity levels from the weakest to strongest. The average intensity of each orange colour class is a very useful feature in statistics.

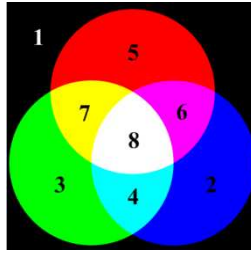


Fig. 55. Predefined orange colour classes.

5.4.1.2 Derived Formula

An RGB image consists of three colour channels, red, green and blue channel. Each channel can be manipulated separately from the others. A channel can be used to generate a grayscale image of the same size as the RGB image. Otsu’s method (described in Section 2.1.1) is made to operate independently on each of the colour channels, and assumes that the image to be thresholded contains two classes of pixels (i.e. foreground=1 and background=0).

It is adamant to custom-build a colour class that is especially designed for identifying the blemishes. As mentioned earlier, looking at the absolute pixel value per se does not suffice for accurate blemish detection. In this section, we introduce the orange colour class that is derived from the combination of colour primaries. Table 11 shows the detailed classifications for eight orange colour classes.

Orange Colour Class				
Class No.	Class Name	Red	Green	Blue
1	Background Class	0	0	0
2	Blue Class	0	0	1
3	Green Class	0	1	0
4	Cyan Class	0	1	1
5	Red Class	1	0	0
6	Magenta Class	1	0	1
7	Yellow Class	1	1	0
8	White Class	1	1	1

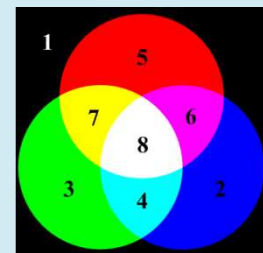


Table 11. Orange colour class classification.

“1” means presence of the primary colour, and “0” means absence. For instance, if a pixel is a member of the orange red class, then its value on the red channel is “1”, and on both green and blue channels are “0”.

The following formula shows the method of classifying a single pixel into one of the eight predefined orange colour classes. In turn, all the pixels will be classified as one of the eight colour classes.

$$ClassNo_{(i,j)} = 4Red_{(i,j)} + 2Green_{(i,j)} + Blue_{(i,j)} + 1 \quad (37)$$

for $1 \leq i \leq Rows, 1 \leq j \leq Columns$

5.4.1.3 General Algorithm

Input: RGB Image.

Output: Orange class distribution matrix.

Data: I = input RGB image, M = orange class distribution matrix, $rows$ = the number of rows in image I , $cols$ = the number of columns in image I , R = isolated red channel, G = isolated green channel, B = isolated blue channel, ro = threshold on the red channel, go = threshold on the green channel, bo = threshold on the blue channel, RB = converted binary image for the red channel, GB = converted binary image for the green channel, BB = converted binary image for the blue channel, p = pixel in image I , c = computed class number for pixel p .

- 1 $(R, G, B) = \text{IsolateColourChannels}(I)$
- 2 $M = \text{Zeros}(rows, cols)$
- 3 $(ro, go, bo) = \text{Otsu}(R, G, B)$
- 4 $RB = \text{ThresBinary}(R, ro)$
 $GB = \text{ThresBinary}(G, go)$
 $BB = \text{ThresBinary}(B, bo)$
- 5 **foreach** *pixel p in image I* **do**
 $c = \text{ComputeClassNo}(p)$

5.4.1.4 Missing Orange Colour Class

There are eight orange colour classes. Some of them might be not available due to the nature of the orange texture. The orange class distribution matrix can be used to analyze the distribution and availability of the orange colour classes. Table 13 is an instance of the orange class distribution matrix. Class five is not available in this case.

Orange Class Distribution Matrix					
1	3	8	2	2	2
4	8	8	6	8	8
8	6	4	1	8	8
8	8	7	8	8	7
8	4	8	4	8	1
1	5	6	8	8	4

Table 13. Example of the orange class distribution matrix.

The class availability is analyzed below for ripe and unripe oranges separately.

1. Experiment One: Ripe orange

One hundred oranges are randomly selected from the database. Fifty of them are blemished oranges and fifty are good oranges. After a class availability test, the result is shown in Table 14. The orange red and magenta classes are not available for all the selected ripe oranges. In other words, the pure red colour and the combination of red and blue colours are not available.

Class Availability Test for Ripe Oranges			
Class No.	Class Name	Color Component	Availability
1	Background Class	None	Yes
2	Blue Class	Blue	Yes
3	Green Class	Green	Yes
4	Cyan Class	Blue, Green	Yes
5	Red Class	Red	No
6	Magenta Class	Red, Blue	No
7	Yellow Class	Red, Green	Yes
8	White Class	Red, Green, Blue	Yes

Table 14. Orange colour class availability test for ripe oranges in the database.

2. Experiment Two: Unripe Orange

Fifty oranges are randomly selected from the database. Twenty of them are blemished oranges and twenty are good oranges. After a class availability test, the orange red and magenta classes are not available for all the selected unripe oranges as well. This may refer to the light source and nature of the orange texture.

5.4.1.5 The Order of Orange Colour Classes

Each of these orange colour classes produces different results in colour and brightness. Due to the observed nature of the orange skin colours, the different orange colour class described in this section is ordered incrementally according to their average intensities. For instance, the combination of red and green colour is brighter than the combination of blue and green colour. For other fruits, the order of the orange colour classes has to be changed accordingly to match the incremental sequence.

5.4.1.6 Summary

All possible colours can be made from three primary colours red, green and blue. Pixels in a given RGB image are classified into one of the eight orange colour classes. The class number for each pixel is stored in the orange class distribution matrix for further analysis.

Some of the orange colour classes might be not available, such as orange red and magenta classes. This is caused by the light source and nature of the orange texture. The brightness of the orange colour classes increases from class one to eight. This order is specially designed for orange only.

5.4.2 Orange Class Mean

5.4.2.1 Overview

Mean has two related meanings in statistics, such as arithmetic mean and population mean. Arithmetic mean is the one selected in this algorithm and often simply called the “mean”. For a given data set, the average is the sum of the measurements divided by the number of measurements and to compute a number as being the average. Changing the order of the measurements does not affect the final result. The formula of mean μ is defined as follows:

$$\mu = \frac{\sum_{i=1}^n P_i}{n} \quad (38)$$

where P is the pixel value, and n is the number of pixels. The orange class mean refers to a measure of the average intensity of each orange colour class. The mean for each orange colour class is computed on three channels separately for a given RGB image. In turn, for each orange colour class, there will be three class means associated with it. Computed orange class means are stored in the orange class mean matrix which consists of eight rows and three columns.

5.4.2.2 General Algorithm

Input: RGB Image, Orange class distribution matrix.

Output: Orange class mean matrix.

```
1 foreach orange colour class do
2   foreach isolated colour channel do
      ComputeClassMean()
      StoreComputedClassMean()
   end
end
```

The derived orange class distribution matrix stores the class information for each pixel in a given RGB image. For all the pixels that belong to the same class, compute the average of pixel values on three channels separately. Table 15 is an example of the orange class mean matrix with specified class numbers.

Orange Class Mean Matrix			
Class No.	Red Channel	Green Channel	Blue Channel
1	23	13	6
2	52	31	18
3	145	73	12
4	0	0	0
5	0	0	0
6	0	0	0
7	225	113	13
8	229	114	23

Table 15. Example of the orange class mean matrix.

5.4.2.3 Algorithm Refinements

The distribution of pixel values within each orange colour class is considered to be a normal distribution. The normal distribution describes data that cluster around the mean. Sometimes a set of numbers might contain outliers. The outlier is the intensity of a pixel which is much lower or higher than the others. The outliers are erroneous data caused by different reasons, such as leaves on the conveying system. The outliers affect the accuracy of the computed orange class means.

There are three steps to recalculate and improve the orange class means.

1. Sort the pixel values.
2. Discard an equal amount of data at the high and the low ends. For most statistical applications, five to twenty-five percent of the ends are discarded. Ten percent of the ends are discarded in this algorithm.
3. Compute the orange class means using the remaining data.

5.4.2.4 Effects of Illumination Intensity Variations on Ripe Orange Skin

Figure 56 is a ripe orange image with two blue lines drawing across the centre.

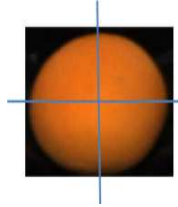


Fig. 56. Example of ripe orange image with two blue lines drawing across the center.

Table 16 shows the selected pixel values along the horizontal blue line on red, green and blue channels separately.

Selected Pixel Values along the Horizontal Blue Line													
Cols	10	20	30	40	50	60	70	80	90	100	110	120	130
R	12	16	104	157	183	204	255	218	176	145	97	20	14
G	10	11	50	77	84	96	104	101	82	68	51	14	12
B	6	9	10	18	18	24	27	25	16	14	13	11	7

Table. 16. Intensity variations along the horizontal blue line.

The average intensity gradually increases from the edges to the center of the image due to its rounded convex contour. The following are some of the observed properties:

1. The most significant intensity variation occurs on the red channel. The pixel value varies between 12 and 255.
2. The average intensity on the green channel is lower than the red channel. The pixel value varies between 10 and 104.
3. The average intensity on the blue channel is very low. The pixel value varies within a very small range.

Figure 57 shows the intensity variation along the vertical blue line on red, green and blue channels separately.

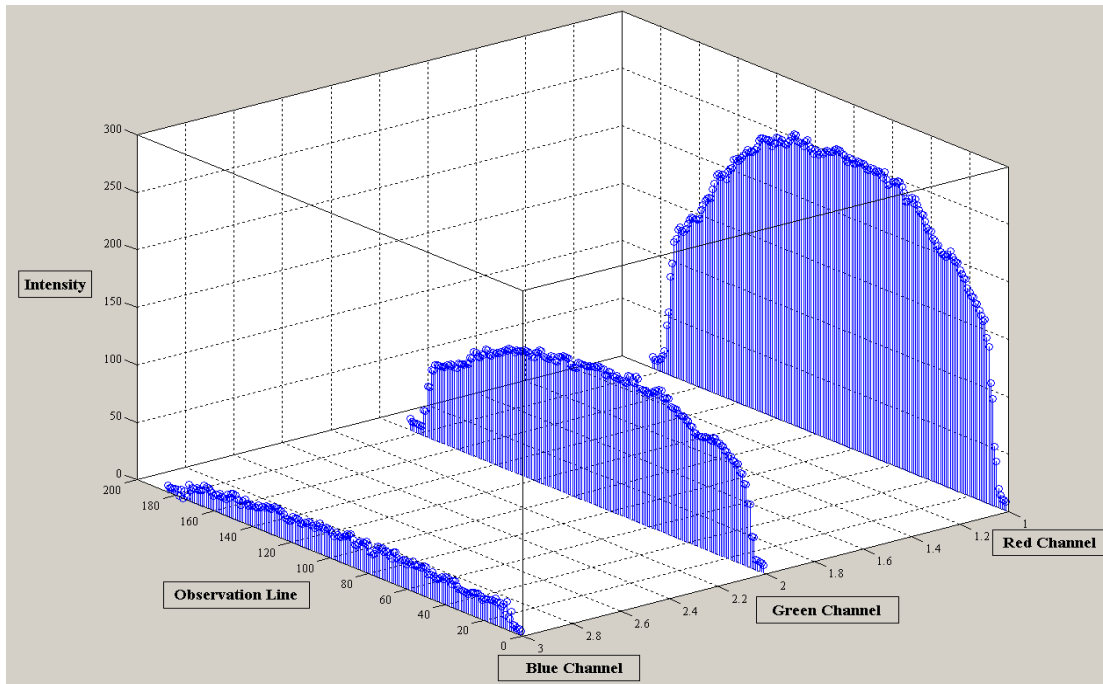


Fig. 57. Intensity variations along the vertical blue line.

Due to the natural of the orange skin colour, the red and green channels of an orange image are more important for statistical analysis. Figure 58 shows two pixels selected from the normal and blemished skin separately.

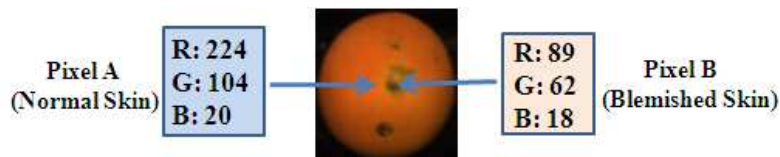


Fig. 58. Two pixels selected from normal and blemished skin separately.

The blemished skin is usually darker than the normal skin. The intensity varies on three channels differently. The absolute pixel value dropped 135 on the red channel, 42 on the green channel, and 5 on the blue channel.

5.4.2.5 Effects of Illumination Intensity Variations on Unripe Orange Skin

Figure 59 is an unripe orange image with two red lines drawing across the centre.

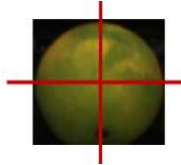


Fig. 59. Example of unripe orange image with two red lines drawing across the center.

Table 17 shows the selected pixel values along the horizontal red line on red, green and blue channels separately.

Selected Pixel Values along the Horizontal Red Line													
Cols	10	20	30	40	50	60	70	80	90	100	110	120	130
R	70	87	96	113	121	137	144	114	110	105	89	77	68
G	71	84	101	107	111	115	118	108	102	100	81	73	70
B	6	19	24	27	28	27	29	25	25	19	18	16	10

Table 17. Intensity variations on the horizontal red line.

The average intensity gradually increases from the edges to the center of the image.

The following are some of the observed properties:

1. The average intensity on the red channel is lower compared with the one for ripe oranges in Table 16. The pixel value varies between 68 and 144.
2. Due to the natural of the unripe orange texture, the intensity variation on the green channel becomes more important in this case. The pixel value varies between 70 and 118.
3. The intensity variation on the blue channel is not important. The average intensity is very low.

