

**EDU_VISTEC: A SOFTWARE FOR COMPUTER VISION EDUCATIONAL
TRAINER USING MATLAB**

LAW KOK SIN

FINAL DRAFT THESIS

Bachelor Degree of Electrical Engineering (Electronic)

Faculty of Electrical & Electronic Engineering

Universiti Malaysia Pahang

11 NOVEMBER 2010

CHAPTER 1

INTRODUCTION

New technologies need to be exploring for the development a system for help in the human daily life. Exploring and learn a new technologies is not that easy because it need some guide for it. Nowadays, the application of image processing and analysis are now used in a wide range of industrial, artistic, and educational applications such as biotechnology, medicine, environmental science and art. Image processing in biotechnology or medicine also need use the basic application of object detection and color recognition. So, development of object detection and color recognition module need the intelligence on new technologies as a contribution due to help the industries and also education field. Designing this module is useful for the new learner and this module is a process to determine the shape detection and color recognition. In this module will be cover several method such as edge detection and using histogram. The basic tools that be used is MATLAB software. That is basic software that used in engineering field especially. Then implement MATLAB GUI which is one of the functions in MATLAB to design as simple module applications. Therefore, the new learner can follow step by step in this module to know more about the application of image processing that helpful in the entire field.

1.1 Problem Statement

Exploring a new technology like image processing is quite hard and need some guide for it. In the internet got a lot of the manual guide for it but it hard for new learner to really understand about it. Unfortunately, the guide from a module or system that can let the new learner to try it out some method that can be use for the image processing applications is much effective. So, a module about that is needed then this project is important to complete for that purpose.

1.2 Objective

This project aims to create a module about the application of image processing which using the MATLAB software as the main tools and use the application Graphics User Integrated (GUI) skill. In these modules which cover two parts, shape detection (circle and square) and color recognitions (RGB). Both of these basic applications are suitable for the new learner for their understanding to help in their learning process so, that is the main purpose of create this module.

1.3 Scope of project

The related scopes of this project are basic application of image processing and MATLAB software. It involves data collection, image acquisition, image processing,

classification and decision. The data collection involves collecting the image consists of different shape or color. Image processing consist of shape detection and color recognition that can be many type of techniques but we used edge detection and histogram for the two applications.

MATLAB Software is utilized where .m file as the location to write program and form linkages between main program and sub programs, also, as the platform where ANN program is trained to be accurate, efficient and user friendly.

1.3 Thesis Outline

This thesis is organized as below:

Chapter 1 will describes the introduction of this system, the purpose of this project, problem statement, the work scope and brief explanation of this project.

In **Chapter 2**, the reviews about the information find on all the material or data used include the software in the development of this project will be shown.

Chapter 3 will explain about all the methods use in development of this system and also step by step on develop the module for training purpose and lastly described about the execution part.

Chapter 4 will show all the results followed by the explanation and discussion about the results from the beginning step until the end of development module.

Last chapter of **Chapter 5** will have a summary to describe the overall part of this project and come up with some recommendations and improvement.

Chapter 2

LITERATURE REVIEW

This chapter will review on the information gathered in developing the simple system or module for object detection and color recognition. The information is the entire basic introduction to develop that system for learning purpose including the basic knowledge about the image processing, creating standalone file and also interfacing MATLAB with GUI.

2.1 Image processing

Image processing is the analysis of a picture using techniques that can identify shades, color and relationships that cannot be perceived by the human eye. Image processing is used to solve identification problems, such as in forensic medicine or in creating weather maps from satellite pictures. [1] It deals with images in bitmapped graphics format that have been scanned in or captured with digital cameras.

Image processing typically attempts to accomplish into three parts that is restoring images, enhancing images and understanding images. Restoration process takes a corrupted image and attempts to recreate a clean original image which removing sensor noise and restoring old, archived film and images. [1]



Figure 2.1: Example of restoration image

Enhancement process alters an image to make its meaning clearer to human observers which often user used to increase the contrast in images that are overly dark or light.



Figure 2.2: Example of enhancement image

Understanding process usually attempts to mimic the human visual system in extracting meaning from an image which includes many different tasks like segmentation, classification and interpretation. The process begin by identifying object in an image, then assigns labels to individual pixels and extract some meaning from the image as a whole.

