CERTIFICATION OF APPROVAL

RIPENESS CLASSIFICATION OF OIL PALM FRUIT TO ENSURE OPTIMUM QUANTITY OF OIL USING IMAGE PROCESSING TECHNIQUES

by

Munirah binti Abdul Hamid

A project dissertation submitted to the
Electrical and Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(ELECTRICAL AND ELECTRONICS ENGINEERING)

Approved by,

____

(PN. ZAZILAH MAY)
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
December 2010

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

____

MUNIRAH BINTI ABDUL HAMID

ABSTRACT

Palm oil is plant oil derived from the pulp of the fruit of oil palm (*Elaeis guineensis).* Malaysia is the second largest producer of palm oil which produced 17.7 million ton of palm oil [6]. They are mostly made into margarine, cooking oil, oleochemicals and specialty fats. Today, with the rapid movement of technology, many methods and techniques are used to ensure optimum quantity and quality of oil based on the ripeness of oil palm fruit. Ripeness classification of fruit is based on color, texture, firmness, size, shape and bruises. The project involves on detecting optimum quantity and quality of oil based on the ripeness of oil palm fruit using suitable Digital Image Processing Techniques. The objective of this project is to develop an easy and flexible system where user can use the system to classify the maturity of fruit using CCD camera and Matlab-Image Processing Toolbox. Image processing technique is an important tool to classify the ripeness of oil palm fruit especially in recognizing the color of the fruit. This paper focused on RGB and HSV techniques. RGB technique can be described by indicating the value of red, green and blue color while HSV is through its brightness level. In this project, the color distribution of the fruit is calculated to make ripeness decision. The scope of study covers the characteristics of the features and the suitable image processing techniques used to detect these features. For each features, a sequence of image processing methods will be applied using Matlab-Image Processing Toolbox to detect its presence and the filtering will be used in imaging to enhance the image quality and the pictures will be taken by CCD camera. Thus, the project achieved most of its objectives and proved its ability to be used as real application in order to get the optimum quantity and quality of oil.

ACKNOWLEDGEMENT

In the name of Allah, the most compassionate, the merciful. First and foremost, I would like to thank Allah the Almighty for His blessings and guidance in allowing me to complete this project within the allocated time.

I would like to convey my gratitude to my family, for their love, morale support and understanding while I spent most of the time with this project throughout these two semesters.

A deep depth of gratitude to my supervisor, Miss Zazilah May who had been supportive by giving useful guidance throughout the completion of this project. Her enthusiasm to guide and supervise me has made this project worthwhile.

Finally, my deepest appreciation to Ms. Sarah, Mr. Mohd Hafizi Kamarudin and Mr. Salman Farisi Sidik for helping out on how to use the Matlab Image Processing Toolbox software. I hope that the next batch of students involved in continuing this project would take the opportunity to learn as much as they can.

Thank you. May Allah s.w.t. bless always be with us.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
CCD	Charge-Coupled Device
FFA	Free Fatty Acid
HSI	Hue, Saturation, Intensity
HSV	Hue, Saturation, Value
NTSC	National Television System Committee
OER	Oil Extraction Rate
PDF	Probability Density Function
RGB	Red, Green, Blue

CHAPTER 1

INTRODUCTION

1.1 Background of Study

Digital Image Processing is a process where the data can be obtained by looking at the image of the fruit with the aid of software that converts the image into digital image and CCD camera to take the picture. Some research has been done in finding the best techniques to classify the oil quantity based on the ripeness of oil palm fruit especially color analysis. Color analysis technique is an optimum indicator of the ripeness of oil palm fruit as this fruit has different color for different ripeness which these projects focus on RGB and HSV techniques.

1.2 Problem Statement

Oil palm fruit is an important fruit as they are mostly made into margarine, cooking oil, oleochemicals and specialty fats. In agriculture applications, especially for fruits, the grading methods used in the palm oil is manually using human grader. This method is subjective means not all the fruits collected is ripe which will bring waste in industry. As a result, poor grading system will affect the quantity and quality of the oil. Thus, to solve this problem, the grading system should be invented to classify the optimum quantity of oil based on the ripeness of the fruit. The ripeness classification consists of three different categories which are unripe, ripe and overripe. The maturity or ripening index is based on different color index. Color analysis technique is an important tool to classify the ripeness of the fruit as the color surface of oil palm fruit act as a major factor in determining the ripeness of this fruit.

1.3 Objectives

The objectives of this project are:

i. To enhance the oil palm fruit grading system by developing the best techniques to ensure optimum quantity of oil based on the ripeness of oil palm fruit using suitable Digital Image Processing Technique.

ii. To provide an easy and flexible system where user can use the system to classify the maturity of oil palm fruit.

iii. To develop the accurate color analysis for the ripe fruit with optimum quantity of oil by building a coding system using MATLAB-Image Processing Toolbox.

1.4 Scope of Study

This project is aimed to enhance the grading system of oil palm fruit by determining the color techniques from image analysis which is best correlate to the oil quantity of oil palm fruit. During the process, the initial job is to do research and looking into facts of oil palm fruit, color analysis techniques, image processing techniques and CCD camera.

Secondly, gain knowledge about the hardware and software that will be used namely CCD camera and MATLAB-Image Processing Toolbox. For CCD camera, lighting and angle effects must be studied for getting the best image quality. Software, MATLAB-Image Processing Toolbox must be practiced in determining the accurate color analysis for the ripe fruit contained optimum quantity of oil.

Finally, picture of the fruit is taken using CCD camera. At final stage, the ripeness testing, image sharpening and image filtering using MATLAB-Image Processing Toolbox will be done constantly. The results will be compared between the unripe, ripe and overripe of oil palm fruit.

CHAPTER 2

LITERATURE REVIEW

2.1 Digital Image Processing

2.1.1 Definitions

An image is a representation of a two-dimensional function, *f(x,y),* where *x* and y are spatial coordinates and the amplitude of *f* at any pair of coordinates (x,y) is called the intensity or grey level of the image [3]. A digital Image is defined as a two-dimensional image when (x,y) and the amplitude values of *f* are all finite and discrete quantities [3]. Hence, digital image is composed of a fixed number of elements such as picture elements, image elements, pels and pixels. Digital image also called bitmap images. Digital image processing refers to processing digital images with the aid of computer algorithms. Furthermore, digital image processing allocate a wider range of algorithms to be applied to the input data and can avoid problems such as noise and signal distortion during processing.

2.1.2 Digital Image Processing Steps

The digital image processing steps consists of many stages such as image acquisition, image enhancement, restoration, colour image processing, wavelets, compression, morphological processing, segmentation, representation and description and object recognition. The flow of process is shown in next page:

Image Acquisition

This is the first step in digital image processing which the image captured is already in digital form. Basically, image acquisition involves pre-processing such as scaling and analyzing. These steps is aim to extract the color and intensity from the images.

Image Enhancement

This is the process of image manipulation in order to produce good quality of image. This process is to bring out detail or certain features of an image. There are two categories in image enhancement which are intensity and spatial filtering. Intensity refers to image adjustment or contrast stretching. While spatial filtering is refer to noise reduction and noise generating. The filtering process consists of lowpass, highpass and high-frequency emphasis filtering.

Image Restoration

Image restoration is the process to improve the appearance of an image based on mathematical or probabilistic of image degradation. It is to obtain an estimation to be as close as possible to the original input image. This process is to reconstruct or recover the image that has been degrading means to remove image blur by applying debluring function.

Colour Image Processing

This step needs the capability to write code using image processing toolbox. Colour image processing involves of three categories which are colour transformations, spatial processing and colour vector processing. Colour transformation deals with processing the pixels of each colour based on their values or intensity transformation and not on spatial coordinate. Besides, spatial processing involves of spatial filtering of each colour planes. The colour vector processing is based on processing of all components of a colour image.

Wavelets

This is the basic step for indicating images in different degrees of resolution which images are divided in sequence into smaller regions. This process is used in tasks ranging from edge detection to image smoothing.

Image Compression

Image compression is to reduce the storage required for an image. This process is to remove the redundant data by transforming a 2-D pixel array into uncorrelated data set. This process is aim to remove the redundancies which are coding redundancy, interpixel redundancy and psychovisual redundancy. Coding redundancy is the code words are used less than optimal. Besides, interpixel redundancy means the correlation between the pixels of an image. The psychovisual is due to data that is ignored by human visual system. There are two compression standards which are JPEG and JPEG 2000 [3].

Morphological Image Processing

Morphological image processing is to extract image components that are useful in the representation of region shape such as boundaries, skeletons and convex hull. This process also used for pre-processing or post-processing likes morphological filtering, thinning and pruning.

Image Segmentation

Image segmentation is purpose to subdivide image into its regions. There are two properties of image intensity value which are discontinuity and similarity. Discontinuity is to partition an image based on abrupt changes in intensity such as edges in an image. Points, lines and edges are used in detecting intensity discontinuity. Besides, similarity is to partition an image into region that are similar according to a set of predefined criteria.

Representation and Description

Representation means to represent the region of external characteristic which is boundary, described by features such as length. Besides, it also to represent the region of internal characteristic which is pixels comprising the region. Internal representation is selected when the main focus is on regional properties such as colour and texture. For description, descriptors should be not sensitive to variations in region size, rotation and translation.

Object Recognition

This process is divided into two principal consists of decision-theoretic and structural. Decision-theoretic deals with patterns described using quantitative descriptors such as texture, length and area. Besides, structural is represented by symbolic information likes strings, properties and relationship between symbols.

2.2 Color Analysis

2.2.1 Color Classification

The maturity or classification of oil palm fruit ripening condition is based on three different categories which are unripe, ripe and overripe. Firstly, the color of unripe bunch is purplish red colored fruits which covering more than 90% of the bunch surface. Secondly, for the ripe bunch, the color is reddish orange. Lastly, for overripe bunch, the color of fruits is darkish red which covers almost more than 80% of the fruits. [8]

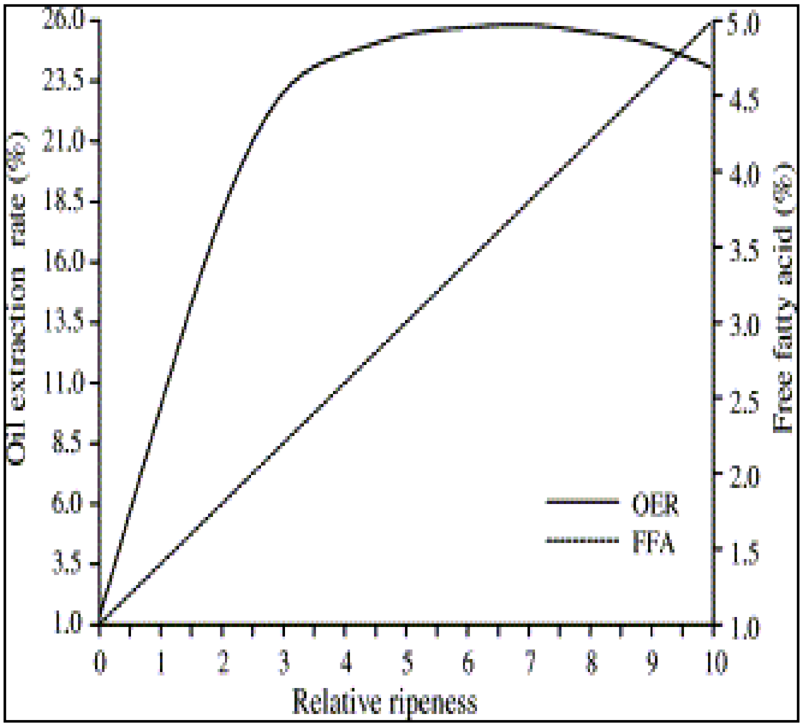


Figure 1: The relationship between degree of ripeness with the oil extraction rate (OER) and free fatty acid (FFA) content of oil palms [8].

From the Figure 1, the oil palm content at early stage of ripening is increase and the rate of increase slowing down before leveling off after the fruits have become optimally ripe. For unripe bunch, the oil content is lower. In fact, every 1% of unripe bunches present in the lots, the oil extraction rate (OER) will weakening decrease by 0.132%. Besides, for ripe bunch, the theoretical oil extraction rate (OER) is between 21% and 23% and the free fatty acid (FFA) content should not be higher than 5% [8].