Figure 60 shows the intensity variation along the vertical red line on red, green and blue channels separately.

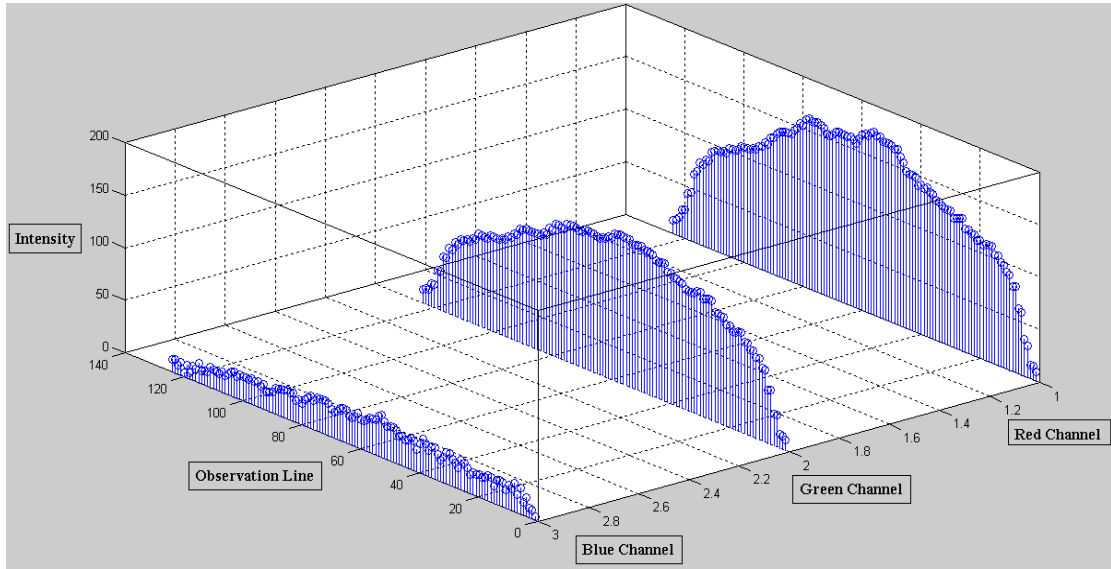


Fig. 60. Intensity variations along the vertical red line.

Figure 61 shows three pixels selected from the normal and blemished skin separately.

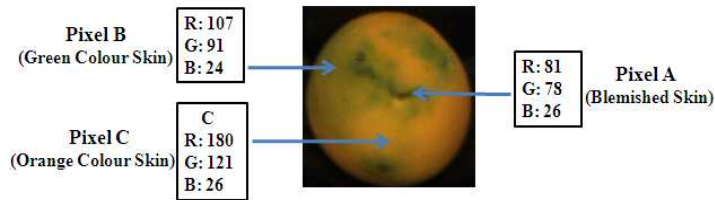


Fig. 61. Three pixels selected from the normal and blemished skin separately.

The intensity variation among three pixels is analyzed below.

1. Pixel C to Pixel B

- The average intensity of orange colour skin is usually brighter than the green colour skin. The intensity variation between different colour skins should not be considered as blemishes.

2. Pixel C to Pixel A, Pixel B to Pixel A

- The intensity of blemished skin is usually darker than the normal skin.
- The intensity variations on both red and green channels are important figures for blemish detection.

5.4.2.6 Summary

Orange class mean is a measurement of the central tendency for each orange colour class. Sometimes a set of numbers might contain outliers. The outlier is the intensity of a pixel which is much lower or much higher than the others. The outliers are erroneous data caused by different reasons, such as leaves on the conveying system. Ten percent of the ends are discarded in this algorithm.

The average intensity gradually increases from the edges to the center of the image. Due to the nature of the orange texture, the intensity variation on the red and green channel is more important for statistical analysis.

5.4.3 Orange Class Standard Deviation

5.4.3.1 Overview

In statistics, standard deviation is a simple measure of the spread of a dataset. Standard deviation can often find the story behind the data, such as the tightness of data samples that are clustered around the mean. A low standard deviation indicates that all the pixel values are very close to the same value (class mean), while the high standard deviation indicates that all the pixel values are clearly more spread out across a large range of levels. The formula of standard deviation σ is defined as follows:

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (P_i - \mu)^2}{n}} \quad (39)$$

where P is the pixel value, μ is the class mean, and n is the number of pixels in the class. There are eight orange colour classes and three colour channels in a given RGB image, so twenty-four orange class standard deviations will be computed in total. Orange class standard deviation is a statistical measure of the dispersion of the class members.

5.4.3.2 General Algorithm

```

Input: RGB Image, Orange class distribution matrix.
Output: Statistical analysis matrix

1  foreach orange colour class do
2  foreach isolated colour channel do
    ComputeClassStandardDeviation()
    StoreComputedClassStandardDeviation ()
  end
end

```

For all the pixels that belong to the same class, compute the class standard deviations on three channels separately for a given RGB image. Table 18 is an example of the statistical analysis matrix on the first dimension with specified class numbers.

Statistical Analysis Matrix– First Dimension								
Class No.	1	2	3	4	5	6	7	8
1	20	--	--	--	--	--	--	--
2	--	40	--	--	--	--	--	--
3	--	--	15	--	--	--	--	--
4	--	--	--	6	--	--	--	--
5	--	--	--	--	70	--	--	--
6	--	--	--	--	--	66	--	--
7	--	--	--	--	--	--	32	--
8	--	--	--	--	--	--	--	18

Table 18. Example of the statistical analysis matrix.

The statistical analysis matrix is a three-dimensional array with eight rows and columns. The elements of a three-dimensional array can be thought of as a set of two-dimensional arrays. The first, second, and third dimensions are used to store the computed data on the red, green and blue channels separately. For instance, the class standard deviation 20 of class one on the red channel is stored at the first row and column on the first dimension.

5.4.3.3 Summary

Orange class standard deviation is a simple measure of the spread of a dataset. A low class standard deviation indicates that all the pixel values are very close to the class mean, while a high class standard deviation indicates that all the pixel values are clearly more spread out across a large range of levels. The orange class standard deviation is derived from the original standard deviation in statistics.

The computed class standard deviations are stored in the statistical analysis matrix for further analysis.

5.4.4 Between-Class Squared Mean Difference

5.4.4.1 Overview

Between-class squared mean difference is a numerical description of how far apart the orange colour classes are. In mathematics, there are two common methods to compute the distance between two objects, e.g., absolute difference and squared difference. Absolute difference is a numerical value without regard to its sign. The following example demonstrates how the absolute difference works.

$$A = 8, B = 6,$$

$$Diff1 = A - B = 2, \quad Diff2 = B - A = -2,$$

$$absDiff1 = |A - B| = 2, \quad absDiff2 = |B - A| = 2$$

Squared difference is a squared numerical value without regard to its sign. The following example demonstrates how the squared difference works.

$$sqrtDiff1 = (A - B)^2 = 4, \quad sqrtDiff2 = (B - A)^2 = 4$$

Absolute difference and squared difference basically works the same on the way of regarding to its sign. However, the squared difference enlarged the distance between two objects. The between-class squared mean difference *smd* is defined as

follows:

$$smd = (\mu_A - \mu_B)^2 \quad (40)$$

where A and B represents any one of the eight orange colour classes, and $A \neq B$. smd is a measure of the squared difference between two class means. A small-valued smd indicates that two classes are similar to each other.

5.4.4.2 General Algorithm

```

Input: Orange class mean distribution matrix.
Output: Statistical analysis matrix

1  foreach orange colour class do
2  foreach isolated colour channel do
    ComputeBetweenClassSquaredMeanDifference ()
    StoreComputedBetweenClassSquaredMeanDifference ()
  end
end

```

Compute the between-class squared mean differences for each orange colour class on three channels separately. Table 19 is an example of the statistical analysis matrix on the first dimension with specified class numbers.

Statistical Analysis Matrix – First Dimension								
Class No.	1	2	3	4	5	6	7	8
1	--	18	10	21	34	56	12	7
2	18	--	3	44	12	15	27	40
3	10	3	--	22	41	53	9	11
4	21	44	22	--	18	23	33	62
5	34	12	41	18	--	15	21	33
6	56	15	53	23	15	--	44	2
7	12	27	9	33	21	44	--	8
8	7	40	11	62	33	2	8	--

Table 19. Statistical analysis matrix with computed between-class variances.

For instance, the smd 18 between class one and two is stored at the second row and first column, which is the same as the one at the first row and second column.

In statistics, a matrix of covariances between elements of a random vector is called covariance matrix (Besson, Bidon & Tourneret, 2008). The formula is defined as follows:

$$C_{(i,j)} = C_{(j,i)} \quad (41)$$

Table 20 shows an example of the covariance matrix. Data along the blue diagonal line follows the symmetry principle, e.g., $C(2,1)$ is the same as $C(1,2)$ in terms of the between-class squared mean difference.

Covariance Matrix								
	1	2	3	4	5	6	7	8
1	--	$C(1,2)$	$C(1,3)$	$C(1,4)$	$C(1,5)$	$C(1,6)$	$C(1,7)$	$C(1,8)$
2	$C(2,1)$	--	$C(2,3)$	$C(2,4)$	$C(2,5)$	$C(2,6)$	$C(2,7)$	$C(2,8)$
3	$C(3,1)$	$C(3,2)$	--	$C(3,4)$	$C(3,5)$	$C(3,6)$	$C(3,7)$	$C(3,8)$
4	$C(4,1)$	$C(4,2)$	$C(4,3)$	--	$C(4,5)$	$C(4,6)$	$C(4,7)$	$C(4,8)$
5	$C(5,1)$	$C(5,2)$	$C(5,3)$	$C(5,4)$	--	$C(5,6)$	$C(5,7)$	$C(5,8)$
6	$C(6,1)$	$C(6,2)$	$C(6,3)$	$C(6,4)$	$C(6,5)$	--	$C(6,7)$	$C(6,8)$
7	$C(7,1)$	$C(7,2)$	$C(7,3)$	$C(7,4)$	$C(7,5)$	$C(7,6)$	--	$C(7,8)$
8	$C(8,1)$	$C(8,2)$	$C(8,3)$	$C(8,4)$	$C(8,5)$	$C(8,6)$	$C(8,7)$	--

Table 20. Example of the covariance matrix.

5.4.4.3 Summary

Between-class squared mean difference is a numerical description of how far apart the orange colour classes are. Absolute difference and squared difference are two common methods to compute the distance between two objects. However, the squared difference enlarged the distance between two objects. The computed between-class squared mean differences are stored in the statistical analysis matrix.

5.4.5 Closest Neighbor Class

5.4.5.1 Overview

The task of this section is to identify which classes are close to each other. The word close in terms of colour expression means that two classes have a similar visual impact. In Figure 62, the closest neighbor class of Class One is Class Four. Class Two and Class Three have very different colours compared to the others, so they are defined as standalone classes.

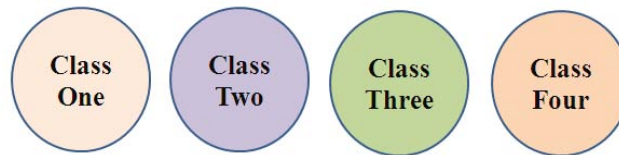


Fig. 62. Similarity of the orange colour classes.

Figure 63 illustrates the relationship among these four classes.



Fig. 63. Demonstration of the closest neighbor class.

Of course, the similarity among different classes cannot be measured by human eye in a real time application. The statistical analysis matrix stores the information which is especially designed for this task. The orange class standard deviation and between-class variation are used as the measure of the closest neighbor class.

5.4.5.2 Mean Selection for Skewed Distributions

The following example demonstrates how the average distance between class one and two is computed.

$$\begin{aligned} \text{distR}[1,2] &= 80 && \text{Distance on the Red Channel between Class One and Two,} \\ \text{distG}[1,2] &= 70 && \text{Distance on the Green Channel,} \end{aligned}$$

$distB[1,2] = 6$ *Distance on the Blue Channel,*

$averageDist[1,2] = (80 + 70 + 6) / 3 = 52$

52 is the average distance between class one and two. In statistics, the arithmetic average of a set of values is one of the most commonly used statistical measurements. However, it is less accurate for skewed distributions. For example, the arithmetic average of six values: 20, 18, 17, 16, 10, 1 is:

$$(20 + 18 + 17 + 16 + 10 + 1) / 6 = 13.6667$$

The average is skewed downwards by a few numbers with very small values, however, the majority numbers are bigger than 10. The data stored in the statistical analysis matrix among three dimensions are considered to be a set of skewed values.

- Data in the first dimension derived from the isolated red channel is the most important.
- Values in the first and second dimensions are usually much bigger than the third dimension.

This is caused by the nature of the orange texture. The skin colour of an orange is more likely to be red, maybe a little bit green, and almost no blue. To compute a more accurate average for a set of skewed values is a challenge. Quadratic mean, geometric mean and harmonic mean are compared in the following examples using the same data, e.g., 80, 70, 60.

1. Example One: Quadratic Mean

Quadratic mean is also called power mean.

- Compute the power of each element in the dataset and sum the results.

$$80^2 + 70^2 + 6^2 = 11336$$

- Divide the sum by the number of elements in the dataset.

$$11336 / 3 = 3779$$

- Compute the square root.

$$sqrt(3779) = 61$$

2. Example Two: Geometric Mean

- Multiply all the elements in the dataset.

$$80 * 70 * 6 = 33600$$

- Compute the one-third power of the multiplication.

$$33600^{1/3} = 32$$

3. Example Three: Harmonic Mean

- Divide each element by one and sum the results.

$$1/80 + 1/70 + 1/6 = 0.1935$$

- Divide the value by the number of elements.

$$3 / 0.1935 = 15$$

Quadratic mean is more towards to the maximum elements in the dataset, harmonic mean is more towards to the minimum elements, and arithmetic mean and geometric mean are in between. Table 21 shows the computed means.