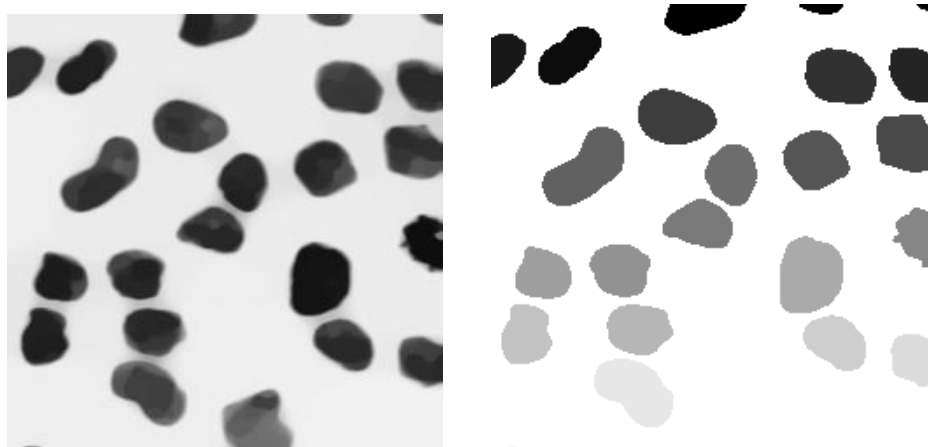


Figure 2.3: Example of understanding images

2.1.1 Threshold image

In many vision applications, it is useful to be able to separate out the regions of the image corresponding to objects in which we are interested, from the regions of the image that correspond to background. Thresholding often provides an easy and convenient way to perform this segmentation on the basis of the different intensities or colors in the foreground and background regions of an image.

Threshold converts each pixel into black, white or unchanged depending on whether the original color value is within the threshold range. Threshold is a very important command that is often used to prepare scanned RGB or RGBa images for vectorization or use as guide layers in the creation of drawings. It can be used with raster data images to set off ranges of values that may then be used for subsequent analysis or as selection masks. [4]

In addition, it is often useful to be able to see what areas of an image consist of pixels whose values lie within a specified range, or *band* of intensities (or colors). Thresholding can be used for this as well. Thresholding is the simplest method of image segmentation. From a grayscale image, thresholding can be used to create binary images.

The input to a thresholding operation is typically a grayscale or color image. In the simplest implementation, the output is a binary image representing the segmentation. Black pixels correspond to background and white pixels correspond to foreground (or *vice versa*). In simple implementations, the segmentation is determined by a single parameter known as the intensity threshold. In a single pass, each pixel in the image is compared with this threshold. If the pixel's intensity is higher than the threshold, the

pixel is set to, say, white in the output. If it is less than the threshold, it is set to black.
[4]