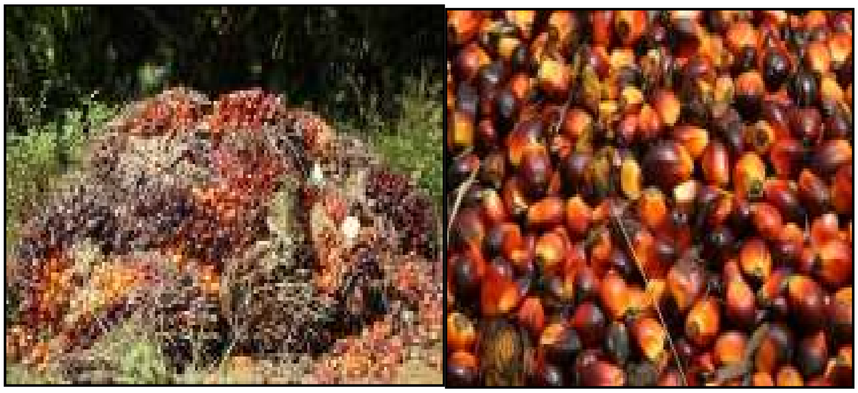


Figure 2: Oil Palm Fruits

The color of each fruit on the bunch varies with location and fruits on the bunch do not ripe at the same time. In fact, when more than 85% of fruits on any bunch show a similar degree of maturity and the remaining 15% which are hiddenly located in the core regions of the bunch, it can be concluded that once a fruit within a bunch is ripe, all other fruits on the bunch are ripe as well. In this work, a fruit located at the middle section was selected for color grading [8].

2.2.2 RGB (Red, Green, Blue) Technique

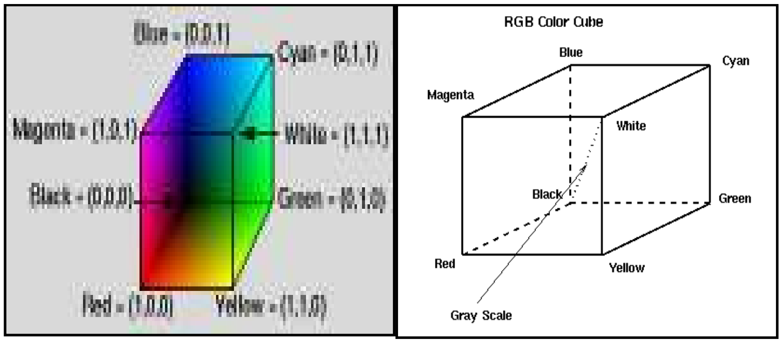


Figure 3: RGB Color Cube

An RGB color images are formed from the three color components of a color pixels corresponding to Red, Green and Blue. An RGB color space basically is shown graphically as RGB cube as in Figure 3. RGB index images has two components which are data matrix of integers, *x* and colourmap matrix, *map.* The length of the map is equal to the number of colors it defines. Furthermore, each row of map contains RGB components. The color of each pixel is obtained using the corresponding value of integer matrix, *x* as an indicator into a map. But, if the three columns of *map* are equal, a grayscale *map* will be produced [3].

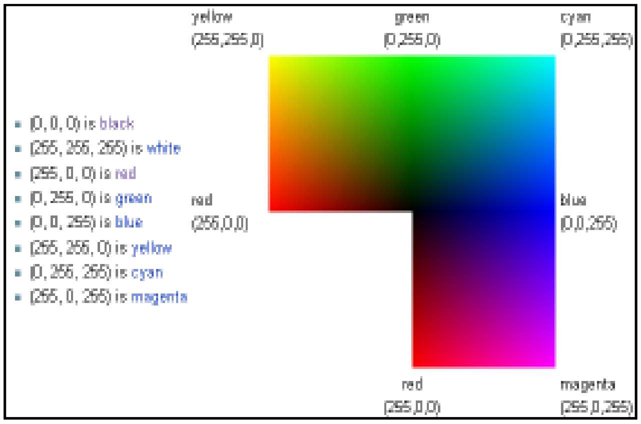


Figure 4: RGB Color Range

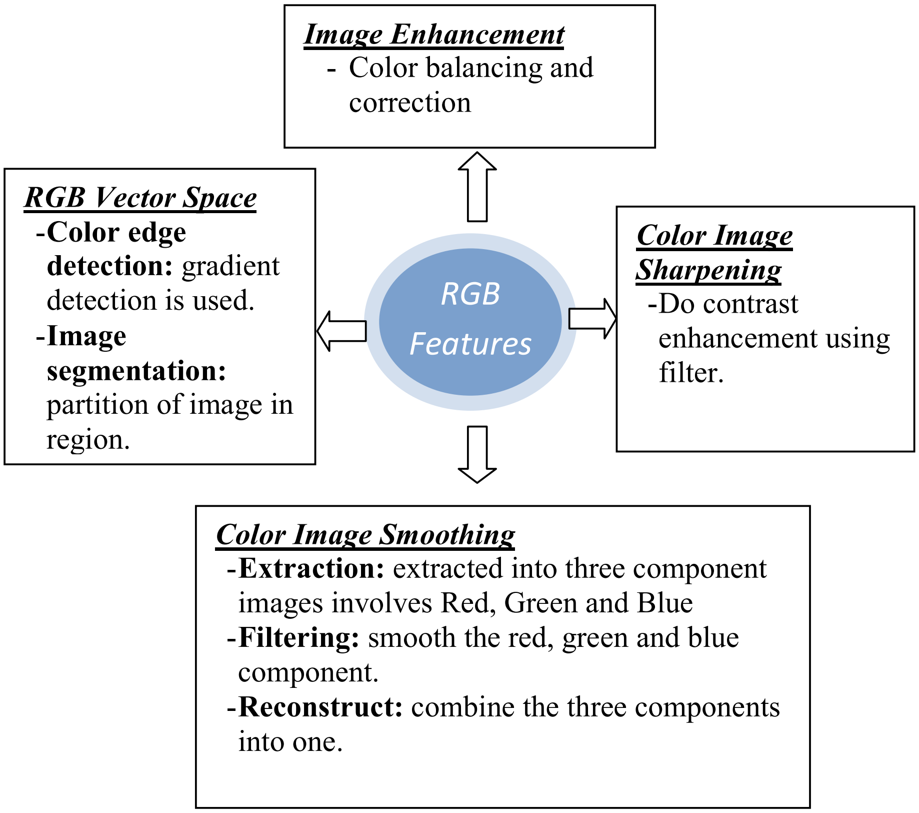


Image Enhancement

- Color balancing and correction

RGB Vector Space

-**Color edge detection:** gradient detection is used.

-**Image segmentation:** partition of image in region.

Color Image Sharpening

-Do contrast enhancement using filter.

Color Image Smoothing

-**Extraction:** extracted into three component images involves Red, Green and Blue

-**Filtering:** smooth the red, green and blue component.

-**Reconstruct:** combine the three components into one.

Figure 5: RGB Features

2.2.3 HSV (Hue, Saturation, Value) Technique

HSV technique used to select colors of paints or inks from color wheel or palette. HSV is more referring to tint, shade and tone. HSV color space is obtained by looking at the RGB color cube along the gray axis that the axis joining the black and white vertices which results in the hexagonally shape color palette.

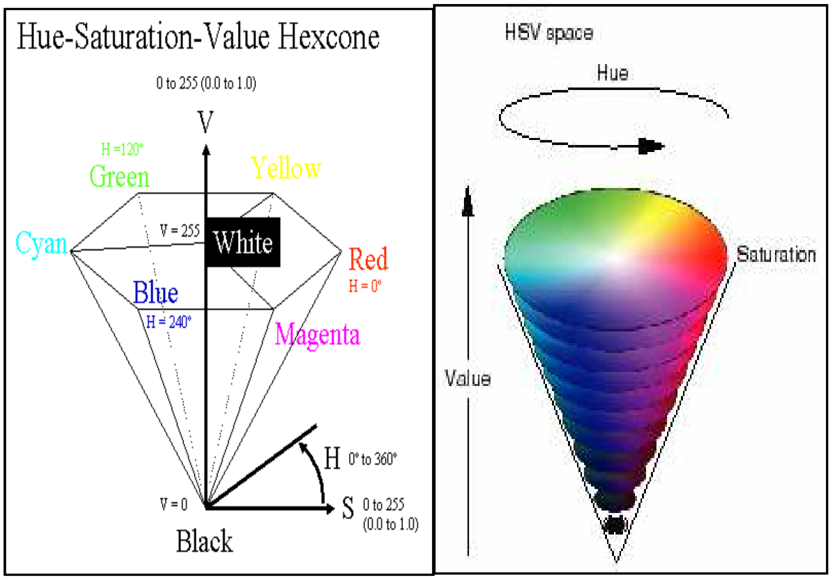


Figure 6: HSV Color Space

As we move along the gray vertical axis, the size of the hexagonal plane perpendicular to the axis will change. Hue is the angle around a color hexagon. Value is measured along the axis of the cone, while saturation is measured from the distance from the V-axis in Figure 6.

Table 1: HSV Color Range

ANGLE	COLOR
0-60	Red
60-120	Yellow
120-180	Green
180-240	Cyan
240-300	Blue
300-360	Magenta

2.2.4 HSI (Hue, Saturation, Intensity) Technique

In this technique, Hue is best to describe a pure color (pure red, pure orange, etc.). Saturation is the measurement of the degree to which pure color is attenuated by white light. Other than that, intensity or grey level is best described for color sensation. The Intensity is along the line joining between black (0, 0, 0) and white (1, 1, 1) vertex from RGB color cube [3]. Intensity value is the intersection value with the intensity axis. Saturation of color increased as a distance far from intensity axis means the saturation point on intensity axis is zero. Furthermore, Hue value unchanged because the colors inside a color triangle contains of different combination of the three vertex colors. The black and white components do not contribute to change in hue but the intensity and saturation do change. We would obtain different hues by rotating the shade plane.

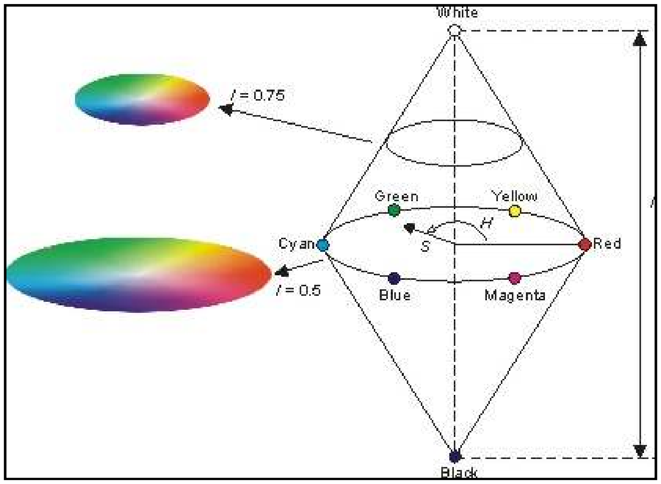


Figure 7: HSI Color Space

CHAPTER 3

METHODOLOGY

3.1 Procedure Identification

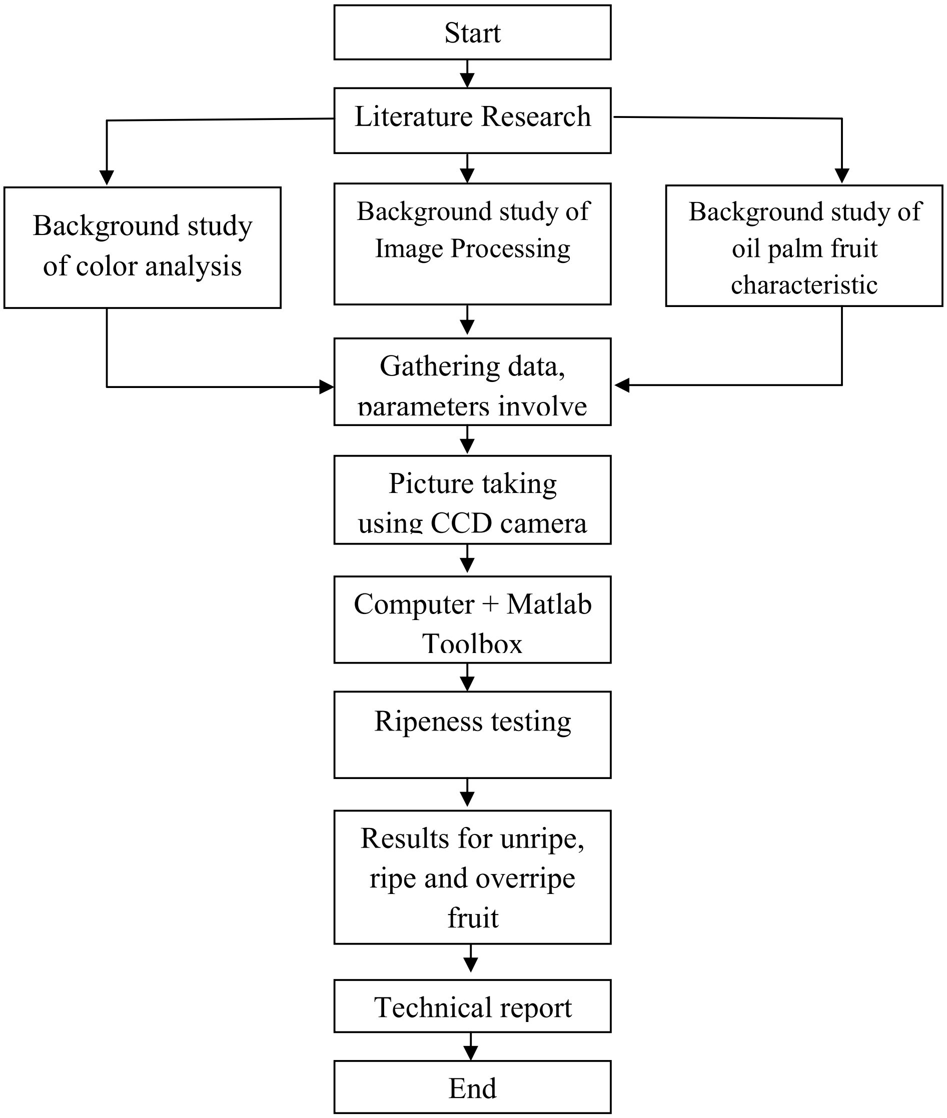

Start
Literature Research
Background study of color analysis
Background study of Image Processing
Background study of oil palm fruit characteristic
Gathering data, parameters involve
Picture taking using CCD camera
Computer + Matlab Toolbox
Ripeness testing
Results for unripe, ripe and overripe fruit
Technical report
End