Max Element	80
Quadratic Mean	61
Arithmetic Mean	52
Geometric Mean	32
Harmonic Mean	15
Min Element	6

Table 21. Example of four different means.

The mean selected for this algorithm is the quadratic mean. The formula is defined as follows:

$$qm = \sqrt{\frac{\sum_{n=1}^3 V_n^2}{3}} \quad (42)$$

where V is the value stored in the statistical analysis matrix, and n represents the first, second and third dimensions in the order of 1, 2 and 3 separately.

5.4.5.3 General Algorithm

```

Input: Statistical analysis matrix.
Output: Closest neighbor array.
Data:  $QM$  = temporary matrix used to store computed quadratic means.  $SA$  =
statistical analysis matrix.

1 foreach element in statistical analysis matrix SA do
    ComputeQuadraticMean()
    StoreComputedQuadraticMean()
end

2 foreach orange colour class do
    FindClosestNeighborClass()
    StoreComputedClosestNeighborClass()
end

```

1. Compute the quadratic mean for each element in the statistical analysis matrix. Table 22 is an example of the statistical analysis matrix with three dimensions. The quadratic mean 7.5 is computed using the numbers 10, 8 and 2, and stored in the temporary quadratic mean matrix at the corresponding position. In real time application, the quadratic mean matrix is a two-dimensional array with eight rows and columns.

Statistical Analysis Matrix									Quadratic Mean Matrix		
First Dimension			Second Dimension			Third Dimension					
10	--	--	8	--	--	2	--	--	7.5	--	--
--	--	--	--	--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--	--	--	--	--
--	--	--	--	--	--	--	--	--	--	--	--

Table 22. Example of the computation for quadratic means.

2. Find the closest neighbor class for each orange colour class using the quadratic mean matrix. Table 23 is an example of the quadratic mean matrix with eight rows and columns. The closest neighbor class of class one is class eight 8, which is derived by finding the column/row number with minimum value in the

first column/row. The closest neighbor array is an array with eight elements, which is designed to store the closest neighbor class number for each orange colour class.

Quadratic Mean Matrix									Closest Neighbor Array	
	1	2	3	4	5	6	7	8	Class No.	Closest Neighbor
1	20	18	3	33	21	67	98	2	1	8
2	18	--	--	--	--	--	--	--	2	--
3	3	--	--	--	--	--	--	--	3	--
4	33	--	--	--	--	--	--	--	4	--
5	21	--	--	--	--	--	--	--	5	--
6	67	--	--	--	--	--	--	--	6	--
7	98	--	--	--	--	--	--	--	7	--
8	2	--	--	--	--	--	--	--	8	--

Table 23. Example of finding the closest neighbor class.

5.4.5.4 Closest Neighbor for Ripe and Unripe Oranges

Table 24 is an example of the orange class mean matrix for a ripe orange from the real application. The class means between class seven and eight are very similar for all three channels, and they are closest neighbors. Class five and six are not available, so the means are filled with zeros.

```
classMean: 8x3 double =
 16.2049  9.7060  4.1241
 43.4271 26.5159 14.1981
135.1845 65.8570  8.5428
165.4578 81.8189 15.2790
      0      0      0
      0      0      0
204.4545 100.1364  9.6818
203.5200 105.1776 17.2501
```

Table 24. Example of the orange class mean matrix for a ripe orange.

Table 25 is an example of the orange class mean matrix for an unripe orange. Class three and four are closest neighbors. Class five and six are not available as well.

```
classMean: 8x3 double =
 15.9614 15.9115  6.9266
 31.4272 33.4868 18.9087
 50.5250 53.9909 12.8225
 49.9139 54.2826 18.7108
      0      0      0
      0      0      0
 64.5042 67.4639 13.4606
 89.2344 88.1236 24.0544
```

Table 25. Example of the orange class mean matrix for an unripe orange.

The skin colour variation causes the difference in the class neighbor classification.

5.4.5.5 Summary

Some of the orange colour classes could be very similar to each other. It is important to find the closest neighbor for each orange colour class. The data stored in the statistical analysis matrix is considered to be a set of skewed values, so the average values among three dimensions are computed using the quadratic mean. The skin colour variation causes the difference in the class neighbor classification.

5.4.6 Class Reclassification

5.4.6.1 Overview

The closest neighbor array presents the relationship among the eight orange colour classes. Table 26 is an example of the closest neighbor array with specified class numbers for demonstration purpose.

Closest Neighbor Array	
Class No.	Closest Neighbor
1	1
2	2
3	3
4	0
5	0
6	0
7	8
8	7

Table 26. Example of the closest neighbor array with specified class numbers.

Three types of classes are defined in this section, such as standalone class, missing class and similar class.

1. Standalone Class

A class has no closest neighbor. In Table 26, class one, two and three are standalone classes. Pixels in this class are different from others in terms of colour components and brightness.

2. Missing Class

A class does not exist. In Table 26, class four, five and six are missing classes. Zero in the closest neighbor array indicates that the current class is not available.

3. Similar Class

Two classes are similar to each other. In Table 26, class seven and eight are closest neighbors. They can be merged together to form a new class.

Table 27 shows the newly derived classes after class reclassification. Class one, two and three stay the same, class four, five and six are deleted, and class seven and eight are merged together to form a new class.

Closest Neighbor Array		Class Reclassification	
Class No.	Closest Neighbor	Class No.	Original Class
1	1	1	1
2	2	2	2
3	3	3	3
4	0	4	8,7
5	0		
6	0		
7	8		
8	7		

Diagram annotations: A bracket labeled "Delete" groups rows 4, 5, and 6. A bracket labeled "Merge" groups rows 7 and 8.

Table 27. Class reclassification.

5.4.6.2 Colour Components and Brightness

Due to the nature of the orange texture, the orange yellow and white classes are usually similar to each other. The blue component for an orange image is very minor. The orange colour classes can also be considered as clusters. Pixels within the same cluster are similar to each other in terms of colour components and brightness. Clustering is a common technique for statistical data analysis.

Figure 64 simulates the derived new classes in Table 27 using clusters. The process of assigning a pattern into one of the pre-defined clusters is called classification. A pattern is a set of measurements, such as intensity level.

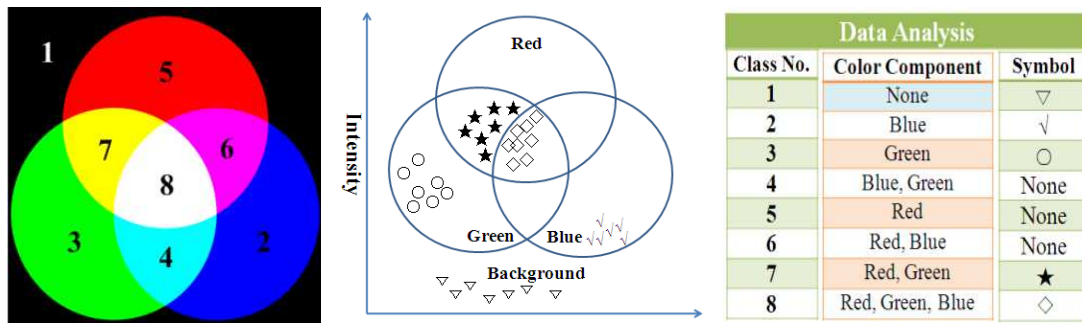


Fig. 64. Clusters.

There are four distinct clusters in Figure 64.

1. Cluster one contains the pixels with very low intensities. Pixels in this cluster belong to the background.
2. Cluster two contains the pixels with pure green colour.
3. Cluster three contains the pixels with pure blue colour. Not many pixels belong to this cluster due to the nature of the orange texture.
4. Cluster four contains the pixels with the combination of red and green colour. Cluster five contains the pixels with the combination of red, green and blue colour. Pixels in cluster four and five have similar colour components and brightness, so they can be merged together to form one cluster.

5.4.6.3 General Algorithm

Input: Closest neighbor array, Orange class mean matrix.

Output: New class mean matrix.

```

1  foreach orange colour classes do
    ReclassifyOrangeColourClasses()
    ComputeNewClassMean()
    StoreComputedNewClassMean()
end

```

Reclassify the orange colour classes according to the closest neighbor array.

- Leave standalone classes as they are.
- Delete all the classes which are not available.
- Merge all the similar classes together to form a new class.

Compute the new class mean for two similar classes (i.e. Class A and Class B).

The formula is defined as follows:

$$NewClassMean = \frac{NumPixel_A ClassMean_A + NumPixel_B ClassMean_B}{NumPixel_A + NumPixel_B} \quad (43)$$

where *NumPixel* is the number of pixels, and *ClassMean* is the original class mean.

The newly computed class means are stored in the new class mean matrix. The new class mean matrix is similar to the orange class mean matrix, which differ only in the number of rows. After class reclassification, the number of new classes is not always the same.

5.4.6.4 Summary

There are three key steps for the class reclassification.

1. Leave standalone classes as they are.
2. Delete all the classes which are not available.
3. Merge similar classes together to form a new class.

The class mean has to be recomputed for newly derived classes. The recomputed class means are stored in the new class mean matrix for pixel reclassification.

5.4.7 Pixel Reclassification

5.4.7.1 Overview

After class reclassification, pixels no longer belong to one of the eight original orange colour classes. Pixels should be reclassified into one of the newly derived classes based on the new class mean matrix. Find which new class each pixel belongs to is the task of this section.

5.4.7.2 Data Analysis

Table 28 is an example of the new class mean matrix.

New Class Mean Matrix			
Class No.	Red Channel	Green Channel	Blue Channel
1	23	13	6
2	52	31	18
3	145	73	12
4	228	113	22

Table 28. Example of the new class mean matrix for demonstration purpose.

For a given pixel p , the pixel values on three channels are 127, 55 and 16 separately. The differences between the pixel value (127) and class means (23, 52, 145, 228) on the red channel are 104, 75, 18, 101, and the pixel value is more close to class three. The differences on the blue channel are 10, 2, 4 and 6, and the pixel value is more close to class two. The differences on the red channel are bigger and more important than the others.

5.4.7.3 General Algorithm

Input: New class mean matrix, RGB image.

Output: Pixel reclassification matrix.

Data: NCM = new class mean matrix, $classMean$ = class mean in new class mean matrix NCM , I = input RGB image, p = pixel in image I .

```
1  foreach pixel p in image I do
2    foreach class mean classMean in new class mean matrix NCM do
      ComputeSumSquaredDifference()
    end
3    FindMinSumSquaredDifference()
4    StoreNewPixelValue()
  end
```

Step through each pixel in the RGB image. For each pixel examined, compute the differences separately for each colour channel (i.e. R, G, B) between the pixel value p and class mean μ for each of the three channels. Next, apply the quadratic mean for combining these differences together, and this will give us the final ssd .

The formula is defined as follow:

$$ssd = \sqrt{\frac{\sum_{n=1}^3 (P_n - \mu_n)^2}{3}} \quad (44)$$

where n represents the red, green and blue channels in the order of 1, 2 and 3.

Pixel will be classified as a member of the class with the minimum ssd . The pixel value is replaced by the corresponding class mean on the red channel and stored in the pixel reclassification matrix which is a two-dimensional array with the same size as the RGB image I . For an illustration of the inner working of this algorithm with data samples, see appendix D.

5.4.7.4 Summary

Pixels no longer belong to one of the eight original orange colour classes after class reclassification. Pixels should be reclassified according to the newly derived classes. The differences between the pixel value and class means on the red channel are bigger and more important than the others, so the sum squared difference is selected for the computation of a set of skewed values.

5.4.8 Blemish Identification

5.4.8.1 Overview

The blemishes on the orange are caused by various reasons, such as poor air during transportation, pre/post-harvest diseases and mechanical damages. The task of this section is to identify the blemished areas. Experiments are presented for ripe and unripe oranges separately.

5.4.8.2 Topmost Layer Slicing

The top layer slicing phase of the algorithm is important as this defines the region of inspection. Figure 65 shows a grey scale image generated using the pixel reclassification matrix.

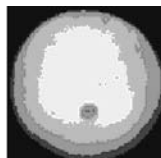


Fig. 65. Image generated using the pixel reclassification matrix.

Only the topmost layer is analyzed and the reasons are explained as follow:

1. The lighting condition is better on the top of the orange.
2. Noise is filtered out, such as background pixels.

3. The orange is rotated on the conveying system, so unprocessed parts can be analyzed in the next image.
4. Reduce unnecessary computations for unstable data.

5.4.8.3 Blemish Segmentation

The blemishes are identified by employing a convex hull approach (described in Section 2.1.3.2) on the topmost layer. Two key steps are described as follow:

1. Segment the blemishes from the topmost layer.
 - Crop the topmost layer, and designate it as $Mask_1$
 - Apply the convex hull technique on the cropped image and designate the result as $Mask_2$.
 - Extract the blemishes by subtracting the cropped image from the convex image (i.e. $Mask_2 - Mask_1$).
2. Count the number of pixels in the segmented image.

Figure 66 illustrates the process of the blemish segmentation.

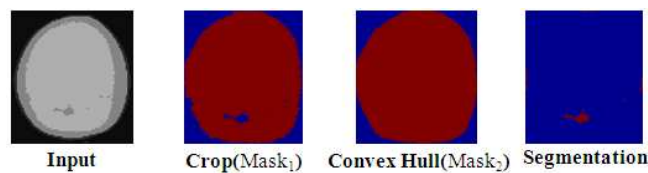


Fig. 66. Segment the blemished area on the very top layer.

5.4.8.4 Refinement of Blemish Segmentation

Figure 67 shows some leftovers in the segmented image which are caused by the rough edges in the cropped image. Experiments show that these little spots around the edges may add up to a relatively big value.