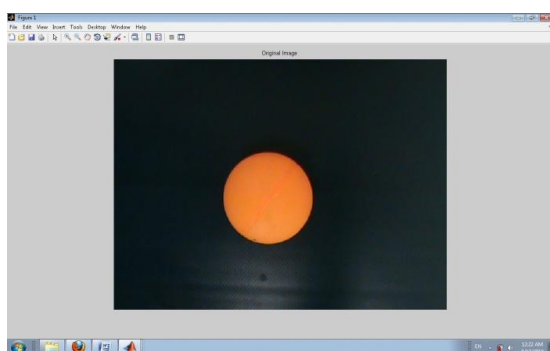


Figure 2.4: Original image

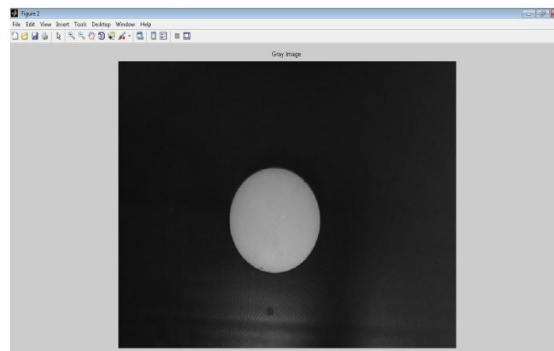


Figure 2.5: RGB to gray scale

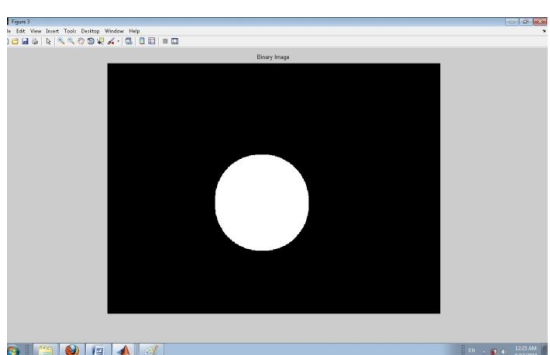


Figure 2.6: Threshold to binary image

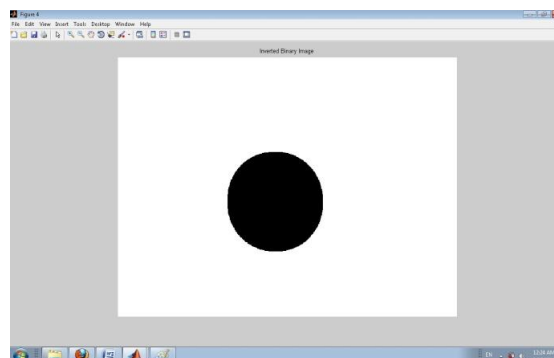


Figure 2.7: Inverse binary image

In more sophisticated implementations, multiple thresholds can be specified, so that a *band* of intensity values can be set to white while everything else is set to black. For color or multi-spectral images, it may be possible to set different thresholds for each color channel, and so select just those pixels within specified cuboids in RGB space. Another common variant is to set to black all those pixels corresponding to background,

but leave foreground pixels at their original color/intensity (as opposed to forcing them to white), so that that information is not lost.

There are categorizing the thresholding methods in six groups:

1. Histogram shape-based methods, where, for example, the peaks, valleys and curvatures of the smoothed histogram are analyzed.
2. Clustering-based methods, where the gray-level samples are clustered in two parts as background and foreground ~object! Or alternately are modeled as a mixture of two Gaussians.
3. Entropy-based methods result in algorithms that use the entropy of the foreground and background regions, the cross-entropy between the original and binaries image, etc.
4. Object attribute-based methods search a measure of similarity between the gray-level and the binaries images, such as fuzzy shape similarity, edge coincidence, etc.
5. The spatial methods use higher-order probability distribution and/or correlation between pixels
6. Local methods adapt the threshold value on each pixel to the local image characteristics.

2.1.2 Edge Detection

Edge detection is a fundamental tool used in most image processing applications to obtain information from the frames as a precursor step to feature extraction and object segmentation. This process detects outlines of an object and boundaries between objects and the background in the image. An edge-detection filter can also be used to improve the appearance of blurred or anti-aliased video streams.

The basic edge-detection operator is a matrix area gradient operation that determines the level of variance between different pixels. The edge-detection operator is calculated by forming a matrix centered on a pixel chosen as the center of the matrix area. If the value of this matrix area is above a given threshold, then the middle pixel is classified as an edge. [6]

Among the gradient-based detectors are Sobel, Prewitt, and Roberts. By default, edge uses the Sobel method to detect edges but the following provides a complete list of all the edge-finding methods supported by this function [6]:

- The Sobel method finds edges using the Sobel approximation to the derivative. It returns edges at those points where the gradient of I is maximum.
- The Prewitt method finds edges using the Prewitt approximation to the derivative. It returns edges at those points where the gradient of I is maximum.
- The Roberts method finds edges using the Roberts approximation to the derivative. It returns edges at those points where the gradient of I is maximum.
- The Laplacian of Gaussian method finds edges by looking for zero crossings after filtering I with a Laplacian of Gaussian filter.
- The zero-cross method finds edges by looking for zero crossings after filtering I with a filter you specify.

- The Canny method finds edges by looking for local maxima of the gradient of I . The gradient is calculated using the derivative of a Gaussian filter. The method uses two thresholds, to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. This method is therefore less likely than the others to be fooled by noise, and more likely to detect true weak edges.

The Prewitt operator measures two components. The vertical edge component is calculated with kernel K_x and the horizontal edge component is calculated with kernel K_y . $|K_x| + |K_y|$ gives an indication of the intensity of the gradient in the current pixel.

$K_x =$			$K_y =$		
-1	0	1	1	1	1
-1	0	1	0	0	0
-1	0	1	-1	-1	-1

Figure 2.8: Prewitt horizontal and vertical operators

Depending on the noise characteristics of the image or streaming video, edge detection results can vary. Gradient-based algorithms such as the Prewitt filter have a major drawback of being very sensitive to noise. The size of the kernel filter and coefficients are fixed and cannot be adapted to a given image. An adaptive edge-detection algorithm is necessary to provide a robust solution that is adaptable to the varying noise levels of these images to help distinguish valid image content from visual artifacts introduced by noise.

The Canny algorithm uses an optimal edge detector based on a set of criteria which include finding the most edges by minimizing the error rate, marking edges as closely as possible to the actual edges to maximize localization, and marking edges only once when a single edge exists for minimal response [5]. According to Canny, the optimal filter that meets all three criteria above can be efficiently approximated using the first derivative of a Gaussian function.

The first stage involves smoothing the image by convolving with a Gaussian filter. This is followed by finding the gradient of the image by feeding the smoothed image through a convolution operation with the derivative of the Gaussian in both the vertical and horizontal directions.