Figure 8: Flow chart of Ripeness Classification of Oil Palm Fruit

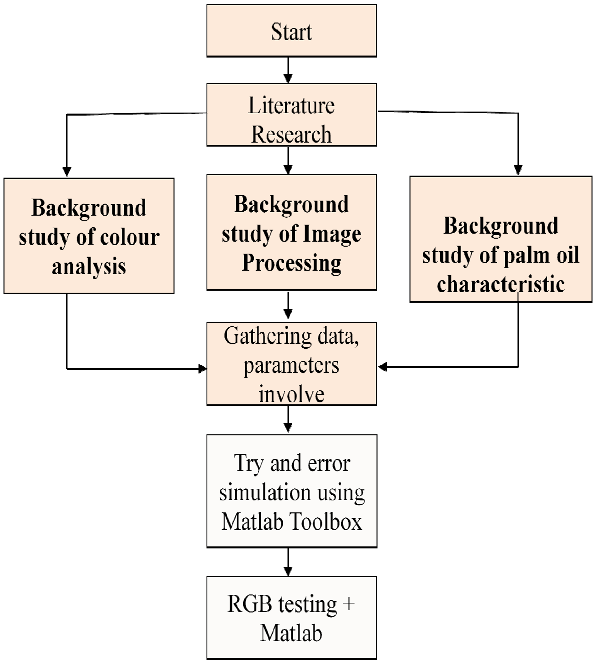

Start
Literature Research
Background study of colour analysis
Background study of Image Processing
Background study of palm oil characteristic
Gathering data, parameters involve
Try and error simulation using Matlab Toolbox
RGB testing + Matlab

Figure 9: Flow chart of Ripeness Classification of Oil Palm Fruit (FYP I)

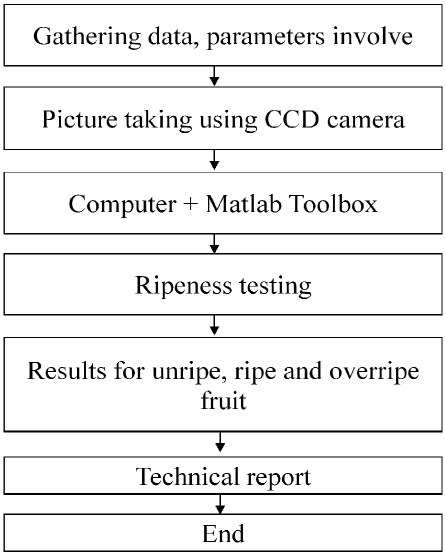

Gathering data, parameters involve
Picture taking using CCD camera
Computer + Matlab Toolbox
Ripeness testing
Results for unripe, ripe and overripe fruit
Technical report
End

Figure 10: Flow chart of Ripeness Classification of Oil Palm Fruit (FYP II)

3.2 Detailed of the Procedure

Throughout this project, there are some procedures to be followed. This is to ensure that the project can be accomplished within the given timeframe.

3.2.1 Data Research and Gathering

Some research has been done covers the study of color classification and analysis. Besides, the facts about oil palm fruit and its application also have been studied.

3.2.2 Picture Taking Using CCD Camera

After finishing with data gathering, next is to take the pictures of oil palm fruit using CCD camera. To take the picture, the lighting and angle factor must be consider. So that, the further study must be carry about CCD camera and digital image for getting a good quality image.

3.2.3 Simulation

Data that was finding before this will be use for simulation. There are two types of the simulation. The first part is manually classifying the ripeness of oil palm fruit by seeing the color of the fruits using eyes. The second part is dealing with MATLAB-Image Processing Toolbox. Further study must be done to get the accurate results for unripe, ripe and overripe fruit.

3.3 Tools and Equipments Used

Table 2: Software and Hardware used

Software	**Hardware**
MATLAB-Image Processing Toolbox	CCD Camera

To develop this project, several tools are needed which are displayed above in Table 2. The project is developed using Matlab Image Processing Toolbox that converts the image into digital image for simulation in classifying the maturity of oil palm fruit, CCD Camera is hardware devices required for taking the picture of the fruits.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Results

Overall, the project achieved most of its objectives and proved its ability to be used as real application. Image processing technique is successfully employed to develop techniques in classifying the ripeness level of oil palm fruit to ensure optimum quantity of oil based on the extracted RGB and HSV color feature of the oil palm fruits. The project is executed by comparing the ripeness level of fruit by experienced human grader with the simulation using matlab image processing toolbox. Based on the simulation process, the percentages of accuracy for unripe, ripe and overripe fruit are 92.5%, 82.5% and 92.5%.

<u>Unripe Fruit:</u>

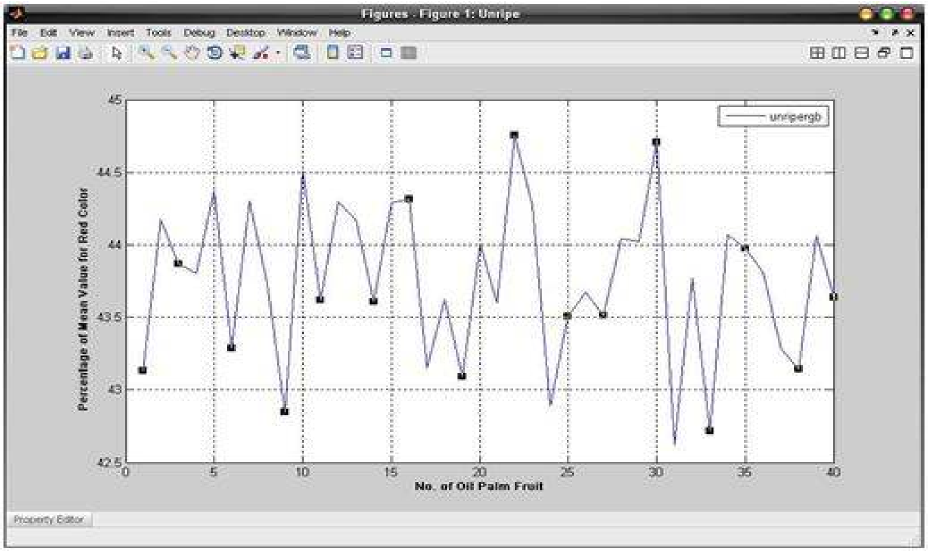


Figure 11: Unripe Oil Palm Fruit

$$Accuracy\ [\frac{unripe\ images}{no.of\ original\ images}] = \frac{37}{40}\ x\ 100\% = 92.5\%$$

<u>Ripe Fruit:</u>

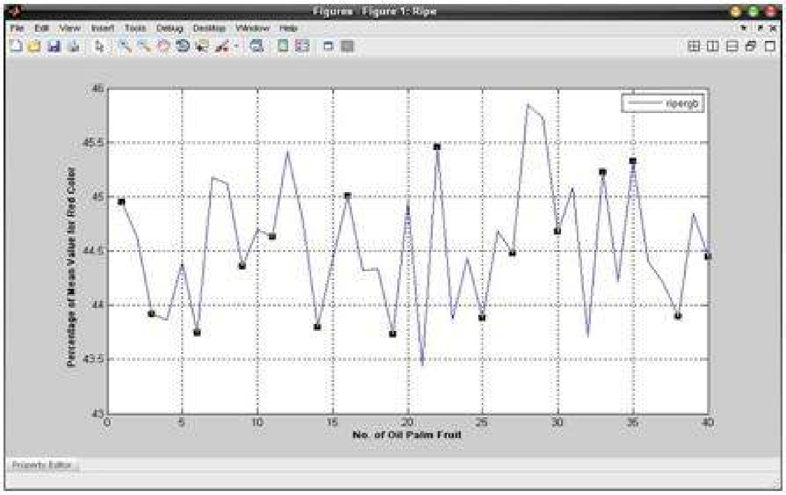


Figure 12: Ripe Oil Palm Fruit

$$Accuracy\ [\frac{ripe\ images}{no.of\ original\ images}] = \frac{33}{40}\ x\ 100\% = 82.5\%$$

<u>Overripe:</u>

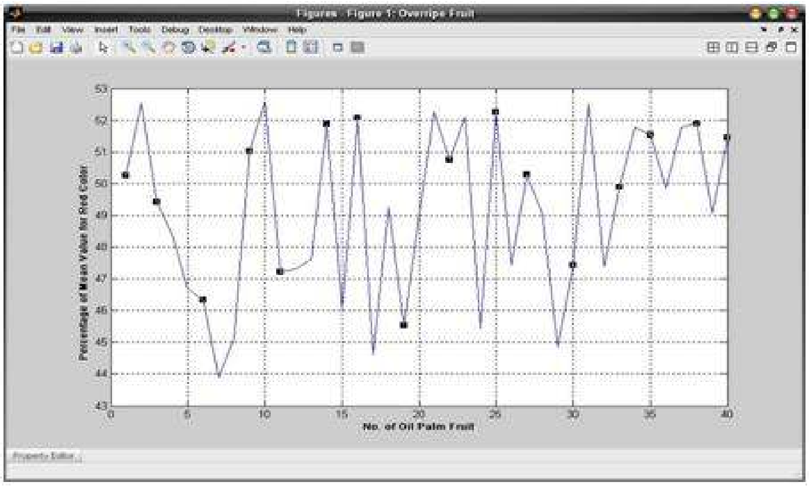


Figure 13: Overripe Oil Palm Fruit

$$Accuracy\ [\frac{overripe\ images}{no.of\ original\ images}] = \frac{37}{40}\ x\ 100\% = 92.5\%$$

4.1.1 Coding Estimation

Throughout this project, certain simulation will be run to obtain the accurate result for color analysis. As a result, noise filtering and good quality image must be taken in order to get accurate results. The purpose is to produce a coding system in order to do the simulation of data which will helps agriculture person in fruit grading system. These project simulations focus on RGB and HSV Techniques.

4.1.2 Try and Error Simulation

<u>Display Image:</u>

'imshow' function is used in calling the image.

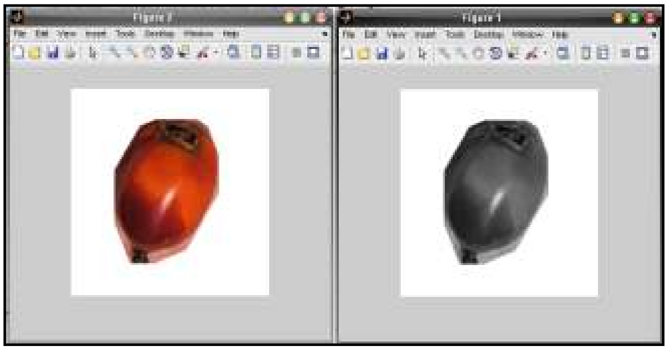


Figure 1
File Edit View Insert Tools Desktop Window Help
Figure 1
File Edit View Insert Tools Desktop Window Help

Figure 14: RGB and Grayscale Image

<u>Creating an Intensity Profile of an Image</u>

The intensity profile of an image is the set of intensity values taken from regularly spaced points along a line segment or multiline path in an image. For points that do not fall on the center of a pixel, the intensity values are interpolated. Based on the result, the red color or red element is the highest than blue and green.

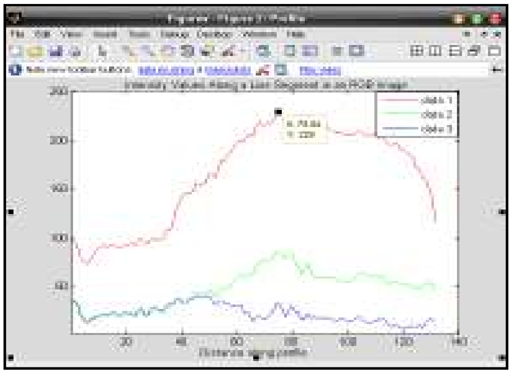


Figure 15: Intensity Profile of an Image

Performing Linear Filtering of Images using 'imfilter'

Filtering of images can be performed using the toolbox function `imfilter`.