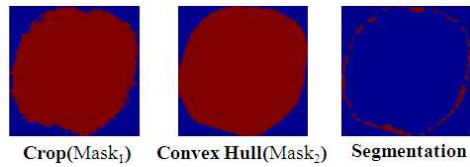


Fig. 67. Spots in the segmented image.

The rough edges in the cropped image should be smoothed using the morphological operators (described in Section 2.1.3) before the convex hull. Figure 68 shows two segmented images before and after smoothing the edges in the cropped image. This step reduces the presence of noise around the edges, while maintaining blemish detection accuracy.

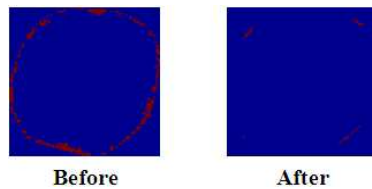


Fig. 68. Segmented image before and after the erode operation.

5.4.8.5 Sample Execution of Blemish Segmentation

This section shows some of the results garnered for a variety of orange grades. Figure 69 shows the blemish identification results for ripe oranges. Here, no blemishes were marked as there is none. The stem was also extracted correctly.

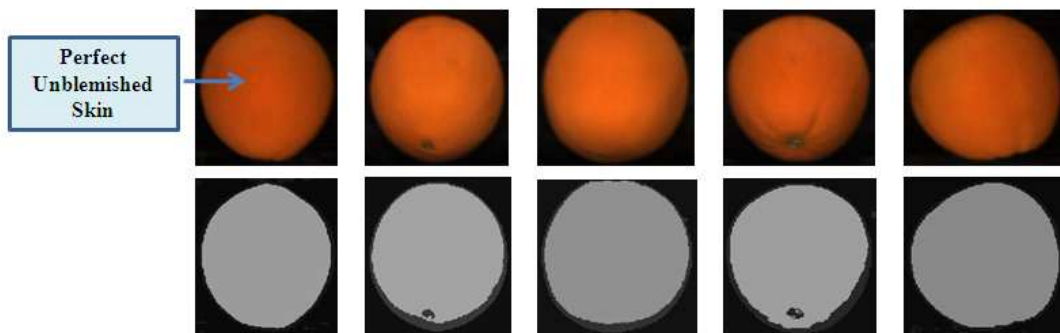


Fig. 69. Blemish identification results for ripe oranges.

Figure 70 shows the blemish identification results for blemished ripe oranges.

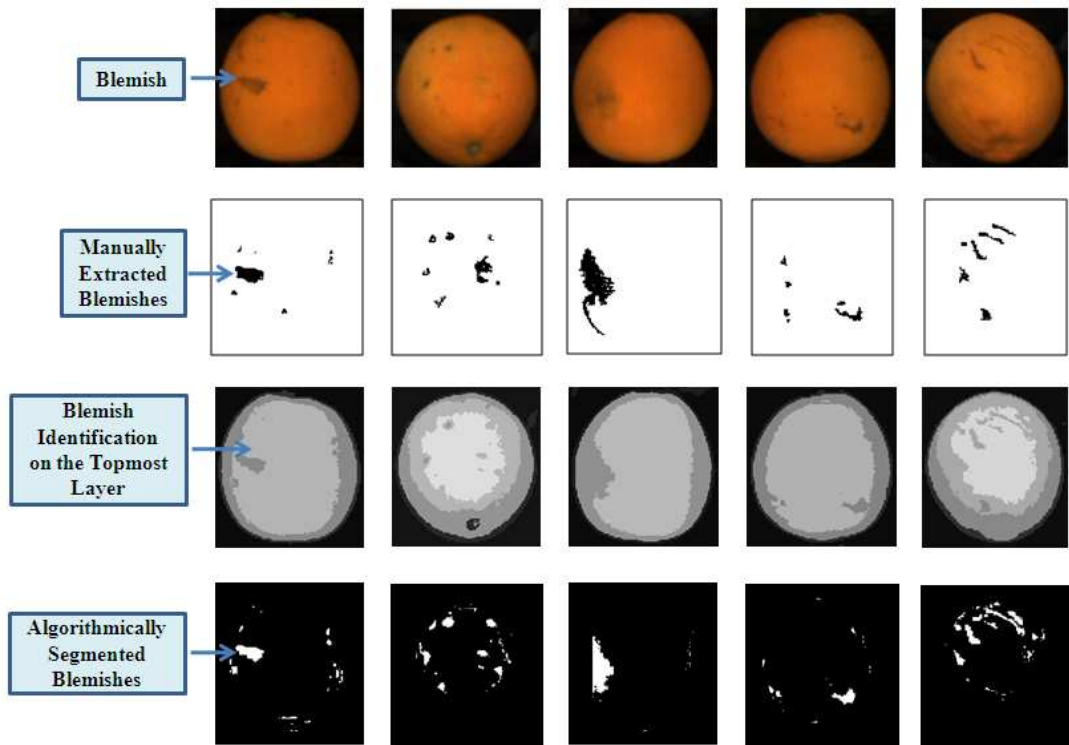


Fig. 70. Blemish identification results for blemished ripe oranges.

Figure 71 shows the blemish identification results for unripe oranges.

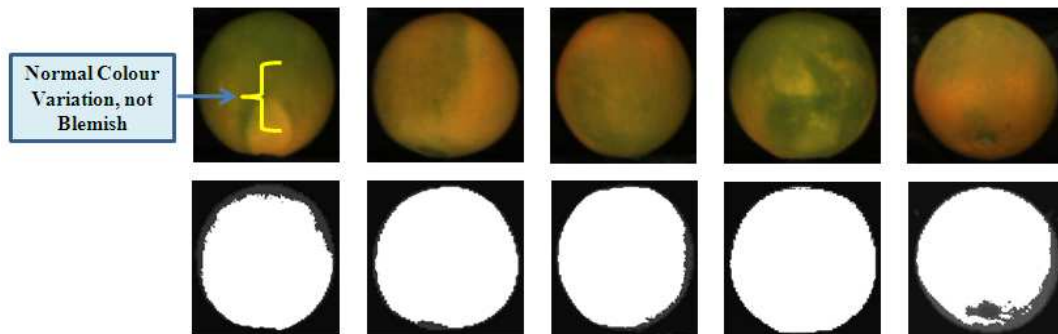


Fig. 71. Blemish identification results for unripe oranges.

Figure 72 shows a series of algorithm executions on a set of photographs captured for a blemished unripe orange. It is evident that the blemishes were accurately identified by the proposed algorithms. The results also prove that the proposed algorithms are robust to confounding colour transition areas (from orange to green and vice-versa).

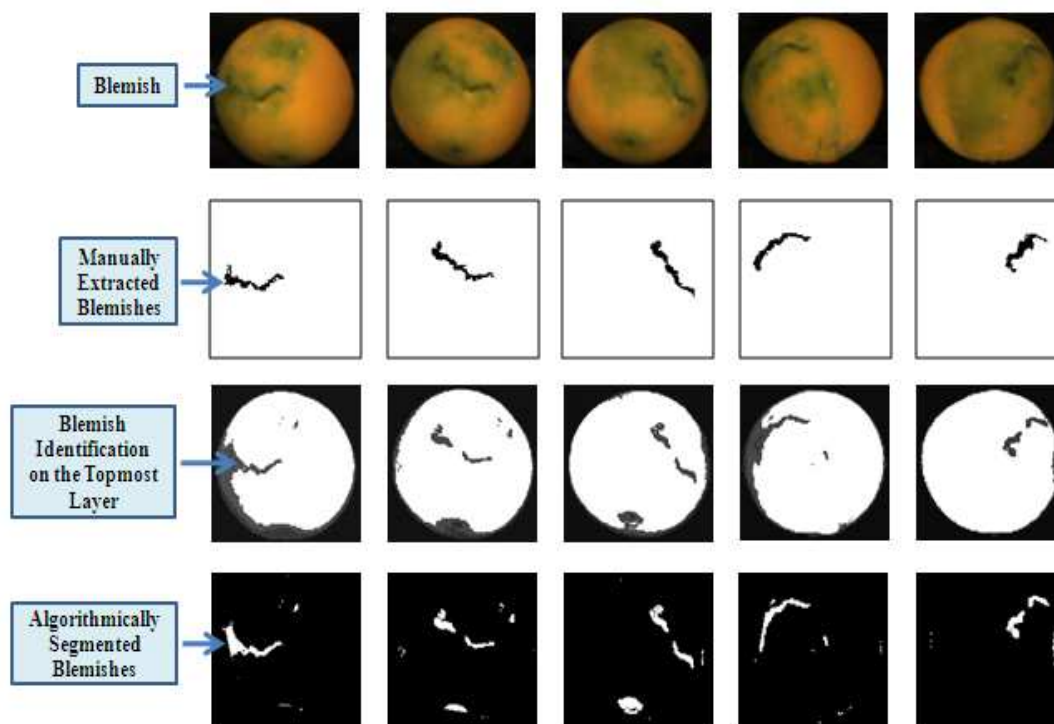


Fig. 72. Blemish identification results for blemished unripe oranges.

5.4.8.6 Experiment and Analysis on Ripe Oranges

Two experiments are presented for good and blemished ripe oranges separately.

1. Experiment One: Good Orange

The skin colour of a good orange is smooth and uniformly distributed, therefore nothing should be detected. In Figure 73, the intensity on the topmost layer is uniformly distributed.

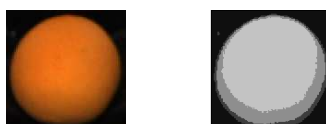


Fig. 73. Result of image processing for a good ripe orange.

The blemish detection algorithm is tested using one hundred oranges. True positive rate is computed for each orange. The algorithm is accurate only if the true positive rate is close to one. The formula is described below:

$$\text{true positive rate} = \frac{\text{number of true positive instances}}{\text{total number of positive instances}} \quad (45)$$

Pixels within topmost layer with the highest intensity will be classified as true positive instances. The total number of positive instances is the number of pixels on the topmost layer.

The computed true positive rates are shown in Figure 74. It is evident to see that the true positive rates are very high. Pixels which are not classified as the true positive instances are mainly caused by the stem. The stem in this stage is treated the same as blemishes.

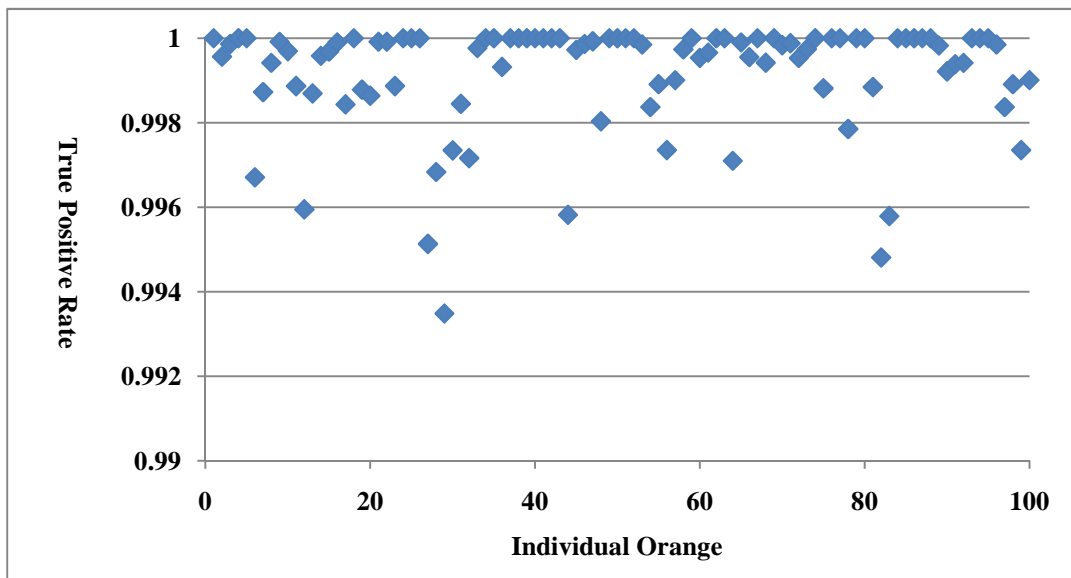


Fig. 74. True positive rates computed based on one hundred good ripe oranges.

2. Experiment Two: Blemished Oranges

Figure 75 illustrates that the intensity on the topmost layer is not uniformly distributed. Holes are identified as blemishes which are usually darker than the normal skin area.

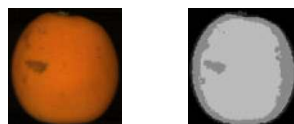


Fig. 75. Result of image processing for a blemished ripe orange.

The blemish detection algorithm is tested on seventy blemished oranges. The blemishes indentified by the algorithm are compared with the one manually

extracted. In Figure 76, the middle image is generated by the algorithm and the right image is extracted manually using the layer in Paint.net.



Fig. 76. Blemish identification testing for blemished ripe orange.

True and false positive rates are computed as the statistical measurement.

- True positive instances are the pixels which are in the blemished area and reported as being positive.
- False positive instances are the pixels which are in the normal skin area and erroneously reported as being positive.

True and false positive rates are plotted in Figure 77 for seventy blemished oranges.