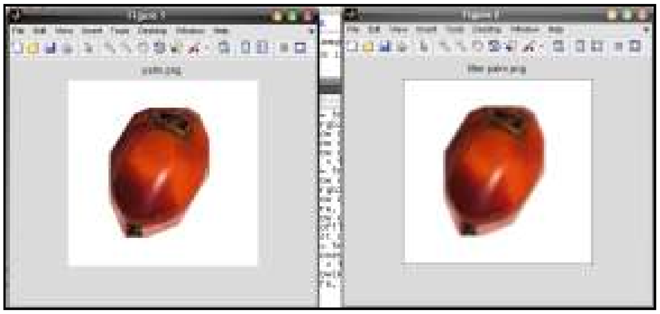


Figure 16: Filtered Images

Sharped an Image

The unsharp masking filter has the effect of making edges and fine detail in the image more crisped.

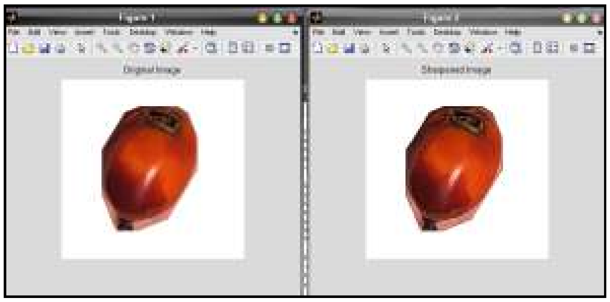


Figure 17: Sharpened Image

Adjusting Intensity Values to a Specified Range

The range of intensity values can be specified in the output image. For example, this code increases the contrast in a low-contrast grayscale image by remapping the data values to fill the entire intensity range [0, 255]. Notice the increased contrast in the image, and that the histogram now fills the entire range.

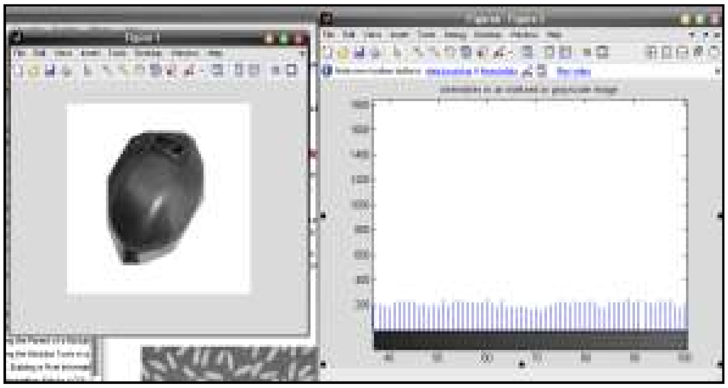
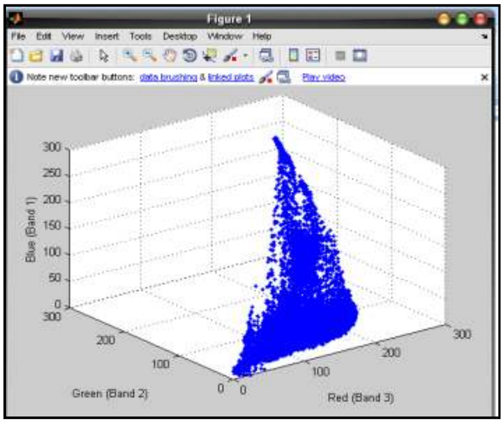


Figure 18: Intensity Values

Enhancing Color Separation Using Decorrelation Stretching

Decorrelation stretching enhances the color separation of an image with significant band-band correlation. The exaggerated colors improve visual interpretation and make feature discrimination easier. The number of color bands, in the image is usually three.



Note new toolbar buttons: data brushing & linked plots Play video

Figure 19: Decorrelation Stretching

4.1.3 Matlab Image Processing Simulation

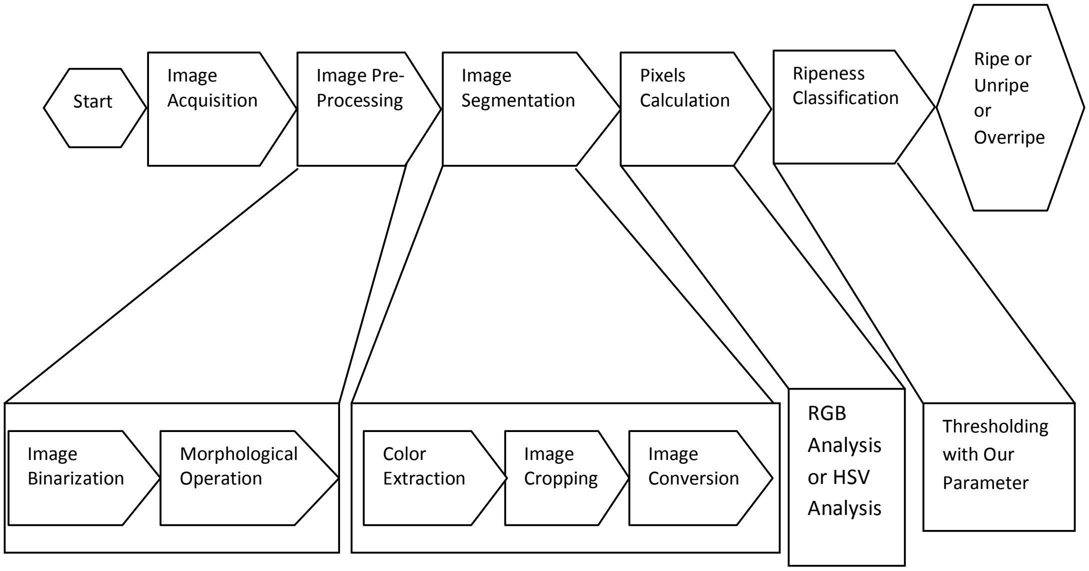


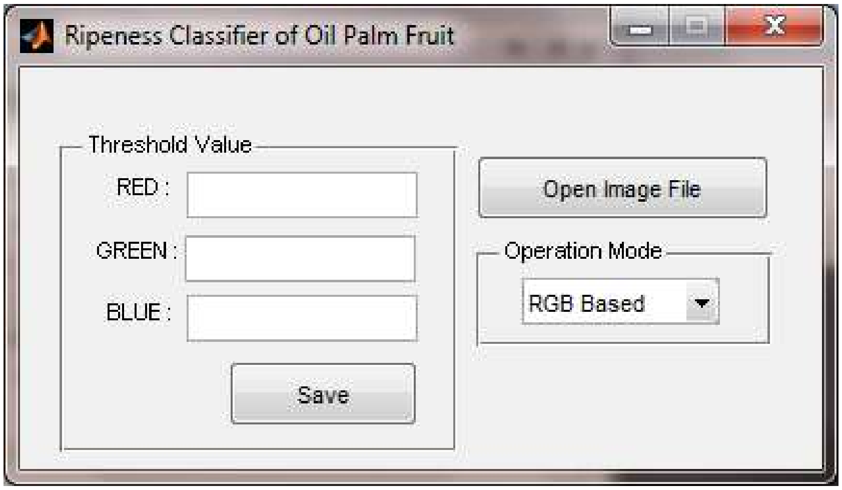
Figure 20: Ripeness Classifier of Oil Palm Fruit Based On Its Color Properties

RGB and HSV techniques is used to get the ripeness or to classify the maturity of the fruit as there are many colors characteristic in oil palm fruit. Firstly, the color distribution of the fruit is calculated to make a ripeness decision which is the percentage of each color component. Based on the ripeness testing simulation using Matlab Image Processing Toolbox, the percentages of Red component exist in the image is assumed as below:

- If Red component exist more than 85%, it is Overripe.
- If Red component exist between 65% to 85%, it is Ripe.
- If Red component exist fewer than 65%, it is Unripe.

Further analysis of other color component (Green, Blue and HSV) can be decided based on input image data and can be known on the way while the program is being made. The input data is images of oil palm fruit with its ripeness (Ripe, Unripe and Overripe). After that, the input data is divided into two groups which are:

- First group: Input data for getting the optimum threshold value.
- Second group: Input data for simulation and testing the program.

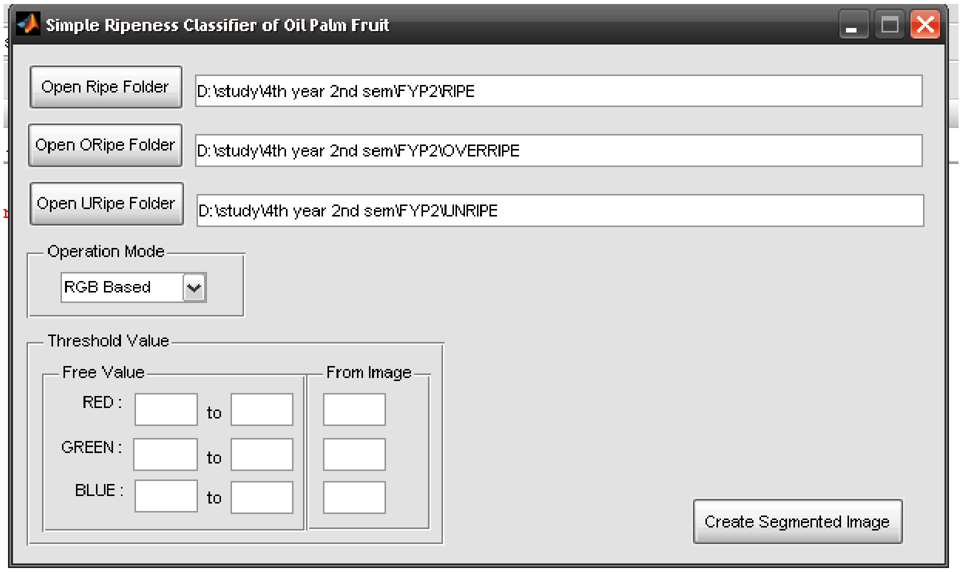


Operation Mode: RGB Based

Figure 21: Threshold Value for RGB

Figure 21 show the classifier for single fruit. First, the optimum threshold value is adjusted by performing image analysis likes color or histogram extraction for several input images with its ripeness (Image in First Group). After getting the optimum threshold value, the program is ready to use and the program can be tested with the second group of input data and see the result.

Image Segmentation is the color based segmentation. This part is an important part of program as the fruit image segmentation must be done before performing classification. This program will create segmented images for all images in selected directory.



Open Ripe Folder: D:\study\4th year 2nd sem\FYP2\RIPE

Open ORipe Folder: D:\study\4th year 2nd sem\FYP2\OVERRIPE

Open URipe Folder: D:\study\4th year 2nd sem\FYP2\UNRIPE

Operation Mode: RGB Based

Figure 22: Ripeness Classifier of Oil Palm Fruit for Three Categories of Ripeness

Create Segmented Image will create or write segmented images to hard disk drive for all images under each category of ripeness. These new images will be saved in the same directory as original images.

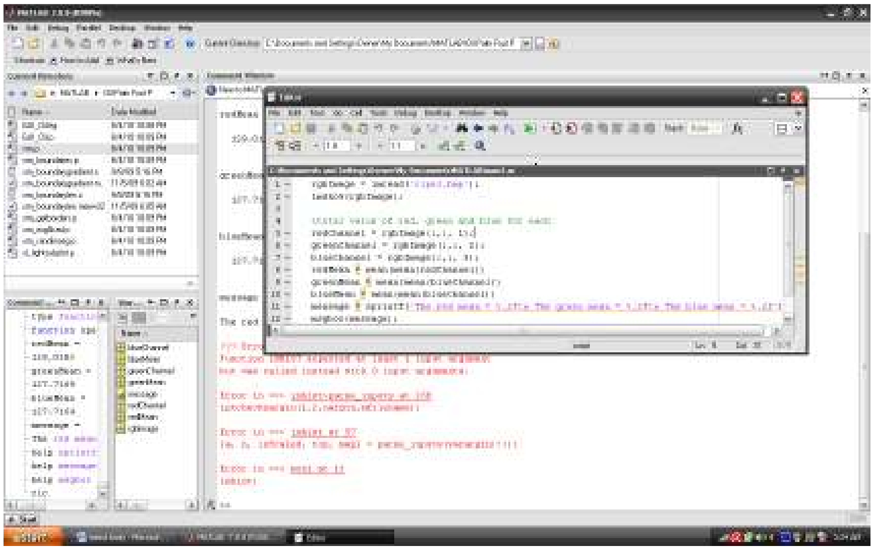


Figure 23: Coding System for Oil Palm fruit

The formulae to calculate the mean value for Red, Green and Blue:

- Red pixel / no. of pixel = Red
- Green pixel / no. of pixel = Green
- Blue pixel / no. of pixel = Blue

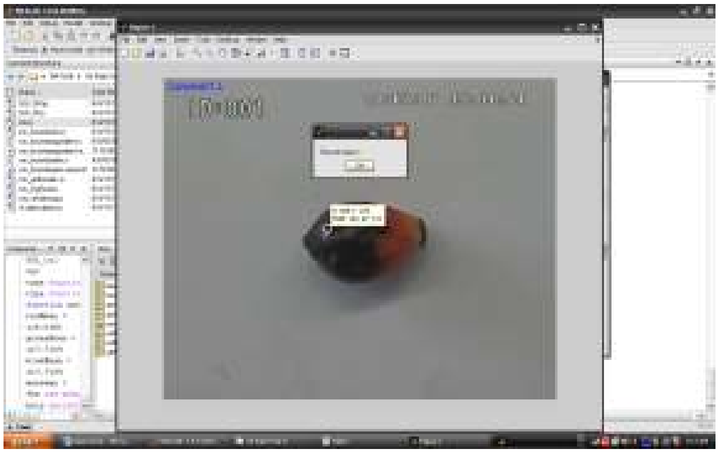


Figure 24: Original Image for Ripe Fruit

4.1.4 Final Matlab Image Processing Simulation

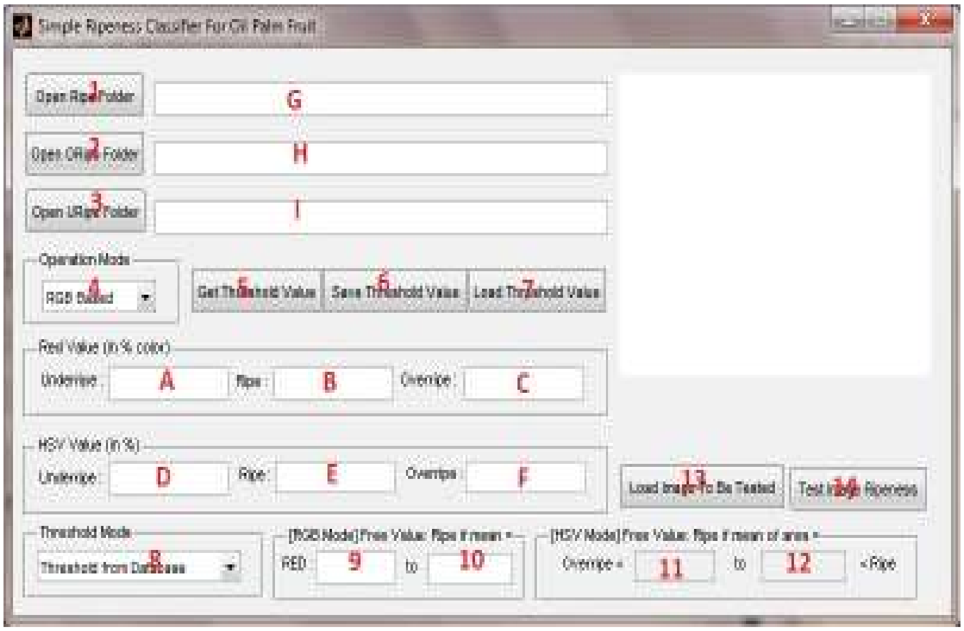


Figure 25: Simple Ripeness Classifier for Oil Palm Fruit

Figure above show the box configuration for ripeness classifier of oil palm fruit. This program can classify the ripeness of the fruit by inserting all the input images of the fruit. In this project, the images were divided into three folders (RIPE, OVERRIPE and UNRIPE).

- "Open Ripe Folder" is for selecting a folder whereby the ripe images is saved.
- "Open ORipe Folder" is for selecting a folder whereby the overripe images is saved.
- "Open URipe Folder" is for selecting a folder whereby the unripe images is saved.

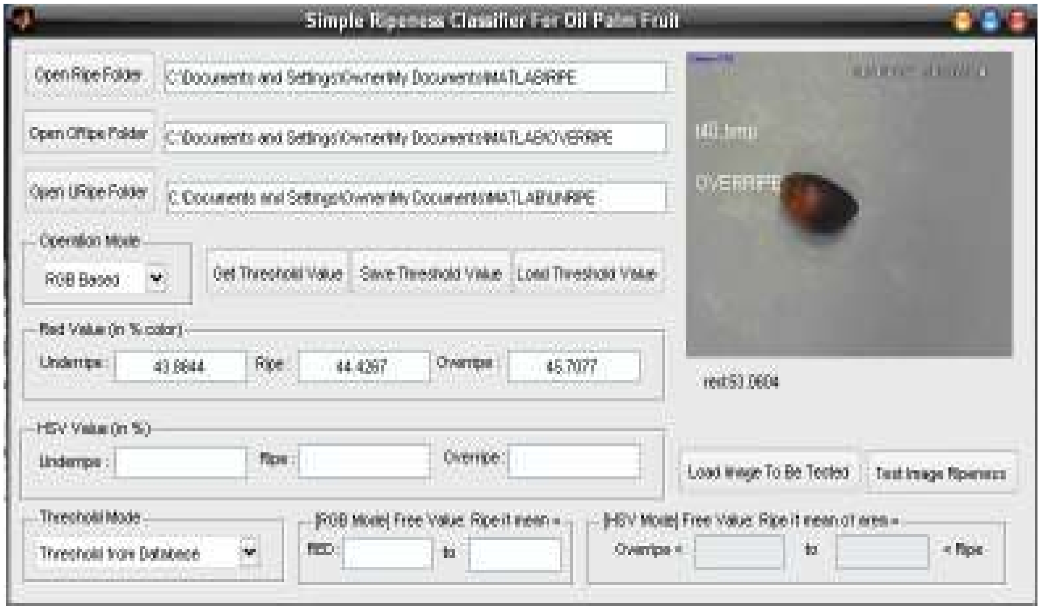
<u>Process of Simple Ripeness Classifier for Oil Palm Fruit</u>

1. A button to open a folder where ripe images are stored (Ripe Folder).
2. A button to open a folder where overripe images are stored (Overripe Folder).
3. A button to open a folder where underripe images are stored (Underripe Folder).
4. A popup menu to select operation mode: RGB or HSV
5. A button to get the threshold value programmatically from ripe, overripe & underripe images stored in Ripe, Overripe and Underripe Folder. This value will be shown in textbox A, B & C (for RGB mode) or textbox D, E, F (for HSV mode) after the operation done.
6. A button to save the threshold value which is getting from "Get Threshold Value". This value will be used as reference for testing the ripeness of an image without re-calculating the threshold.
7. A button to load threshold value which has been saved before.
8. A popup menu to select threshold mode: From Database or User-Defined.
   - Threshold value from database:

- If RGB Mode is selected, program will use value in textbox A, B and C as the threshold to define the ripeness of a test image.
- If HSV Mode is selected, program will use value in textbox D, E and F as the threshold to define the ripeness of a test image.

- Threshold value from User-Defined:
  - If RGB Mode is selected, program will use value in editbox 9 and 10 as the threshold to define the ripeness of a test image. The user first type the value on these editboxes.
  - If HSV Mode is selected, program will use value in editbox 12 and 13 as the threshold to define the ripeness of a test image. The user first type the value on these editboxes.

9. User-Defined Threshold Value for RGB Mode: minimum mean of Red color value.
10. User-Defined Threshold Value for RGB Mode: maximum mean of Red color value.
11. User-Defined Threshold Value for HSV Mode: minimum mean of area for underripe.
12. User-Defined Threshold Value for HSV Mode: maximum mean of area for underripe.
13. A button to select an image to be tested.
14. A button to perform image ripeness operation.


Simple Ripeness Classifier For Oil Palm Fruit

Open Ripe Folder: C:\Documents and Settings\Owner\My Documents\MATLAB\RIPE

Open ORipe Folder: C:\Documents and Settings\Owner\My Documents\MATLAB\OVERRIPE

Open URipe Folder: C:\Documents and Settings\Owner\My Documents\MATLAB\UNRIPE

Operation Mode: RGB Based

Red Value (in % color): Underripe: 43.8644 Ripe: 44.4267 Overripe: 45.7077

HSV Value (in %): Underripe: Ripe: Overripe:

Threshold Mode: Threshold from Database

[RGB Mode] Free Value: Ripe if mean = RED: to

[HSV Mode] Free Value: Ripe if mean of area = Overripe < to < Ripe

40.bmp

OVERRIPE

red:53.0604

Figure 26: Example of Simple Ripeness Classifier for Oil Palm Fruit

Mean of Red Value (RGB Mode) for first-40 images stored in each folder is:

Underripe: 43.8644

Ripe: 44.4267

Overripe: 45.7077

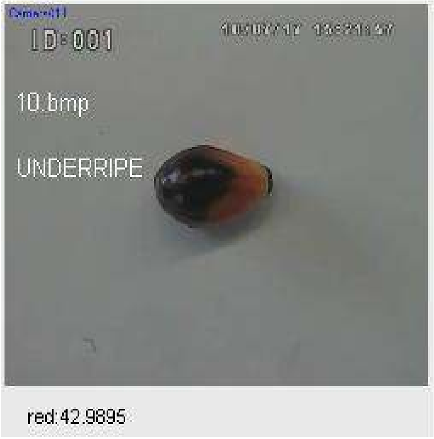
Mean of Area Value (HSV Mode) for first-40 images stored in each folder is:

Overripe: 891.7

Underripe: 2352.125

Ripe: 2903.675

Results of the System

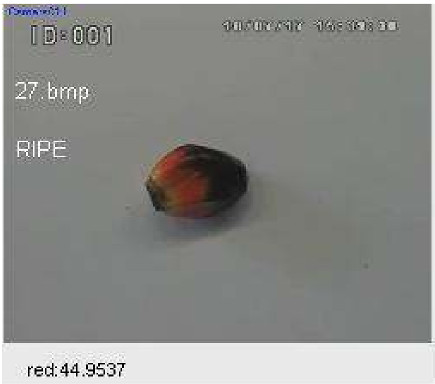


10.bmp

UNDERRIPE

red:42.9895

Figure 27: Example of Underripe Oil Palm Fruit

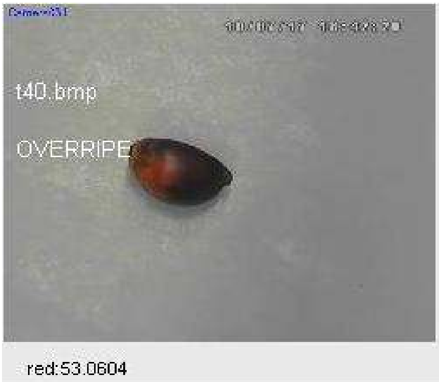


27.bmp

RIPE

red:44.9537

Figure 28: Example of Ripe Oil Palm Fruit


t40.bmp

OVERRIPE

red:53.0604

Figure 29: Example of Overripe Oil Palm Fruit

Philosophy of the Program

RGB Technique:

The program will calculate the mean red color percentage in images. Firstly, program will crop on the center of image where the fruit found. The cropping is done by finding the center of fruit and crops it so the width and height of cropped image is 0.25 of width/height of original image. This will make sure the size of fruit is always maintained are cropped perfectly. This cropping technique can be used because the color of background is same for all images. This algorithm will be performed for all ripe, underripe and overripe images.

Read image:

```
imgInput =
imread(strcat(get(handles.editRipeFolder,'string'),'\',files(i).na
me));
imgInput2 = double(imgInput);
```

Find the fruit:

```
qlevel = 4;
[maps,imgInput2] = srm(imgInput2,qlevel);
mask = srm_boundaries(imgInput2,maps);
BW = uint8(mask);
```

Get the center point of fruit:

```
s = regionprops(BW,'Centroid');
x = round(s.Centroid(1)); y = round(s.Centroid(2));
```

Crop image:

```
imc = imcrop(imgInput,[x-width/4 y-height/4 width/2 height/2]);
```

After cropping, the program will calculate the mean of red value in single image, and store each red value for total mean of all fruit:

```
RedMean = (mean(mean(double(imc(:,:,1))))/255)*100;
RedRipeMean(j) = RedMean;
```

Finally, after find each red –mean value, the program will calculate total of mean value:

```
RipeRedMean= mean(RedRipeMean);
```

HSV Technique:

First, the program will calculate the value of Hue, Saturation and Value. After finding the HSV Value, the program will convert this value to black and white (binary) image. Finally the program will performs an operation for these binary images. The result of the operation would be a single area of white pixel (with black pixel as background). This white pixel area describes the maximum HSV value in the image.

4.2 Discussion

4.2.1 Technique Chosen

RGB Technique has been chosen to classify the ripeness of oil palm fruit in order to get optimum of oil. The reasons for choosing RGB technique are:

- Color of oil palm fruit is comprised with primary color (R, G, B).
- Oil palm fruit has color that easily distinguished from natural environment.
- RGB color info or Red, Green and Blue color component already supplied by the camera and image.
- The values of Red, Green and Blue color component can be simply compared.
- RGB is more sensitive to color; shows more color region.

4.2.2 Problems Involved

- Changes in weather's condition cause illumination changes.
- Differences between R, G, B color component during sunrise and sunset.