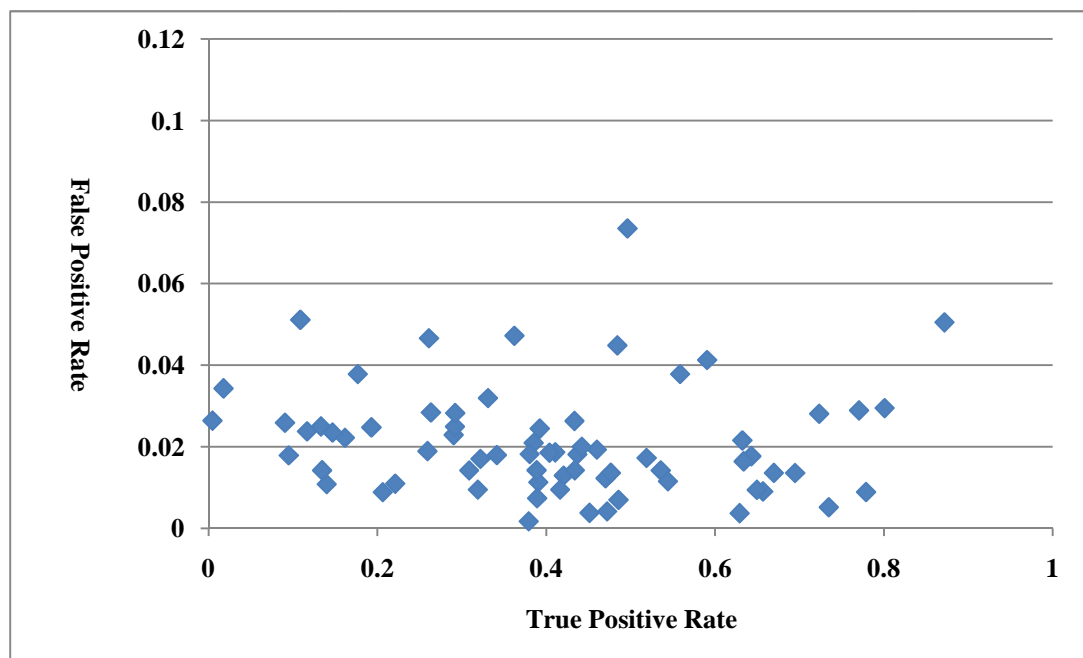


Fig. 77. True and false positive rates plotted for seventy blemished ripe oranges.

The true positive rates are low. There are many human factors which may affect the testing result.

1. The blemishes are marked manually and it might be too big or too small.
2. Some blemishes are not marked but identified by the blemish detection

algorithm. This is caused by the following three reasons:

- The original image is not clear enough.
- The examiner has different thought for the blemishes.
- The examiner forgot to mark some blemishes.

3. Figure 78 shows some blemishes are identified outside of the topmost layer which will not be considered in the current image. The layers are generated dynamically at the run time, so it is hard to estimate which part of the orange is within the topmost layer. A rough estimation is made in this case.

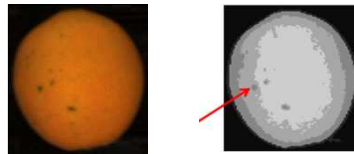


Fig. 78. Blemishes identified outside of the topmost layer.

4. Figure 79 shows some blemishes are hard to differentiate by human eye due to the poor lighting conditions.

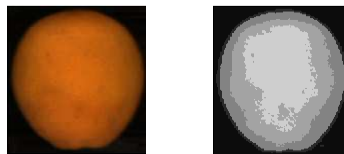


Fig. 79. Some blemishes are hard to detect by human eye.

Due to various negative human factors, the statistical analysis in this section is used for reference only.

5.4.8.7 Experiment and Analysis on Unripe Oranges

Two experiments are presented for good and blemished unripe oranges separately.

1. Experiment One: Good Unripe Orange

Figure 80 shows a processed image with layers painted in colour. Hole A is caused by the skin colour variation from orange to green. Hole B is caused by the

stem at the bottom of the image.

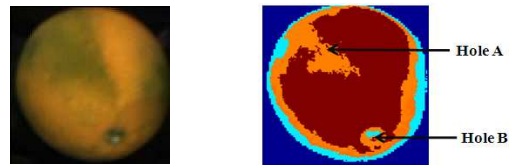


Fig. 80. Example of skin colour variation on an unripe orange.

The colour transaction area should not be identified as blemishes, although the intensity is usually darker on the green skin area. Figure 81 shows the result of merging the top two layers. Hole A is disappeared, and Hole B stays unchanged. The stem is treated as blemishes in this stage.

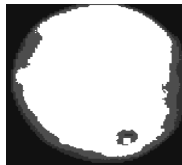


Fig. 81. Result of merging the top two layers.

The blemish detection algorithm is tested on sixty good unripe oranges. The computed true positive rates are plotted in Figure 82. It is evident to see that the true positive rates are very high.

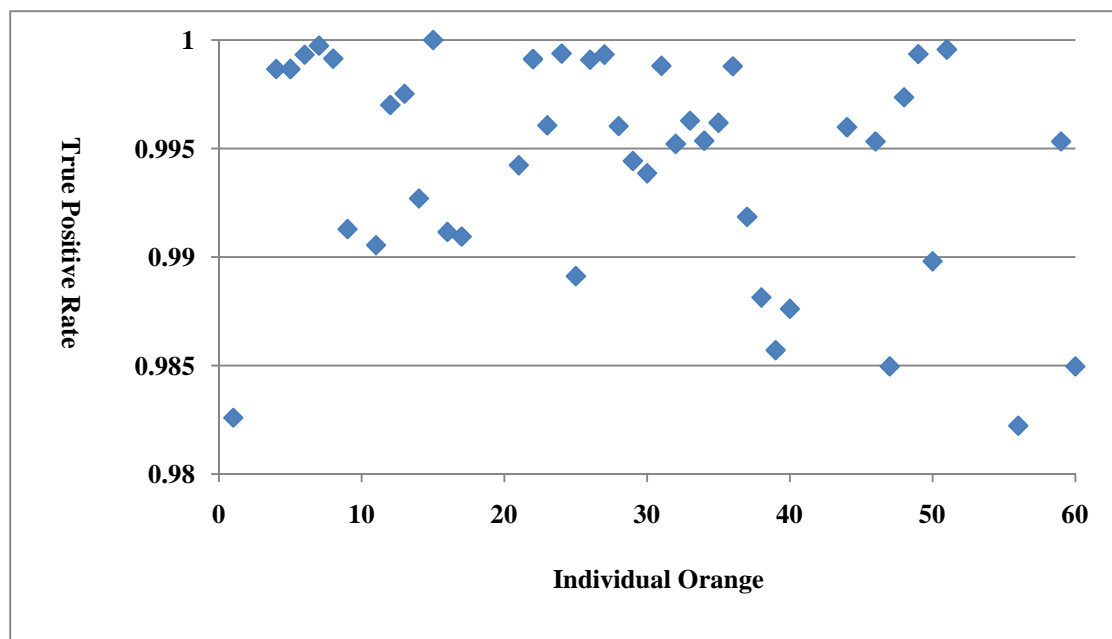


Fig. 82. True positive rates computed based on sixty good unripe oranges.

2. Experiment Two: Blemished Unripe Orange

The average intensity of unripe oranges is lower than the ripe oranges, and the intensity variation between the normal and blemished skin is smaller. Figure 83 shows two images captured for the same orange from different angles. Pixel A and B are traced for inspection purpose. The colour channel values of Pixel B vary a lot when the position is changed.



Fig. 83. Intensity variation for blemished unripe oranges.

Figure 84 illustrates that the blemishes are identified more accurately when the position is changed.



Fig. 84. Blemish identification with different positions.

5.5 Stem Detection and Removal

The stem should be treated differently from blemishes, and therefore should be isolated. Stem isolation is performed basically using Otsu's method (described in Section 2.1.1) operating on the red channel. The result is used as a threshold that segregates the stem from the orange skin.

The basis of this technique is illustrated by example. Figure 85 shows three selected pixels from different skin areas. The colour channel values of a stem are usually smaller than the blemishes. The pixel value varies the most on the red channel.

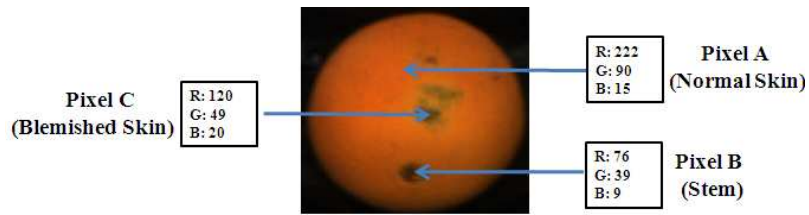


Fig. 85. Intensity variation among three selected pixels.

Figure 86 on the other hand, illustrates how the stem pixels are segmented into the background class (conveying system) using the threshold derived from Otsu's method on the red channel. Holes within the foreground class (orange) will not be classified as blemishes.

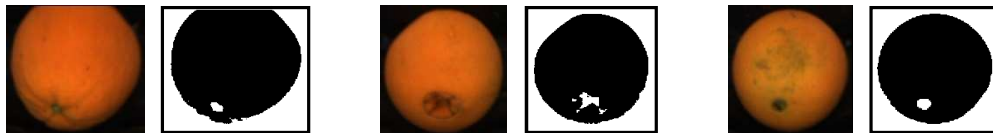


Fig. 86. Stem detection using Otsu's method.

Figure 87 shows some deep blemishes with very low intensities on the orange are also segmented into the background class. However, the misclassified pixels are reduced to the minimum extent.



Fig. 87. Blemishes misclassified as the stem.

Figure 88 points out the deep blemish which causes the misclassification.

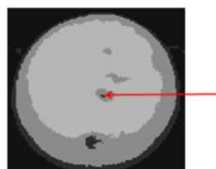


Fig. 88. Depth of the blemish.

5.6 Blemish Quantification

Blemishes are quantified based on the proportion of blemishes found for each orange fruit. We define this as bnp/tnp . On the other hand, the proportion of good orange skin is defined relative to the proportion of blemishes. This is defined as p for each orange fruit as follows:

$$p = 1 - \frac{bnp}{tnp} \quad (46)$$

where bnp is the number of blemish pixels (described in section 5.4.8.3), and tnp the total number of pixels on the topmost layer.

The factors which may affect the accuracy of the system are explored as follow:

1. The blemishes are outside of the topmost layer which is not analyzed in the current image.
2. The rough edges are not smoothed perfectly, and some leftovers are treated as blemishes.
3. The stem is partially removed.

5.7 Grading

The novel algorithm was tested on 170 oranges. 100 were manually classified as good oranges by experts, while 70 were classified as blemished oranges by the same inspectors.

Figure 89 shows the classification results before smoothing the rough edges.

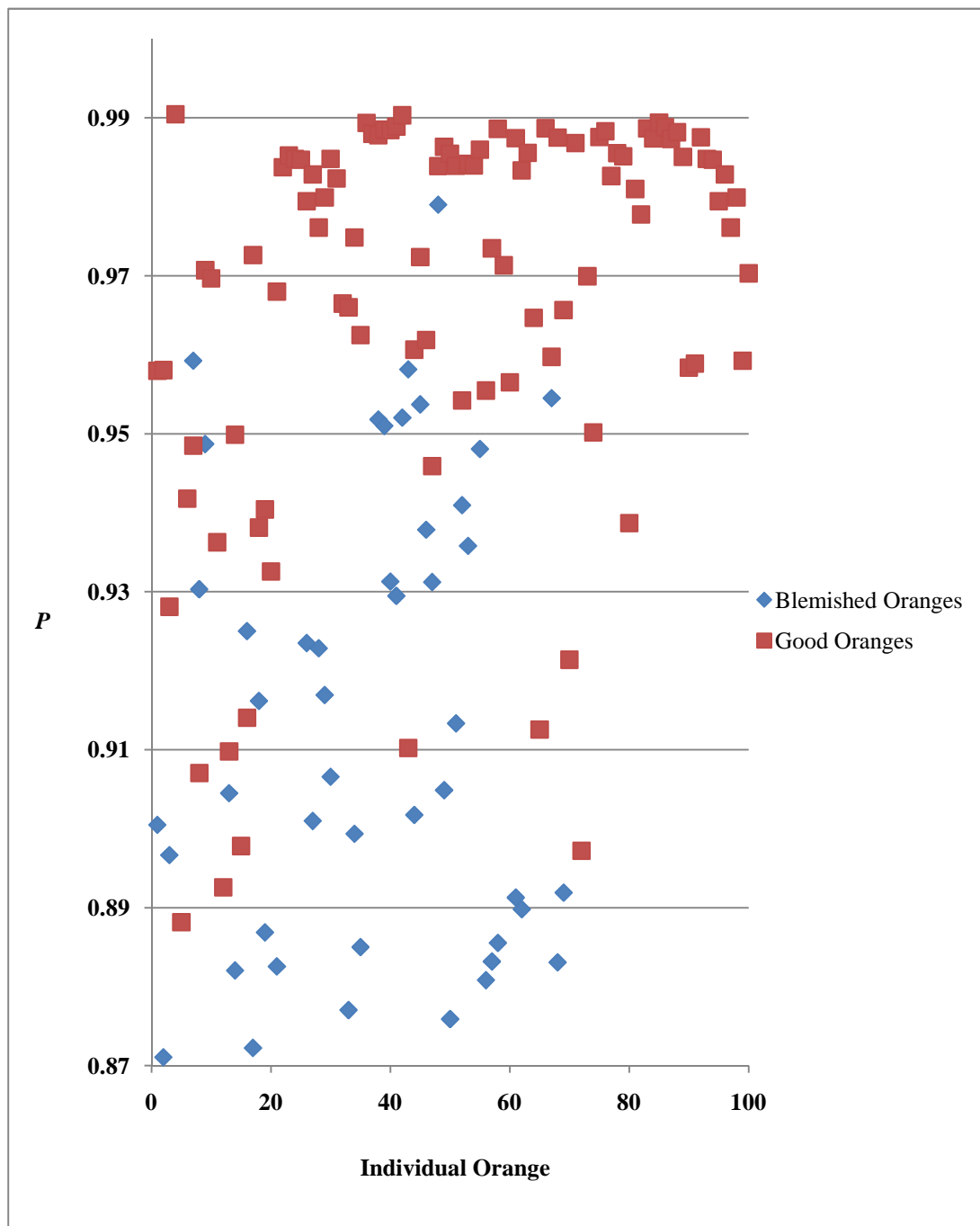


Fig. 89. Classification results before smoothing the rough edges.