4.2.3 Process

These are the process that can be used in order to solve the problems:

- Finding desired color under different lighting condition.
- Comparing between values color component.

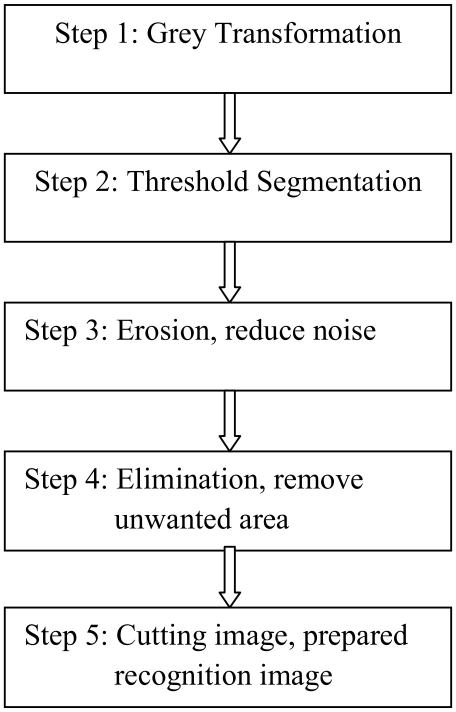

Step 1: Grey Transformation
Step 2: Threshold Segmentation
Step 3: Erosion, reduce noise
Step 4: Elimination, remove unwanted area
Step 5: Cutting image, prepared recognition image

Figure 30: Method for Ripeness Classification of Oil Palm Fruit using MATLAB-Image Processing Toolbox

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusion

This project involves detecting optimum quantity of oil based on the ripeness of oil palm fruit using suitable Digital Image Processing Techniques. The objective of this project is to develop a technique for classification of oil palm fruit using CCD camera and MATLAB-Image Processing Toolbox. Image analysis is used to extract the relevant features present in an image. This project focuses on main color techniques which are RGB and HSV. The simulation testing is done in order to get the good quality of images for optimum quantity of oil.

As a conclusion, this project achieved most of its objectives and proved its ability to be used as real application. Besides, image processing technique is successfully employed to develop techniques in classifying the ripeness level for oil palm fruit in order to get optimum quantity of oil based on the extracted RGB and HSV color feature of the fruit. Hopefully, with the existing of this system can be applied and will help people in agriculture. This will allow the people in agriculture to determine the grades of their fruit before selling them at the mill.

5.2 Recommendations

Future work can be done to improve the accuracy of the results as discussed above. The simulation done in this study is based on color analysis techniques of Digital Image Processing which are RGB and HSV techniques. Further improvement on the image simulation is necessary in order to make the analysis more realistic. Besides, in order to get more accurate results, all the images can be enhanced using artificial intelligence method such as Fuzzy-logic, Artificial Neural Network (ANN) and genetic algorithm. In fact, for better improvement the threshold or percentage value for unripe, ripe and overripe fruit can be classified as unripe when the value occurred is 80%, for ripe is between 80% to 90% and for overripe is over than 90%.

REFERENCES

[1] Gerald C. Holst, Terrence S. Lomheim. 2007, *CCD Sensors and Camera Systems,* SPIE Press & JCD Publishing

[2] John L. Semmlow. 2004, *Biosignal and Biomedical Image Processing,* Marcel Dekker, Inc.

[3] Rafael C. Gonzales, Richard E. Woods, Steven L. Eddins. 2004, *Digital Image Processing using Matlab,* Pearson Education, Inc.

[4] Meftah Salem M. Alfatni, Abdul Rashid Mohamed Shariff, Helmi Zulhaidi Mohd Shafri, Osama M. Ben Saaed, Omar M. Eshanta, 2008, " Oil Palm Fruit Bunch Grading System Using Red, Green and Blue Digital Number," *Asian Network for Scientific Information* : 2008

[5] Jing Zhang, Lei Wang, Longzheng Tong, 2007, "Feature Reduction and Texture Classification in MRI-Texture Analysis of Multiple Sclerosis," *IEEE/ICME International Conference on Complex Medical Engineering:* 2007

[6] Malaysian Oil palm fruit Industry Performance 2008. *Global Oils and Fats Business Magazine Vol.6* – Issue 1 (Jan-March), 2009.

[7] Sandesh B. Kamath, Shalini Chidambar, B. R. Brinda, M.A. Kumar, R. Sarada and G.A. Ravishandar, 2005, " *Digital Image Processing – An Alternate Tool For Monitoring Of Pigment Levels In Cultured Cells With Special Reference To Green Alga Haematococcus Pluvialis*," Central Food Technological Research Institute, Mysore 570020, India: 2005.

[8] M. Z. Abdullah, L. C. Guan, K. C. Lim and A. A. Karim, 2004, " *Application of Computer Vision in The Food Industry*," Journal of Food Engineering, 2004.

[9] Malaysian Oil palm fruit Board. *13 Apr 2010*<http://www.mpob.gov.my/>

[10] A. Materka, M.Strzelecki, 1998, "Texture *Analysis Methods* ," Technical University of Lodz, Institute of Electronics, COST B11 report, Brussels, 1998.

[11] Stephen Wang-Cheung Lam, "*Texture Feature Extraction Using Gray Level Gradient Based Co-occurrence Matrices,*" Department of Computer Science, University of Hong Kong, 1996.

[12] Robert M. Haralick, K. Shanmugam and Its'Hak Dinstein, "*Texture Features for Image Classification,*" IEEE Transactions on Systems, MAN and Cybernetics, November 1973.

[13] J.J Vaquero, F. del Pozo, E. J. Gotnez, "*RGB-HIS Technique for Multimodality Medical Image Display,*" Universidad Politecnica de Madrid, Spain, 1993.

[14] M. A. Tahic, A. Bouridane, F. Kurugollin and A. Amiru, 2004, "*Accelerating the Computation of GLCM and Haralick Texture Features on Reconfigurable Hardware,*" International Conference on Image Processing (ICIP), 2004.

[15] A. Khoshroo,A. Keyhani, R. A. Zoroofi, S. Rafiee, Z. Zamani, M. R. Alsharif, 2009, "*Classification of Pomegranate Fruit using Texture Analysis of MR Images,"* Agricultural Engineering International, March 2009.

[16] Farah Yasmin Adul Rahman, Shah Rizam Mohd Shah Baki, Ahmad Ihsan Mohd Yassin, Nooritawati Md. Tahir and Wan Illia Wan Ishak, 2009, "*Monitoring of Watermelon Ripeness Based on Fuzzy Logic,"* World Congress on Computer Science and Information Engineering, 2009.

[17] Yew Ai Tan, Kum Wan Low, Chak Khiam Lee and Kum Sang Low, 2010, "*Imaging Technique for Quantification of Oil Palm Fruit Ripeness and Oil Content,"* European Journal of Lipid Science and Technology, 2010.

[18] R.M. Hudzari, W.I. Wan Ishak and M.M. Noorman, 2010, "*Parameter Acceptance of Software Development for Oil Palm Fruit Maturity Prediction,"* Journal of Software Engineering 4 (3): 244-256, 2010.

[19] Idris Omar, Mohd Ashhar Khalid, Mohd Haniff Harun and Mohd Basri Wahid, 2003, "*Colour Meter for Measuring Fruit Ripeness,"* Malaysian Palm Oil Board, Ministry of Primary Industries, Malaysia, 2003.

[20] Jalani B. S., Yusof Basiron, Ariffin Darus, Chan K. W. and N. Rajanaidu, 2002, "*Prospects of Elevating National Oil Palm Productivity: A Malaysian Perspective,"* Oil Palm Industry Economic Journal Vol. 2(2), 2002.

[21] Kamarul Hawari Ghazali, Mohd. Marzuki Mustafa and Aini Hussain, 2008, "*Machine Vision System for Automatic Weeding Strategy using Image Processing Technique,"* American-Eurasian J. Agric. & Environ. Sci., 3 (3): 451-458, 2008.

[22] M.Z. Abdullah, L.C. Guan, Y.P. Ean, A.M.D. Manan and B.M.N. Mohdazemi, 2008, "*Using Machine Vision to Inspect Oil Palm and White Powder Starch,"* Crop Research & Application Unit (CRAUN), 2001.

[23] Ubong Lydia Jau, Chee Siong The and Giap Weng Ng, 2008, "*A Comparison of RGB and HIS Color Segmentation in Real – Time Video Images: A Preliminary Study on Road Sign Detection,"* EDT from IEEE Xplore, 2008.

APPENDIX A

Gantt Chart for FYP I

No	Detail/ Week	1	2	3	4	5	6	7		8	9	10	11	12	13	14
1	Selection of Project Topic	🟧	🟧						Mid-semester break							
2	Preliminary Research Work		🟧													
3	Submission of Preliminary Report		🟥													
4	Project Work; Research on Color Analysis, Image Processing and Oil Palm fruit characteristic			🟧	🟧	🟧	🟧	🟧								
5	Submission of Progress Report									🟥						
6	Seminar										🟧	🟧	🟧	🟧		
7	Project Work Continue; Build Coding System and Simulation									🟧	🟧	🟧	🟧	🟧		
8	Submission of Interim Report Final Draft														🟥	
9	Oral Presentation															🟥

🟥 Suggested milestone

🟧 Process

Gantt Chart for FYP II

No	Detail/ Week	1	2	3	4	5	6	7		8	9	10	11	12	13	14
1	Project Work Continue	▨	▨	▨					Mid-semester break							
2	Submission of Progress Report 1				🟥											
3	Project Work Continue				▨	▨	▨	▨								
4	Submission of Progress Report 2									🟥						
5	Seminar (compulsory)										▨	▨	▨			
6	Project Work Continue									▨	▨	▨	▨			
7	Poster Exhibition											🟥				
8	Submission of Dissertation (soft bound)													🟥		
9	Oral Presentation														🟥	
10	Submission of Project Dissertation (hard bound)															🟥

🟥 Suggested milestone

▨ Process

APPENDIX B

Coding Estimation

Coding Estimation Using Command Window

```
>> SimpleOilPalmClassifier
```

The Image processing for ripe Fruit starting from image 1 to image 50:

```
Processing ripe image 1
Processing ripe image 2
Processing ripe image 3
Processing ripe image 4
Processing ripe image 5
Processing ripe image 6
Processing ripe image 7
Processing ripe image 8
Processing ripe image 9
Processing ripe image 10
Processing ripe image 11
Processing ripe image 12
Processing ripe image 13
Processing ripe image 14
Processing ripe image 15
```

The system will continue processing for overripe fruit starting from image 1 to image 66:

```
Processing overrripe image: t53.bmp
Processing overrripe image: t54.bmp
Processing overrripe image: t55.bmp
Processing overrripe image: t56.bmp
Processing overrripe image: t57.bmp
Processing overrripe image: t58.bmp
Processing overrripe image: t59.bmp
Processing overrripe image: t6.bmp
Processing overrripe image: t60.bmp
Processing overrripe image: t61.bmp
Processing overrripe image: t62.bmp
Processing overrripe image: t63.bmp
```

Finally, the system will process for unripe image starting from image 1 to image 54:

```
Processing underripe image: u12.bmp
Processing underripe image: u13.bmp
Processing underripe image: u14.bmp
Processing underripe image: u15.bmp
Processing underripe image: u16.bmp
Processing underripe image: u17.bmp
Processing underripe image: u18.bmp
Processing underripe image: u19.bmp
Processing underripe image: u20.bmp
Processing underripe image: u21.bmp
Processing underripe image: u22.bmp
Processing underripe image: u23.bmp
Processing underripe image: u24.bmp

End of Operation
```

M-file

```
rgbImage = imread('ripe1.bmp');
imshow(rgbImage);

%total value of red, green and blue for each
redChannel = rgbImage(:,:, 1);
greenChannel = rgbImage(:,:, 2);
blueChannel = rgbImage(:,:, 3);
redMean = mean(mean(redChannel))
greenMean = mean(mean(blueChannel))
blueMean = mean(mean(blueChannel))
message = sprintf('The red mean = %.2f\n The green mean = %.2f\n The
blue mean = %.2f')
msgbox(message);
```