Figure 90 shows the classification results after smoothing the rough edges. The performance is improved.

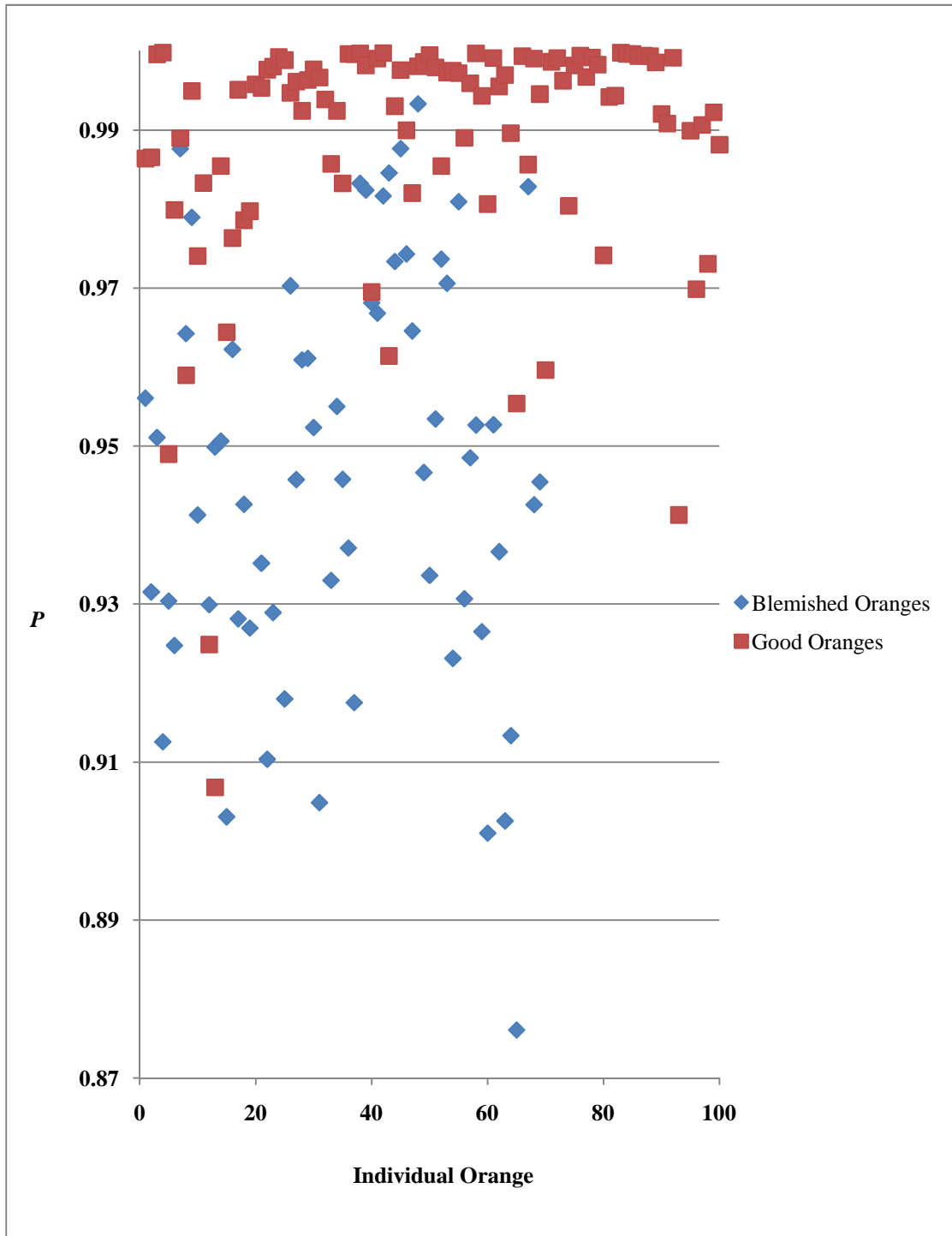


Fig. 90. Classification results after smoothing the rough edges.

Figure 91 shows the classification results after removing the stem.

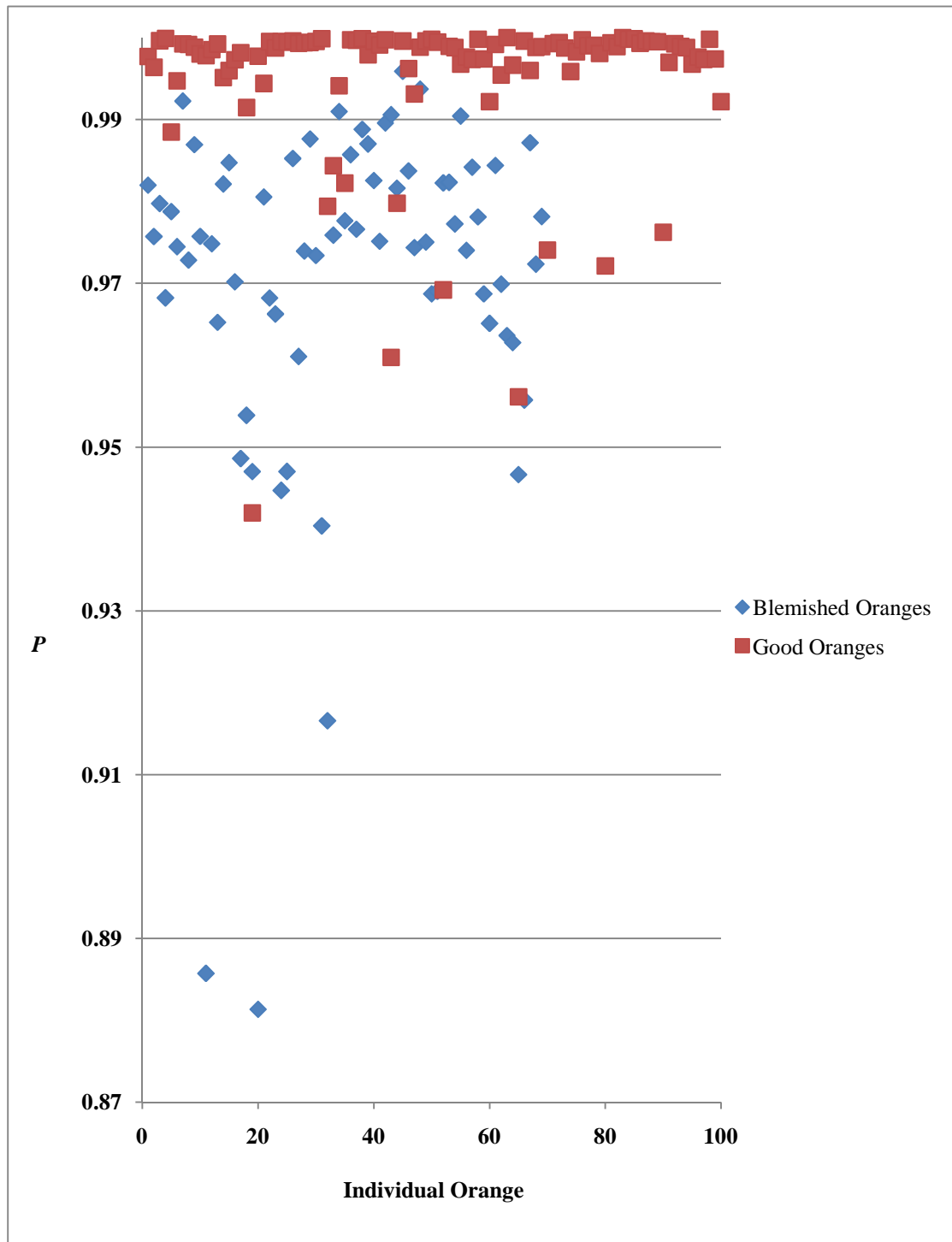


Fig. 91. Classification results after removing the stem.

The layers are generated dynamically at run time, which differ in the size for the same orange due to the lighting changes. Blemishes should be quantified based on stable and predictable measurements. p is redefined for each orange fruit as follows:

$$p = 1 - \frac{bnp}{tp} \quad (48)$$

where tp is the total number of pixels on the whole orange. Figure 92 shows the classification results.

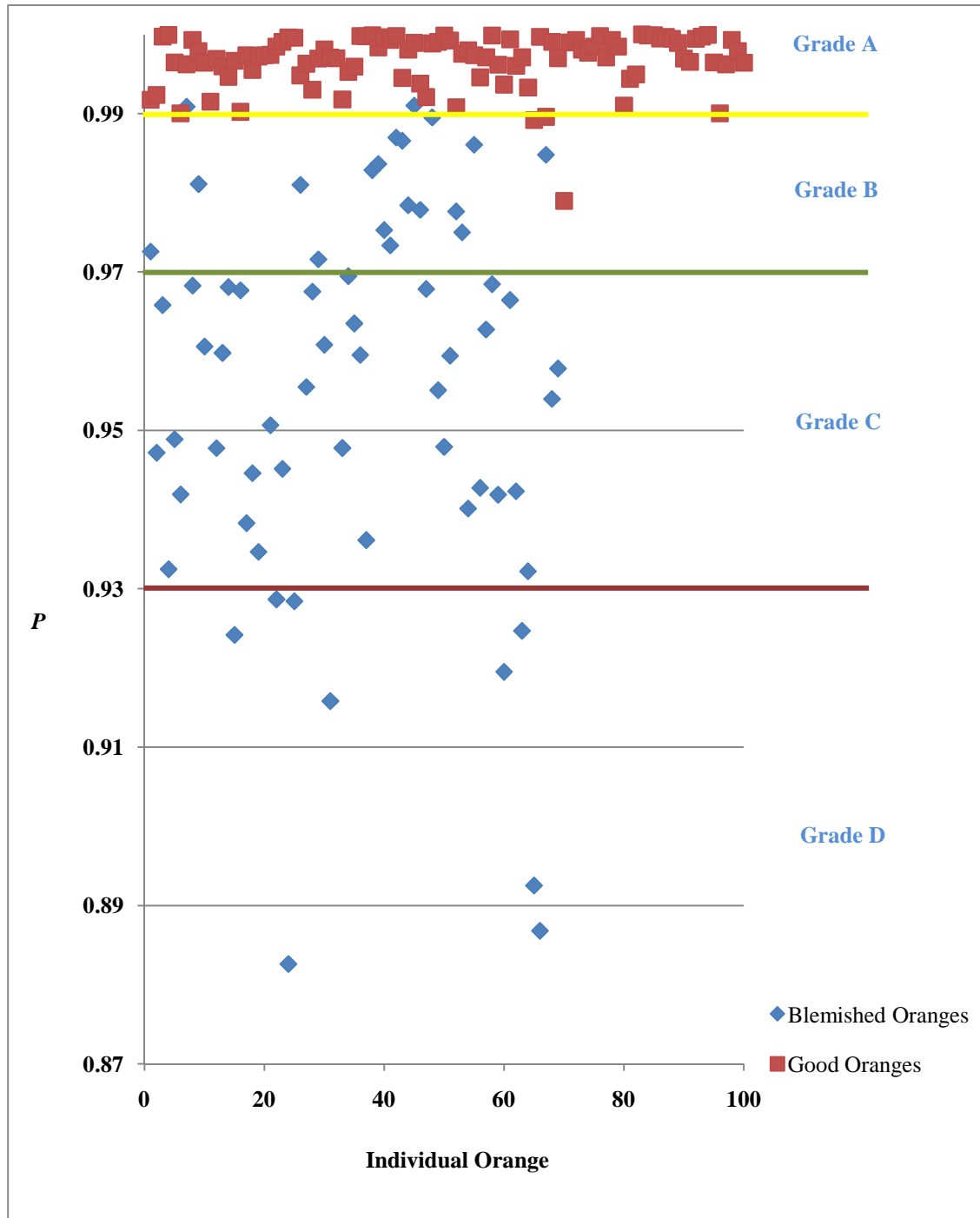


Fig. 92. Classification results using the new formula.

It is evident that the classified good oranges mostly clustered within the Grade A quality band, according to the proposed algorithms. This amounts to 96% correct classification. On the other hand, the classified blemished oranges scattered across the graph in different quality bands. This proves that the algorithm is able to work out varying degrees of quality for blemished oranges. Altogether, the algorithm garnered 97% correct classification results for the blemished fruits.

Also, as depicted in Figure 92, p is predefined to be 0.99 for the highest quality grade. Therefore, if p is greater than or equal to 0.99, then the orange is classified to be of high quality. Table 29 summarizes the classification results produced by the novel algorithm.

Fruit Image Types	No. of Images	No. of Correctly Classified Images	No. of Wrongly Classified Images	% of Correct Classification
Good Orange	100	96	4	96%
Blemished Orange	70	68	2	97%

Table 29. Summary of classification results using the novel algorithm.

Moreover, the grading criteria can be easily be adjusted according to the requirement set by the market, by defining the different quality bands. For instance:

1. Grade A: p above 0.99
2. Grade B: p between 0.97 and 0.99
3. Grade C: p between 0.93 and 0.95
4. Grade D: p below 0.93

Chapter 6

Conclusion and Future Work

An adaptive intelligent fruit grading system for ripe and unripe oranges is proposed in this research. The contributions of this work are summarized as follows:

1. It was observed that the global intensity variation between pixels from the same orange is not sufficient to classify defects (described in Section 5.4.2.4 and 5.4.2.5). Most of the existing algorithms are not addressing this significant issue and one solution to the problem is presented here.
2. Most of the fruit grading algorithms are largely based on the use of neural networks to achieve a more thorough analysis of the fruit's surface (described in Section 3). As part of the requirements that makes it difficult to correctly grade fruits are thousands of training examples that have to be gathered before the experiment and analysis. This process is considered to be very time-consuming and computationally expensive. On the other hand, the novel algorithm presented here does not rely on training. It can be implemented, tested and modified without any rigorous and lengthy training requirements.
3. The novel algorithm may be suitable for grading some other fruits that have the rounded convex surface property (described in Section 5.4.1.5). However, the orange colour class proposed here will have to be modified slightly to be adapted for grading other fruits.
4. Some existing algorithms are able to grade fruits into different quality bands (i.e. histogram-based analysis, etc.). However, such algorithms cannot locate explicitly where the blemishes are on the fruit. The novel algorithm is able to locate the blemish and measure its area with high level of accuracy (described in Section 5.4.8).