Simple Oil Palm Classifier

```
function varargout = SimpleOilPalmClassifier(varargin)
% SIMPLEOILPALMCLASSIFIER M-file for SimpleOilPalmClassifier.fig
%      SIMPLEOILPALMCLASSIFIER, by itself, creates a new SIMPLEOILPALMCLASSIFIER or
raises the existing
%      singleton*.
%
%      H = SIMPLEOILPALMCLASSIFIER returns the handle to a new SIMPLEOILPALMCLASSIFIER
or the handle to
%      the existing singleton*.
%
%      SIMPLEOILPALMCLASSIFIER('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SIMPLEOILPALMCLASSIFIER.M with the given input
arguments.
%
%      SIMPLEOILPALMCLASSIFIER('Property','Value',...) creates a new
SIMPLEOILPALMCLASSIFIER or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before SimpleOilPalmClassifier_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to SimpleOilPalmClassifier_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help SimpleOilPalmClassifier

% Last Modified by GUIDE v2.5 07-Oct-2010 21:49:21

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @SimpleOilPalmClassifier_OpeningFcn, ...
                   'gui_OutputFcn',  @SimpleOilPalmClassifier_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before SimpleOilPalmClassifier is made visible.
function SimpleOilPalmClassifier_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to SimpleOilPalmClassifier (see VARARGIN)

% Choose default command line output for SimpleOilPalmClassifier

set([handles.edtH1 handles.edtH2],'Enable','off');
handles.output = hObject;

handles.imageinput = [];
handles.ripemeanred = 0;
handles.uripemeanred = 0;
handles.oripemeanred = 0;
handles.arrayripergb = [];
handles.arrayuripergb = [];
handles.arrayoripergb = [];

% handles.ripemeanh = 0;
% handles.ripemeans = 0;
% handles.ripemeanv = 0;
handles.uripemeana = 0;
% handles.uripemeanh = 0;
% handles.uripemeans = 0;
% handles.uripemeanv = 0;
handles.uripemeana = 0;
% handles.oripemeanh = 0;
% handles.oripemeans = 0;
% handles.oripemeanv = 0;
handles.oripemeana = 0;
% handles.arrayripehsv = [];
% handles.arrayuripehsv = [];
% handles.arrayoripehsv = [];
handles.arrayuripehsv = [];
handles.arrayoripehsv = [];
handles.arrayripehsv = [];
```

```
% --- Outputs from this function are returned to the command line.
function varargout = SimpleOilPalmClassifier_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in btnURipeFolder.
function btnURipeFolder_Callback(hObject, eventdata, handles)
% hObject    handle to btnURipeFolder (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
dirName = uigetdir();
if dirName ~= 0
    set(handles.editURipeFolder,'string',dirName);
end

% --- Executes on button press in btnORipeFolder.
function btnORipeFolder_Callback(hObject, eventdata, handles)
% hObject    handle to btnORipeFolder (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
dirName = uigetdir(path);
if dirName ~= 0
    set(handles.editORipeFolder,'string',dirName);
end

% --- Executes on button press in btnRipeFolder.
function btnRipeFolder_Callback(hObject, eventdata, handles)
% hObject    handle to btnRipeFolder (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
dirName = uigetdir(path);
if dirName ~= 0
    set(handles.editRipeFolder,'string',dirName);
end

function editRipeFolder_Callback(hObject, eventdata, handles)
% hObject    handle to editRipeFolder (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editRipeFolder as text
%        str2double(get(hObject,'String')) returns contents of editRipeFolder as a
double


% --- Executes during object creation, after setting all properties.
function editRipeFolder_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editRipeFolder (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
function editORipeFolder_Callback(hObject, eventdata, handles)
% hObject    handle to editORipeFolder (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function editURipeFolder_Callback(hObject, eventdata, handles)
% hObject    handle to editURipeFolder (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in btnGetTh.
function btnGetTh_Callback(hObject, eventdata, handles)
% hObject    handle to btnGetTh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% % handles    structure with handles and user data (see GUIDATA)
if(get(handles.popupmenu1,'Value') == 1)
    ripeprocessingc(handles);
    uripeprocessingc(handles);
    oripeprocessingc(handles);
else
    ripeprocessinghsv(handles);
    uripeprocessinghsv(handles);
    oripeprocessinghsv(handles);
end
disp('End of Operation');


% --- Executes on button press in btnSaveTh.
function btnSaveTh_Callback(hObject, eventdata, handles)
% hObject    handle to btnSaveTh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%% load threshold value from database
% --- Executes on button press in btnLoadTh.
function btnLoadTh_Callback(hObject, eventdata, handles)
% hObject    handle to btnLoadTh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if(get(handles.popupmenu1,'Value') = 1)
    [file,path] = uigetfile('*.mat','Select the MAT-file');
    if(file ~= 0)
        % show RIPE
        set(handles.edtrr,'String',num2str(ripemeanred));
        % show URIPE
        set(handles.edtur,'String',num2str(uripemeanred));
        % show ORIPE
        set(handles.edtor,'String',num2str(oripemeanred));
    end
else
    [file,path] = uigetfile('*.mat','Select the MAT-file');
    if(file ~= 0)
        filenamefull = fullfile(path,file);
        load(filenamefull);
        % show RIPE
        set(handles.edthsvr,'String',num2str(ripemeanarea));
        % show URIPE
        set(handles.edthsvu,'String',num2str(uripemeanarea));
        % show ORIPE
        set(handles.edthsvo,'String',num2str(oripemeanarea));
    end
end

% --- Executes on button press in btnLoadTest.
function btnLoadTest_Callback(hObject, eventdata, handles)
% hObject    handle to btnLoadTest (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
loadImageForTesting(handles);

% --- Executes on button press in btnTest.
function btnTest_Callback(hObject, eventdata, handles)
% hObject    handle to btnTest (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if(get(handles.popupmenu1,'Value');
    IdentifyRipenessc(handles)
else
    IdentifyRipenessHSV(handles)
end
%% save threshold RGB
function saveThresholdRGB(handles)
ripemeanred = handles.ripemeanred;
uripemeanred = handles.uripemeanred;
oripemeanred = handles.oripemeanred;
arrayripergb = handles.arrayripergb;
arrayuripergb = handles.arrayuripergb;
arrayoripergb = handles.arrayoripergb;

[file,path] = uiputfile('data_threshold_rgb.mat','Save file name');
if (file ~= 0 )
    save(path,file)

'ripemeanred','uripemeanred','oripemeanred','arrayripergb','arrayuripergb','arrayoriperg
b');
end

%% save threshold HSV
function saveThresholdHSV(handles)
ripemeanarea = handles.ripemeana;
uripemeanarea = handles.uripemeana;
oripemeanarea = handles.oripemeana;
arrayripehsv = handles.arrayripehsv;
arrayuripehsv = handles.arrayuripehsv;
arrayoripehsv = handles.arrayoripehsv;

[file,path] = uiputfile('data_threshold_hsv.mat','Save file name');
if (file ~= 0 )
    save(path,file)

'ripemeanarea','uripemeanarea','oripemeanarea','arrayripehsv','arrayuripehsv','arrayorip
ehsv')
end


%% load single image for testing
function loadImageForTesting
[file, path] = uigetfile({'*.bmp';'*.jpg'},'Select an image file');
if(file ~= 1);
    handles = guidata(handles.figure1);
    set(handles.textRipe,'String','');
    imgInput = imread(fullfile(path,file));
    handles.imageinput = imgInput;
    guidata(handles.figure1, handles);
    imshow(imgInput);
    text(20,150,file,'Color','w');
end

%% Identify Ripeness of single fruit using RGB technique
% calculate mean color
imgInput2 = double(imgInput);
qlevel;
[maps,imgInput2] = srm(imgInput2,qlevel);
mask = srm_boundaries(imgInput2,maps);
s = regionprops(BW,'Centroid');
x = round(s.Centroid(1)); y = round(s.Centroid(2));
imc = imcrop(imgInput,[x-width/4 y-height/4 width/2 height/2]);
RedMean = (mean(mean(double(imc(:,:,1))))/255)*100;
set(handles.textRipe,'String',['red:',num2str(RedMean)],'FontSize',8);
% get range value for thresholding: from database/real value
if(get(handles.popupmenu2,'Value') == 1)
    redUripe = str2double(get(handles.edtur,'String'))
    redRipe  = str2double(get(handles.edtrr,'String'))
    redORipe = str2double(get(handles.edtor,'String'))
    redRipeMeanMin = redUripe + ((redRipe - redUripe)/2);
    redRipeMeanMax = redRipe + ((redORipe - redRipe)/2);
    if RedMean <= redRipeMeanMin
        strR = 'UNDERRIPE';
    elseif RedMean > redRipeMeanMin && RedMean <= redRipeMeanMax
        strR = 'RIPE';
    elseif RedMean > redRipeMeanMax
        strR = 'OVERRIPE';
    end
% get range value for thresholding: from user input value
else
    if RedMean <= str2double(get(handles.edtR1,'String'))
        strR = 'UNDERRIPE';
    elseif RedMean > str2double(get(handles.edtR1,'String')) && RedMean <=
str2double(get(handles.edtR2,'String'))
        strR = 'RIPE';
    elseif RedMean > str2double(get(handles.edtR2,'String'))
        strR = 'OVERRIPE';
    end
end
text(20,250,strR,'Color','w','FontSize',10);

%% Identify Ripeness of single fruit using HSV technique
function IdentifyRipenessHSV(handles)
% calculate mean color
imgInput2 = double(imgInput);
qlevel = 4;
[maps,imgInput2] = srm(imgInput2,qlevel);
mask = srm_boundaries(imgInput2,maps);
s = regionprops(BW,'Centroid');
x = round(s.Centroid(1)); y = round(s.Centroid(2));
imc = imcrop(imgInput,[x-width/4 y-height/4 width/2 height/2]);
% convert to HSV colorspace
hsvImage = rgb2hsv(imc);
hImage = hsvImage(:,:,1);
sImage = hsvImage(:,:,2);
vImage = hsvImage(:,:,3);
% Assign the low and high thresholds for each color band.
hueThresholdLow = 0;
hueThresholdHigh = graythresh(hImage);
saturationThresholdLow = graythresh(sImage);
saturationThresholdHigh = 1.0;
valueThresholdLow = graythresh(vImage);
valueThresholdHigh = 1.0;
% Now apply each color band's particular thresholds to the color band
hueMask = (hImage >= hueThresholdLow) & (hImage <= hueThresholdHigh);
saturationMask = (sImage >= saturationThresholdLow) & (sImage <=
saturationThresholdHigh);
valueMask = (vImage >= valueThresholdLow) & (vImage <= valueThresholdHigh);
% Then we will have the mask of only the red parts of the image.
regionObjectsMask = uint8(hueMask & saturationMask & valueMask);
% Remove small objects.
regionObjectsMask = uint8(bwareaopen(regionObjectsMask, 100));
% Smooth the border using a morphological closing operation, imclose().
SE = strel('disk', 4);
regionObjectsMask = imclose(regionObjectsMask, SE);
% Fill in any holes in the regions, since they are most likely red also.
regionObjectsMask = uint8(imfill(regionObjectsMask, 'holes'));
% Convert to input image type
regionObjectsMask = cast(regionObjectsMask, class(imgInput));
% Measure the mean HSV and area of all the detected blobs.
[meanHSV, areas, numberOfBlobs] = MeasureBlobs(regionObjectsMask, hImage, sImage,
vImage);
% store values in array for total mean measurement
f = 1;
if(numberOfBlobs > 1)
    areas_row = areas(:,1)
    areas = max(areas_row)
    f = find(areas_row==max(areas_row))
end
areas = max(areas);
set(handles.textRipe,'String',['HSV Area:',num2str(areas)],'FontSize',8);
% get range value for thresholding: from database/real value
if(get(handles.popupmenu2,'Value') == 1)
    hsvURipe = str2double(get(handles.edthsvu,'String'));
    hsvRipe  = str2double(get(handles.edthsvr,'String'));
    hsvORipe = str2double(get(handles.edthsvo,'String'));
    RipeHSVMin = hsvORipe + ((hsvURipe - hsvORipe)/2);
    RipeHSVMax = hsvURipe + ((hsvRipe - hsvURipe)/2);
    if areas <= RipeHSVMin
        strR = 'OVERRIPE';
    elseif areas > RipeHSVMin && areas <= RipeHSVMax
        strR = 'UNDERRIPE';
    elseif areas > RipeHSVMax
        strR = 'RIPE';
    end
% get range value for thresholding: from user input value
else
    if areas <= str2double(get(handles.edtH1,'String'))
        strR = 'OVERRIPE';
    elseif areas > str2double(get(handles.edtH1,'String')) && areas <=
str2double(get(handles.edtH2,'String'))
        strR = 'UNDERRIPE';
    elseif areas > str2double(get(handles.edtH2,'String'))
        strR = 'RIPE';
    end
end
text(20,250,strR,'Color','w','FontSize',10);


%% process images inside ripe folder using RGB technique
function ripeprocessingc(handles)
if(~isempty(get(handles.editRipeFolder,'string')))
    files = dir(get(handles.editRipeFolder,'string'));
    j = 0;
    RedRipeMean = []; RedMean = 0;
    for i=1:size(files)
        if(files(i).isdir == 0)
            try
                imgInput =
imread(strcat(get(handles.editRipeFolder,'string'),'\',files(i).name));
                j = j + 1;
                if j == 1
                    height = size(imgInput);
                    width  = size(imgInput);
                end
                disp(['(C)Processing ripe image: ',files(i).name]);
                imgInput2 = double(imgInput);
                qlevel = 4;
                s = regionprops(BW,'Centroid');
                x = round(s.Centroid(1)); y = round(s.Centroid(2));
                imc = imcrop(imgInput,[x-width/4 y-height/4 width/2 height/2]);
                RedMean = (mean(mean(double(imc(:,:,1))))/255)*100; RedRipeMean(j) =
RedMean;
            catch
            end
        end
    end
    % mean value: RIPE
    RipeRedMean= mean(RedRipeMean);%sum(RedRipeMean)/length(RedRipeMean);