Oranges are assessed according to their surface texture, such as bruising, discolouration and other blemishes. All these defect types contribute roughly equally to the final grading decision. The grade is a measure of the size of these surface defects over the whole orange. An extension of the proposed research would be to incorporate some measure of depth for the blemishes, based on their relative intensities. In effect, it could be deduced that the darker the blemishes are, the deeper the damage is.

Appendix A. Otsu's Method

Algorithm Details with Sample Computations

Subset of a grey scale image is extracted to be the sample data for demonstration purpose. The sample data is a two dimensional array of picture elements which is used to make up an image. It has a width(W=6) and height(H=6). The total number of pixels in this sample data(N=36) is WxH(6x6). Table 30 shows the extracted sample data.

Sample Data						
	1	2	3	4	5	6
1	5	0	0	0	20	40
2	120	0	0	10	70	4
3	20	90	70	120	220	7
4	10	0	100	50	144	1
5	30	60	0	1	0	1
6	20	10	20	1	10	30

Table 30. Sample data selected for Otsu's method implementation.

The algorithm of Otsu's thresholding method is implemented in this section.

1. Initialize grey scale ranging from 0 to 255(L-1). L is chosen to be 256.

$$L = 256$$

2. Initialize class probability(ω_1) of class one to be 0.

$$\omega_1 = 0$$

3. Initialize the maximum or minimum between-class variance to be 0;

$$Min\sigma_w^2 = 0, \quad \text{minimizing the between-class variance}$$

or

$$Max\sigma_b^2 = 0, \quad \text{maximizing the between-class variance}$$

4. Find the maximum and minimum grey level separately.

$$MinGreyLevel = 0$$

$$MaxGreyLevel = 220$$

5. Compute image histogram which is the distribution of values for the pixels in

the sample data. The histogram is shown in Figure 93.

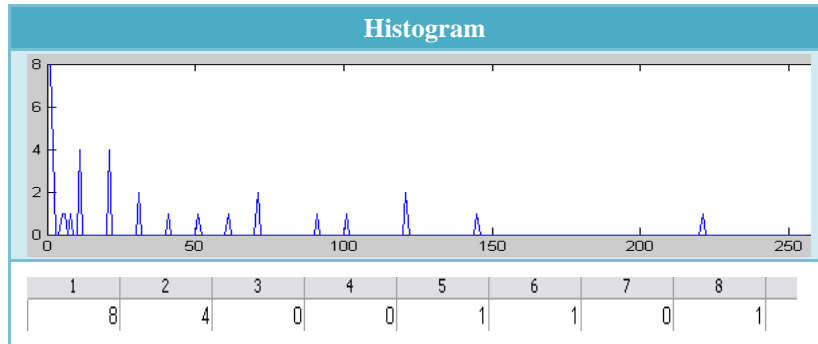


Fig. 93. Histogram analysis for Otsu's method implementation.

Index starting value of an excel spreadsheet is 1. So the first column indexed 1 is corresponding to grey level 0. It can be interpreted as there are 8 pixels at grey level 0 in the sample data.

6. Compute grey levels probabilities P_i for each grey level. The computed probabilities are shown in Figure 94.

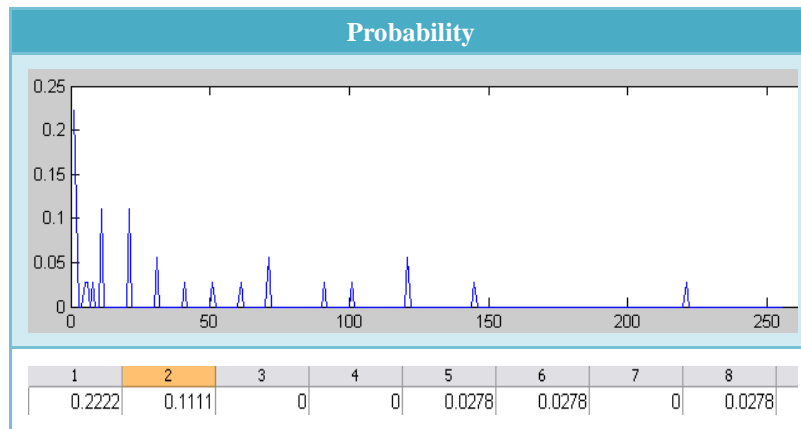


Fig. 94. Probability analysis for Otsu's method implementation

$$P_0 = n_0 / N, \quad 0.2222 = 8 / 36, \quad \text{at grey level } 0$$

$$P_1 = n_1 / N, \quad 0.1111 = 4 / 36, \quad \text{at grey level } 1$$

$$P_2 = n_2 / N, \quad 0 = 0 / 36, \quad \text{at grey level } 2$$

..... ,,

$$P_{255} = n_{255} / N, \quad 0 = 0 / 36, \quad \text{at grey level } 255$$

7. Compute grey levels mean M .

$$M = \sum(P_i * i) = (0.2222 * 0) + (0.1111 * 1) + (0 * 2) + \dots + (0 * 255)$$

$$= 35.6667, \quad 0 \leq i \leq L$$

8. Step through all possible thresholds. The algorithm used is to maximize the between-class variance.

$$t = [\text{MinGreyLevel} \dots \text{MaxGreyLevel}] = [0 \dots 220]$$

➤ **At $t = 0$, $0 \leq i \leq t$**

$$\omega_1 = \sum P_i = P_0 = 0.2222$$

$$\omega_2 = 1 - \omega_1 = 1 - 0.2222 = 0.7778$$

$$\text{temp} = \sum(i * P_i) = 0 * 0.2222 = 0$$

$$\mu_1 = \sum(i * P_i) / \omega_1 = \text{temp} / \omega_1 = 0 / 0.2222 = 0$$

$$\mu_2 = (M - \sum(i * P_i)) / \omega_2 = (M - \text{temp}) / \omega_2$$

$$= (35.6667 - 0) / 0.7778 = 45.8559$$

$$\sigma_b^2 = \omega_1 * \omega_2 * (\mu_1 - \mu_2)^2 = 0.2222 * 0.7778 * (0 - 45.8559)^2 = 363.4147$$

➤ **At $t = 1$, $0 \leq i \leq t$**

$$\omega_1 = \sum P_i = P_0 + P_1 = 0.2222 + 0.1111 = 0.3333$$

$$\omega_2 = 1 - \omega_1 = 1 - 0.3333 = 0.6667$$

$$\text{temp} = \sum(i * P_i) = 0 * 0.2222 + 1 * 0.1111 = 0.1111$$

$$\mu_1 = \sum(i * P_i) / \omega_1 = \text{temp} / \omega_1 = 0.1111 / 0.3333 = 0.3333$$

$$\mu_2 = (M - \sum(i * P_i)) / \omega_2 = (M - \text{temp}) / \omega_2$$

$$= (35.6667 - 0.1111) / 0.6668 = 45.8559 = 53.3307$$

$$\sigma_b^2 = \omega_1 * \omega_2 * (\mu_1 - \mu_2)^2$$

$$= 0.3333 * 0.6667 * (0.3333 - 53.3307)^2 = 624.1298$$

➤ **At $t = 2$, $0 \leq i \leq t$**

$$\omega_1 = \sum P_i = P_0 + P_1 + P_2 = 0.2222 + 0.1111 + 0 = 0.3333$$

$$\omega_2 = 1 - \omega_1 = 1 - 0.3333 = 0.6667$$

$$\text{temp} = \sum(i * P_i) = 0 * 0.2222 + 1 * 0.1111 + 2 * 0 = 0.1111$$

$$\mu_1 = \sum(i * P_i) / \omega_1 = temp / \omega_1 = 0.1111 / 0.3333 = 0.3333$$

$$\begin{aligned} \mu_2 &= (M - \sum(i * P_i)) / \omega_2 = (M - temp) / \omega_2 \\ &= (35.6667 - 0.1111) / 0.6668 = 45.8559 = 53.3307 \end{aligned}$$

$$\begin{aligned} \sigma_b^2 &= \omega_1 * \omega_2 * (\mu_1 - \mu_2)^2 \\ &= 0.3333 * 0.6667 * (0.3333 - 53.3307)^2 = 624.1298 \end{aligned}$$

.....

9. This step is equivalent to step 8. The algorithm used is to minimize the between-class variance. Step through all possible thresholds.

$$t = [MinGreyLevel...MaxGreyLevel] = [0...220]$$

➤ **At $t = 0$, $0 \leq i \leq t$**

$$\omega_1 = \sum P_i = P_0 = 0.2222$$

$$\omega_2 = 1 - \omega_1 = 1 - 0.2222 = 0.7778$$

$$temp = \sum(i * P_i) = 0 * 0.2222 = 0$$

$$\mu_1 = \sum(i * P_i) / \omega_1 = temp / \omega_1 = 0 / 0.2222 = 0$$

$$\begin{aligned} \mu_2 &= (M - \sum(i * P_i)) / \omega_2 = (M - temp) / \omega_2 \\ &= (35.6667 - 0) / 0.7778 = 45.8559 \end{aligned}$$

$$temp1 = \sum(i - \mu_1)^2 * P_i = (0 - 0)^2 * 0.2222 = 0$$

$$\begin{aligned} temp2 &= \sum(i - \mu_2)^2 * P_i \\ &= (1 - 45.8559)^2 * 0.1111 \quad + \\ &\quad (2 - 45.8559)^2 * 0 \quad + \\ &\quad \dots \quad \dots \quad \dots \quad + \\ &\quad (220 - 45.8559)^2 * 0.0278 \end{aligned}$$

$$= 2181.9 \quad t+1 \leq i \leq MaxGreyLevel$$

$$\sigma_1^2 = \sum(i - \mu_1)^2 * P_i / \omega_1 = temp1 / \omega_1 = 0 / 0.2222 = 0$$

$$\sigma_2^2 = \sum(i - \mu_2)^2 * P_i / \omega_2 = temp2 / \omega_2 = 2181.9 / 0.7778 = 2805.2$$

$$\sigma_w^2 = \omega_1 * \sigma_1^2 + \omega_2 * \sigma_2^2 = 0.2222 * 0 + 0.7778 * 2805.2 = 2181.9$$

➤ **At t = 1, 0 ≤ i ≤ t**

$$\omega_1 = \sum P_i = P_0 + P_1 = 0.2222 + 0.1111 = 0.3333$$

$$\omega_2 = 1 - \omega_1 = 1 - 0.3333 = 0.6667$$

$$temp = \sum(i * P_i) = 0 * 0.2222 + 1 * 0.1111 = 0.1111$$

$$\mu_1 = \sum(i * P_i) / \omega_1 = temp / \omega_1 = 0.1111 / 0.3333 = 0.3333$$

$$\begin{aligned} \mu_2 &= (M - \sum(i * P_i)) / \omega_2 = (M - temp) / \omega_2 \\ &= (35.6667 - 0.1111) / 0.6668 = 45.8559 = 53.3307 \end{aligned}$$

$$\begin{aligned} temp1 &= \sum(i - \mu_1)^2 * P_i = (0 - 0.3333)^2 * 0.2222 + (1 - 0.3333)^2 * 0.1111 \\ &= 0.0741 \end{aligned}$$

$$\begin{aligned} temp2 &= \sum(i - \mu_2)^2 * P_i \\ &= (2 - 53.3307)^2 * 0 \quad + \\ &\quad (3 - 53.3307)^2 * 0 \quad + \\ &\quad \dots \quad \dots \quad \dots \quad + \\ &\quad (220 - 53.3307)^2 * 0.0278 \\ &= 1921.1 \quad t+1 \leq i \leq MaxGreyLevel \end{aligned}$$

$$\sigma_1^2 = \sum(i - \mu_1)^2 * P_i / \omega_1 = temp1 / \omega_1 = 0.0741 / 0.3333 = 0.2223$$

$$\sigma_2^2 = \sum(i - \mu_2)^2 * P_i / \omega_2 = temp2 / \omega_2 = 1921.1 / 0.6667 = 2881.5059$$

$$\begin{aligned} \sigma_w^2 &= \omega_1 * \sigma_1^2 + \omega_2 * \sigma_2^2 = 0.3333 * 0.2223 + 0.6667 * 2881.5059 \\ &= 1921.2 \end{aligned}$$

➤ **At t = 2, 0 ≤ i ≤ t**

$$\omega_1 = \sum P_i = P_0 + P_1 + P_2 = 0.2222 + 0.1111 + 0 = 0.3333$$

$$\omega_2 = 1 - \omega_1 = 1 - 0.3333 = 0.6667$$

$$temp = \sum(i * P_i) = 0 * 0.2222 + 1 * 0.1111 + 2 * 0 = 0.1111$$

$$\mu_1 = \sum(i * P_i) / \omega_1 = temp / \omega_1 = 0.1111 / 0.3333 = 0.3333$$

$$\begin{aligned} \mu_2 &= (M - \sum(i * P_i)) / \omega_2 = (M - temp) / \omega_2 \\ &= (35.6667 - 0.1111) / 0.6668 = 45.8559 = 53.3307 \end{aligned}$$

$$\begin{aligned} temp1 &= \sum(i - \mu_1)^2 * P_i = (0 - 0.3333)^2 * 0.2222 + \\ &\quad (1 - 0.3333)^2 * 0.1111 + \\ &\quad (2 - 0.3333)^2 * 0 \\ &= 0.0741 \end{aligned}$$

$$\begin{aligned} temp2 &= \sum(i - \mu_2)^2 * P_i \\ &= (3 - 53.3307)^2 * 0 \quad + \\ &\quad (4 - 53.3307)^2 * 0.0278 \quad + \\ &\quad \dots \quad \dots \quad \dots \quad + \\ &\quad (220 - 53.3307)^2 * 0.0278 \\ &= 1921.1 \quad t+1 \leq i \leq MaxGreyLevel \end{aligned}$$

$$\sigma_1^2 = \sum(i - \mu_1)^2 * P_i / \omega_1 = temp1 / \omega_1 = 0.0741 / 0.3333 = 0.2223$$

$$\sigma_2^2 = \sum(i - \mu_2)^2 * P_i / \omega_2 = temp2 / \omega_2 = 1921.1 / 0.6667 = 2881.5059$$

$$\begin{aligned} \sigma_w^2 &= \omega_1 * \sigma_1^2 + \omega_2 * \sigma_2^2 \\ &= 0.3333 * 0.2223 + 0.6667 * 2881.5059 \\ &= 1921.2 \end{aligned}$$

.....,

10. Desired threshold corresponds to the maximum $Max\sigma_b^2$ or minimum $Min\sigma_w^2$ which is at grey level 60.

Appendix B. Histogram Analysis

Algorithm Details with Sample Computations

The following example shows how the algorithm works on the subset of an isolated red component (Table 31). The task is to compute a new value for the pixel $P(4,4)$. Currently the pixel $P(4,4)$ has a value 10.