    set(handles.edtrr,'String',num2str(RipeRedMean));

%% process images inside underripe folder using RGB technique
function uripeprocessingc(handles)
if(~isempty(get(handles.editURipeFolder,'string')))
    files = dir(get(handles.editURipeFolder,'string'));
    j = 0;
    RedRipeMean = []; RedMean = 0;
    for i=1:size(files)
        if(files(i).isdir == 0)
                imgInput =
imread(strcat(get(handles.editURipeFolder,'string'),'\',files(i).name));
                j = j + 1;
                disp(['(C)Processing underripe image: ',files(i).name]);
                imgInput2 = double(imgInput);
                qlevel = 4;
                [maps,imgInput2] = srm(imgInput2,qlevel);
                mask = srm_boundaries(imgInput2,maps);
%                 idx = find(mask==1);
                BW = uint8(mask);
                s = regionprops(BW,'Centroid');
%                 figure, imshow(imgInput), hold on,
                x = round(s.Centroid(1)); y = round(s.Centroid(2));
%                 plot(x,y,'r*'), hold off;
                imc = imcrop(imgInput,[x-width/4 y-height/4 width/2 height/2]);
%                 figure, imshow(imc);
                RedMean = (mean(mean(double(imc(:,:,1))))/255)*100; RedRipeMean(j) =
RedMean;
            end
        end
    end
    % mean value: URIPE
    RipeRedMean = mean(RedRipeMean);%sum(RedRipeMean)/length(RedRipeMean);
    set(handles.edtur,'String',num2str(RipeRedMean));

%% process image inside overripe folder using RGB technique
function oripeprocessingc(handles)
if(~isempty(get(handles.editORipeFolder,'string')))
    files = dir(get(handles.editORipeFolder,'string'));
    j = 0;
    RedRipeMean = []; GreenRipeMean = []; BlueRipeMean = [];
    RedMean = 0; GreenMean = 0; BlueMean = 0;
    for i=1:size(files)
        if(files(i).isdir == 0)
                imgInput =
imread(strcat(get(handles.editORipeFolder,'string'),'\',files(i).name));
                j = j + 1;
                disp(['(C)Processing overrripe image: ',files(i).name]);
                imgInput2 = double(imgInput);
                qlevel = 4;
                [maps,imgInput2] = srm(imgInput2,qlevel);
                mask = srm_boundaries(imgInput2,maps);
%                 idx = find(mask==1);
                BW = uint8(mask);
                s = regionprops(BW,'Centroid');
%                 figure, imshow(imgInput), hold on,
                x = round(s.Centroid(1)); y = round(s.Centroid(2));
%                 plot(x,y,'r*'), hold off;
                imc = imcrop(imgInput,[x-width/4 y-height/4 width/2 height/2]);
%                 figure, imshow(imc);
                RedMean = (mean(mean(double(imc(:,:,1))))/255)*100; RedRipeMean(j) =
RedMean;
            end
        end
    end
    RipeRedMean = mean(RedRipeMean);%sum(RedRipeMean)/length(RedRipeMean);
    set(handles.edtor,'String',num2str(RipeRedMean));

%% process image inside ripe folder using HSV
function ripeprocessinghsv(handles)
if(~isempty(get(handles.editRipeFolder,'string')))
    files = dir(get(handles.editRipeFolder,'string'));
    j = 0;
    AllMeanH = []; AllMeanV = []; AllMeanS = []; AllAreas = [];
    for i=1:size(files)
        (files(i).isdir == 0)
            try
%                 try to read image
                imgInput =
imread(strcat(get(handles.editRipeFolder,'string'),'\',files(i).name));
                end
%                 cropping image
                disp(['(H)Processing ripe image: ',files(i).name]);
                imgInput2 = double(imgInput);
                qlevel = 4;
                [maps,imgInput2] = srm(imgInput2,qlevel);
                mask = srm_boundaries(imgInput2,maps);
                BW = uint8(mask);
                s = regionprops(BW,'Centroid');
                x = round(s.Centroid(1)); y = round(s.Centroid(2));
                imc = imcrop(imgInput,[x-width/4 y-height/4 width/2 height/2]);
%                 convert to HSV colorspace
                hsvImage = rgb2hsv(imc);
                hImage = hsvImage(:,:,1);
                sImage = hsvImage(:,:,2);
                vImage = hsvImage(:,:,3);
%                 Assign the low and high thresholds for each color band.
                hueThresholdLow = 0;
                hueThresholdHigh = graythresh(hImage);
                saturationThresholdLow = graythresh(sImage);
                saturationThresholdHigh = 1.0;
                valueThresholdLow = graythresh(vImage);
                valueThresholdHigh = 1.0;
%                 Now apply each color band's particular thresholds to the color band
                hueMask = (hImage >= hueThresholdLow) & (hImage <= hueThresholdHigh);
                saturationMask = (sImage >= saturationThresholdLow) & (sImage <=
saturationThresholdHigh);
                valueMask = (vImage >= valueThresholdLow) & (vImage <=
valueThresholdHigh);
%                 Remove small objects.
                regionObjectsMask = uint8(bwareaopen(regionObjectsMask, 100));
%                 Smooth the border using a morphological closing operation, imclose().
                SE = strel('disk', 4);
                regionObjectsMask = imclose(regionObjectsMask, SE);
%                 Fill in any holes in the regions, since they are most likely red also.
                regionObjectsMask = uint8(imfill(regionObjectsMask, 'holes'));
%                 Convert to input image type
                regionObjectsMask = cast(regionObjectsMask, class(imgInput));
%                 Measure the mean HSV and area of all the detected blobs.
                [meanHSV, areas, numberOfBlobs] = MeasureBlobs(regionObjectsMask,
hImage, sImage, vImage);
%                 store values in array for total mean measurement
                f = 1;
                if(numberOfBlobs > 1)
                    areas_row = areas(:,1);
                    areas = max(areas_row);
                    f = find(areas_row==max(areas_row));
                end
                AllMeanH(j) = meanHSV(:,1);
                AllMeanS(j) = meanHSV(:,2);
                AllMeanV(j) = meanHSV(:,3);
                AllAreas(j) = max(areas);
            catch
            end
        end
    end
    % mean value of HSV Area
    MA = mean(AllAreas);
    set(handles.edthsvr,'String',num2str(MA));
    % update handles
    handles = guidata(handles.figure1);
    handles.ripemeana = MA;
    handles.arrayripehsv = AllAreas;
    guidata(handles.figure1, handles);
end

%% process image inside uripe folder using HSV
function uripeprocessinghsv(handles)
if(~isempty(get(handles.editURipeFolder,'string')))
    files = dir(get(handles.editURipeFolder,'string'));
    j = 1;
    AllMeanH = []; AllMeanV = []; AllMeanS = []; AllAreas = [];
    for i=1:size(files)
        if(files(i).isdir == 1)
            try
%                 try to read image
```

```
                imgInput = 
imread(strcat(get(handles.editURipeFolder,'string'),'\',files(i).name));
                j = j + 1;
%                 cropping image
                disp(['(H)Processing underripe image: ',files(i).name]);
                imgInput2 = double(imgInput);
                qlevel = 4;
                [maps,imgInput2] = srm(imgInput2,qlevel);
                mask = srm_boundaries(imgInput2,maps);
                BW = uint8(mask);
                s = regionprops(BW,'Centroid');
                x = round(s.Centroid(1)); y = round(s.Centroid(2));
                imc = imcrop(imgInput,[x-width/4 y-height/4 width/2 height/2]);
%                 convert to HSV colorspace
                hsvImage = rgb2hsv(imc);
                hImage = hsvImage(:,:,1);
                sImage = hsvImage(:,:,2);
                vImage = hsvImage(:,:,3);
%                 Assign the low and high thresholds for each color band.
                hueThresholdLow = 0;
                hueThresholdHigh = graythresh(hImage);
                saturationThresholdLow = graythresh(sImage);
                saturationThresholdHigh = 1.0;
                valueThresholdLow = graythresh(vImage);
                valueThresholdHigh = 1.0;
%                 Now apply each color band's particular thresholds to the color band
                hueMask = (hImage >= hueThresholdLow) & (hImage <= hueThresholdHigh);
                saturationMask = (sImage >= saturationThresholdLow) & (sImage <= 
saturationThresholdHigh);
                valueMask = (vImage >= valueThresholdLow) & (vImage <= 
valueThresholdHigh);
%                 Combine the masks to find where all 3 are "true."
%                 Then we will have the mask of only the red parts of the image.
                regionObjectsMask = uint8(hueMask & saturationMask & valueMask);
%                 Remove small objects.
                regionObjectsMask = uint8(bwareaopen(regionObjectsMask, 100));
%                 Smooth the border using a morphological closing operation, imclose().
                SE = strel('disk');
                regionObjectsMask = imclose(regionObjectsMask, SE);
%                 Fill in any holes in the regions, since they are most likely red also.
                regionObjectsMask = uint8(imfill(regionObjectsMask, 'holes'));
%                 Convert to input image type
                regionObjectsMask = cast(regionObjectsMask, class(imgInput));
%                 Measure the mean HSV and area of all the detected blobs.
                [meanHSV, areas, numberOfBlobs] = MeasureBlobs(regionObjectsMask, 
hImage, sImage, vImage);
%                 store values in array for total mean measurement
                f = 1;
                if(numberOfBlobs > 1)
                    areas_row = areas;
                    areas = max(areas_row);
                    f = find(areas_row==max(areas_row));
                end
                AllMeanH = meanHSV(f,1);
                AllMeanS = meanHSV(f,2);
                AllMeanV = meanHSV(f,3);
                AllAreas = max(areas);
            catch
            end
        end
    end
    % mean value of HSV Area
    MA = mean(AllAreas);
    set(handles.edthsvu,'String',num2str(MA));

%% process image inside oripe folder using HSV
function oripeprocessinghsv(handles)
if(~isempty(get(handles.editORipeFolder,'string')))
    files = dir(get(handles.editORipeFolder,'string'));
    j = 1;
    AllMeanH = []; AllMeanV = []; AllMeanS = []; AllAreas = [];
    for i=1:size(files)
        if(files(i).isdir == 0)
            try
%                 try to read image
                imgInput =
imread(strcat(get(handles.editORipeFolder,'string'),'\',files(i).name));
                j = j + 1;
%                 cropping image
                disp(['(H)Processing overripe image: ',files(i).name]);
                imgInput2 = double(imgInput);
                qlevel = 4;
                [maps,imgInput2] = srm(imgInput2,qlevel);
                mask = srm_boundaries(imgInput2,maps);
                BW = uint8(mask);
                s = regionprops(BW,'Centroid');
                x = round(s.Centroid(1)); y = round(s.Centroid(2));
                imc = imcrop(imgInput,[x-width/4 y-height/4 width/2 height/2]);
%                 convert to HSV colorspace
                hsvImage = rgb2hsv(imc);
                hImage = hsvImage(:,:,:);
                sImage = hsvImage(:,:,:);
                vImage = hsvImage(:,:,:);
%                 Assign the low and high thresholds for each color band.
                hueThresholdLow = 0;
                hueThresholdHigh = graythresh(hImage);
                saturationThresholdLow = graythresh(sImage);
                saturationThresholdHigh = 1;
                valueThresholdLow = graythresh(vImage);
                valueThresholdHigh = 1;
%                 Now apply each color band's particular thresholds to the color band
                hueMask = (hImage >= hueThresholdLow) & (hImage <= hueThresholdHigh);
                saturationMask = (sImage >= saturationThresholdLow) & (sImage <=
saturationThresholdHigh);
                valueMask = (vImage >= valueThresholdLow) & (vImage <=
valueThresholdHigh);
%                 Combine the masks to find where all 3 are "true."
%                 Then we will have the mask of only the red parts of the image.
                regionObjectsMask = uint8(hueMask & saturationMask & valueMask);
%                 Remove small objects.
                regionObjectsMask = uint8(bwareaopen(regionObjectsMask, 100));
%                 Smooth the border using a morphological closing operation, imclose().
                SE = strel('disk');
                regionObjectsMask = imclose(regionObjectsMask, SE);
%                 Fill in any holes in the regions, since they are most likely red also.
                regionObjectsMask = uint8(imfill(regionObjectsMask, 'holes'));
%                 Convert to input image type
                regionObjectsMask = cast(regionObjectsMask, class(imgInput));
%                 store values in array for total mean measurement
                f = 1;
                if(numberOfBlobs > 1)
                    areas_row = areas(:,0);
                    areas = max(areas_row);
                    f = find(areas_row==max(areas_row));
                end
                AllMeanH(j) = meanHSV(:,1);
                AllMeanS(j) = meanHSV(;,2);
                AllMeanV(j) = meanHSV(:,3);
                AllAreas(j) = max(areas);
            catch
            end
        end
    end
```