	1	2	3	4	5	6	7
1	10	20	20	50	80	30	50
2	30	30	60	40	20	80	20
3	40	60	20	60	50	100	60
4	20	90	30	10	10	40	30
5	40	30	20	70	60	0	50
6	40	10	10	50	40	20	10
7	70	60	50	30	80	10	20

Table 31. Sample data selected for image preprocessing.

1. On the vertical direction, both pixels $P(1,4)$ and $P(7,4)$ are three pixels away from the pixel $P(4,4)$.

1. Compute the arithmetic average between $P(1,4)$ and $P(7,4)$.

$$average = (P(1,4) + P(7,4)) / 2 = (50 + 30) / 2 = 40$$

2. Compute the variance between $P(4,4)$ and the average.

$$variance = P(4,4) - average = 10 - 40 = -30$$

3. The first candidate value is derived by the variance plus 128. Pixel values can't be negative, so the constant value 128 is added on top of the variance.

$$valueOne = 128 + variance = 128 - 30 = 98$$

2. On the horizontal direction, both pixels $P(4,1)$ and $P(4,7)$ are three pixels away from the pixel $P(4,4)$.

1. Compute the arithmetic average between $P(4,1)$ and $P(4,7)$.

$$average = (P(4,1) + P(4,7)) / 2 = (20 + 30) / 2 = 25$$

2. Compute the variance between $P(4,4)$ and the average.

$$variance = P(4,4) - average = 10 - 25 = -15$$

3. The second candidate value is derived by the variance plus 128.

$$valueTwo = 128 + variance = 128 - 15 = 113$$

3. On the left diagonal direction, both pixels $P(1,1)$ and $P(7,7)$ are three pixels away from the pixel $P(4,4)$.

1. Compute the arithmetic average between $P(1,1)$ and $P(7,7)$.

$$average = (P(1,1) + P(7,7)) / 2 = (10 + 20) / 2 = 15$$

2. Compute the variance between $P(4,4)$ and the average.

$$variance = P(4,4) - average = 10 - 15 = -5$$

3. The third candidate value is derived by the variance plus 128.

$$valueThree = 128 + variance = 128 - 5 = 123$$

4. On the right diagonal direction, both pixels $P(1,7)$ and $P(7,1)$ are three pixels away from the pixel $P(4,4)$.

1. Compute the arithmetic average between $P(1,7)$ and $P(7,1)$.

$$average = (P(1,7) + P(7,1)) / 2 = (50 + 70) / 2 = 60$$

2. Compute the variance between $P(4,4)$ and the average.

$$variance = P(4,4) - average = 10 - 60 = -50$$

3. The fourth candidate value is derived by the variance plus 128.

$$valueFour = 128 + variance = 128 - 50 = 78$$

The new pixel value for $P(4,4)$ is the maximum value among these four candidate values.

$$P(4,4) = \text{Max}[98 \ 113 \ 123 \ 78] = 123$$

Appendix C. Local Defect Search

Algorithm Details with Sample Computations

There are two types of block-wise features, one is mean block variance, and the other one is squared block variance. The following example demonstrates how the algorithm works on a single pixel $P(4,4)$ based on Table 31. $P(4,4)$ has eight neighborhood-pixels, such as $P(3,3)$, $P(3,4)$, $P(3,5)$, $P(4,5)$, $P(5,5)$, $P(5,4)$, $P(5,3)$ and $P(4,3)$.

1. Compute the variance between $P(4,4)$ and eight neighborhood-pixels separately.

$$V1 = (P(3,3) - P(4,4))^2 = (20 - 10)^2 = 100$$

$$V2 = (P(3,4) - P(4,4))^2 = (60 - 10)^2 = 2500$$

$$V3 = (P(3,5) - P(4,4))^2 = (50 - 10)^2 = 1600$$

$$V4 = (P(4,5) - P(4,4))^2 = (10 - 10)^2 = 0$$

$$V5 = (P(5,5) - P(4,4))^2 = (60 - 10)^2 = 2500$$

$$V6 = (P(5,4) - P(4,4))^2 = (70 - 10)^2 = 3600$$

$$V7 = (P(5,3) - P(4,4))^2 = (20 - 10)^2 = 100$$

$$V8 = (P(4,3) - P(4,4))^2 = (30 - 10)^2 = 400$$

2. Compute the arithmetic mean of the eight variances in step one.

$$\begin{aligned} M(4,4) &= (V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8) / 8 \\ &= (100 + 2500 + 1600 + 0 + 2500 + 3600 + 100 + 400) / 8 \\ &= 10800 / 8 = 1350 \end{aligned}$$

3. For each pixel within the region, repeat the computations in step one and two.
4. Add up all the values computed in step three.
5. Divide the computed value in step four by the total number of pixels within the region. The new value computed in this step is called the mean block variance. The mean block variance is given a value of 1400 for demonstration purpose.

$$MBV = 1400$$

6. Compute the squared difference between $M(4,4)$ and MBV .

$$V = (M(4,4) - MBV)^2 = (1350 - 1400)^2 = 2500$$

7. For each pixel within the region, repeat the computations in step six.
8. Add up all the values computed in step seven.
9. Divide the computed value in step eight by the mean block variance. The new value computed in this step is called the squared block variance.

Appendix D. Pixel Reclassification

Algorithm Details with Sample Computations

The following computations demonstrate how the pixel reclassification works. The colour channel values of $P(4,2)$ are 50, 20 and 7 on three channels separately. Table 32 is an instance of the new class mean matrix for demonstration purpose.

New Class Mean Matrix			
Class No.	Red Channel	Green Channel	Blue Channel
1	23	13	6
2	52	31	18
3	145	73	12
4	228.82	113.95	22.55

Table 32. New class mean matrix for demonstration purpose.

$$ssd(1) = \sqrt{((23-50).^2 + (13-20).^2 + (6-7).^2) / 3} = 16.11$$

$$ssd(2) = \sqrt{((52-50).^2 + (31-20).^2 + (18-7).^2) / 3} = 9.05$$

$$ssd(3) = \sqrt{((145-50).^2 + (73-20).^2 + (12-7).^2) / 3} = 62.87$$

$$ssd(4) = \sqrt{((228.82-50).^2 + (113.95-20).^2 + (22.55-7).^2) / 3} = 116.96$$

$P(4,2)$ is classified as a member of the class two. The pixel value is replaced by the corresponding class mean 52 on the red channel and stored in the pixel reclassification matrix which is a two-dimensional array with the same size as the RGB image.

References

1. Berry, D. (1987). Colour recognition using spectral signatures. *Pattern Recognition Letters*, 6(1), 69-75. Retrieved from Potral.
2. Besson, O., Bidon, S. & Tourneret, J. Y. (2008). Covariance matrix estimation with heterogeneous samples. *IEEE Transactions on Signal Processing*, 56(3), 909-920. doi: 10.1109/TSP.2007.908995
3. Brosnan, T. & Sun, D. (2004). Improving quality inspection of food products by computer vision – a review. *Journal of Food Engineering*, 61(1), 3-16. doi: 10.1016/S0260-8774(03)00183-3
4. Bumbaca, F. & Smith, K. C. (1987). Design and implementation of a colour vision model for computer vision applications. *Computer Vision, Graphics, and Image Processing*, 39(2), 226-245. Retrieved from Potral.
5. Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6), 679-698. Retrieved from Potral.
6. Cao, L., Shi, Z. K. & Cheng, K. (2002). A fast method of automatic multilevel thresholding. *International Conference on Computer Graphics and Spatial Information System*.
7. Chang, W., Chen, S., Lin, S., Huang, P. & Chen, Y. (1994). Vision based fruit sorting system using measures of fuzziness and degree of matching. *IEEE International Conference on Systems, Man, and Cybernetics*, 3, 2600-2604. doi: 10.1109/ICSMC.1994.400263
8. Chen, Y., Chao, K. & Kim, M. (2002). Machine Vision technology for agricultural applications. *Computers and electronics in Agriculture*, 36(2-3), 173-191. doi: 10.1016/S0168-1699(02)00100-x

9. Du, C. & Sun, D. (2004). Recent developments in the applications of image processing techniques for food quality evaluation. *Trends in Food Science Technology*, 15 (5), 230-249. doi: 10.1016/j.tifs.2003.10.006
10. Deng, Y., Manjunath, B. S., Kenney, C., Moore, M. S. & Shin, H. (2001). An efficient color representation for image retrieval. *IEEE Transactions on Image Processing*, 10(1), 140-147. doi: 10.1109/83.892450
11. Egmont-Petersen, M., Ridder, D. & Handels, H. (2002). Image processing with neural networks. *Pattern Recognition*, 35(10), 2279-2301. doi: 10.1016/S0031-3203(01)00178-9
12. Farid, H. (2001). Blind inverse gamma correction. *IEEE Transactions on Image Processing*, 10(10), 1428-1433. doi: 10.1109/83.951529
13. Guo, F., & Cao, Q. (2004). Study on color image processing based intelligent fruit sorting system. *Fifth World Congress on Intelligent Control and Automation*, 6, 4802-4805. doi: 10.1109/WCICA.2004.1343622
14. Onyuksel, I. & Hosseini, S. H. (2002). Amdahl's law: a generalization under processor failures. *IEEE Transactions on Reliability*, 44(3), 455-462. doi: 10.1109/24.406581
15. Lekic, N., Mijanovic, Z. & Gobovic, D. (2002). The simple multiprocessor communication system. *2002 International Conference on Electronics, Circuits and System*, 3, 1039-1042. doi: 10.1109/ICECS.2002.1046428
16. Bharati, M., Liu, J. & John, F. (2004). Image texture analysis: methods and comparisons. *Chemometrics and Intelligent Laboratory Systems*, 72(1), 57-71. doi: 10.1016/j.chemolab.2004.02.005
17. Messom, C. (2008). *Parallel distributed System*[Handout]. Auckland, New Zealand: University of Massey: Bachelor in Computer Science.
18. Palatin, H., Buzek, P. & Beran, M. (2007). *Parallel Image Processing(PIP)*. Retrieved March 12, 2007, from <http://www.ms.mff.cuni.cz/~beran/pip/pip.html.en>

19. Preparata, F. P. & Hong, S. J. (1977). Convex hull of finite sets points in two and three dimensions. *Communications of the ACM*, 20(2), 87-93. Retrieved from Potral.
20. Johnson, M. (2008). *Machine vision*[Handout]. Auckland, New Zealand: University of Massey: Bachelor in Computer Science.
21. Recce, M., Taylor, J., Piebe, A., & Tropicano, G. (1996). High speed vision-based quality grading of oranges. *Proceedings of the 1996 International Workshop on Neural Networks for Identification, Control, Robotics, and Signal/Image Processing*, 136-144. doi: 10.1109/NICRSP.1996.542754
22. Rogowitz, B. Z. (2001). Human vision and the expanding field of electronic imaging. *2001 International Conference on Image Processing*, 2, 1-2. Retrieved from IEEE Xplore.
23. Thomas, H. (2009). *Projections of world production and consumption of citrus to 2010*. Retrieved June 1, 2009, from <http://www.fao.org/DOCREP/003/X6732E/x6732e02.htm#2>
24. Unay, D. (2005). *Multispectral image processing and pattern recognition techniques for quality inspection of apple fruits*. PhD's Thesis, Faculté Polytechnique de Mons, Belgium.
25. Vijayarekha, K., & Govindaraj, R. (2006). Citrus fruit external defect classification using wavelet packet transform textures and ANN. *IEEE International Conference on Industrial Technology*, 2872-2877. doi: 10.1109/ICIT.2006.372646
26. Werblin, F., Jacobs, A. & Teeters, J. (1996). The computational eye. *IEEE Spectrum*, 33(5), 30-37. doi: 10.1109/6.490054
27. Wikipedia. (2008). *RGB colour model*. Retrieved December 20, 2008, from <http://en.wikipedia.org/wiki/RGB>

28. Zhang, J. & Hu, J. (2008). Image segmentation based on 2D Otsu method with histogram analysis. *2008 International Conference on Computer Science and Software Engineering*, 6, 105-108. doi: 10.1109/CSSE.2008.206
29. Zhou, Y., Li, S. & Jin, R. (2002). A new fuzzy neural network with fast learning algorithm and guaranteed stability for manufacturing process control. *Fuzzy Sets and Systems*, 132(2), 201-216. doi: 10.1016/S0165-0114(01)00234-2