2.1.3 Histogram Method

In an image processing context, the histogram of an image normally refers to a histogram of the pixel intensity values. This histogram is a graph showing the number of pixels in an image at each different intensity value found in that image. For an 8-bit grayscale image there are 256 different possible intensities, and so the histogram will graphically display 256 numbers showing the distribution of pixels amongst those grayscale values. Histograms can also be taken of color images --- either individual histogram of red, green and blue channels can be taken, or a 3-D histogram can be produced, with the three axes representing the red, blue and green channels, and brightness at each point representing the pixel count. The exact output from the operation depends upon the implementation --- it may simply be a picture of the required histogram in a suitable image format, or it may be a data file of some sort representing the histogram statistics.

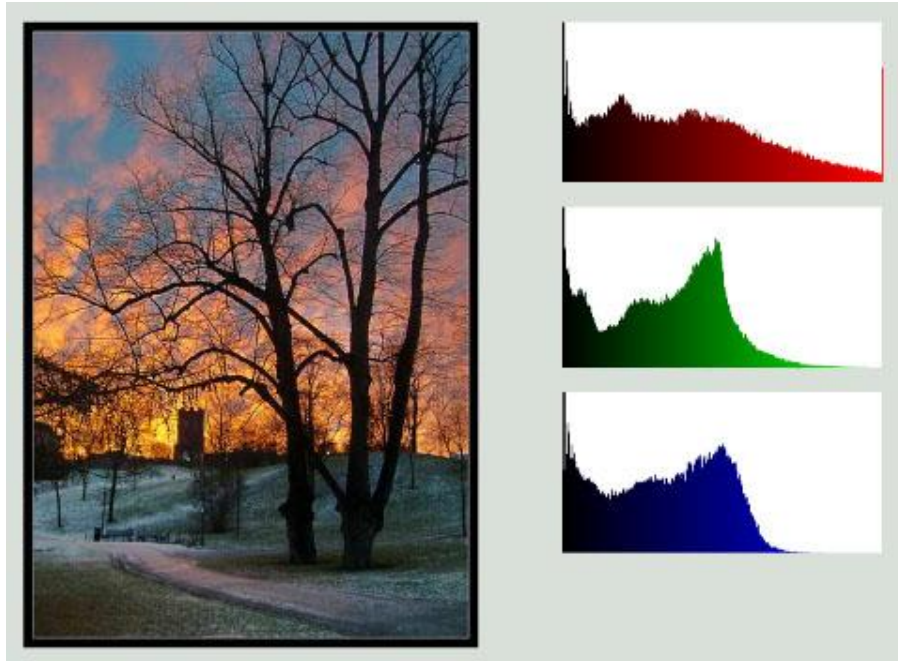


Figure 2.9: Example of image histogram

Digital image pixel tonality for 24 bit RGB color is expressed as a number between 0 and 255. 0 equals pure black and 255 equals pure white. The midpoint at about 127 would be the equivalent of middle gray in density. Digital image histograms are presented as a bar chart with the horizontal axis being the tonal range of your. The left side of the graph is 0 (zero) and the right side of the graph is 255. The vertical axis is the relative number of pixels at each of the 255 tonal values. The taller the hump in the graph, the more pixels resides at that particular tonal range.

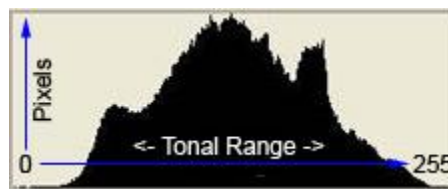


Figure 2.10: Example of histogram show pixel value

Histograms can be quite different looking depends to the content of the image. A histogram of a high key image with a majority of the content being very bright will produce a histogram that has most of the histogram graph located from the center to the right of center. A low key image with lots of dark and shadow areas will produce a histogram graph that is mostly center and left of center. A low key image with lots of dark and shadow areas will produce a histogram graph that is mostly center and left of center. There really isn't just one proper histogram for any given image. You can shift the tonal range (the histogram) around to lighten, darken or adjust the contrast in an image. To take advantage of the information supplied by an image's histogram you have to be able to visually interpret the image content, taking into consideration the location and approximate percentage of highlight, shadow and midtone pixels in the image itself.



Figure 2.11: Low key image with the majority of pixels to the left of center of the graph



Figure 2.12: High key images with the majority of the pixels to the right of center of the graph

2.2 RGB

For sciences communications, there are two main color spaces which are RGB and CMYK. RGB uses additive color mixing and is the basic color model used in television, computers and for web graphics or others medium that projects color with light.

The secondary colors of RGB divides to three parts which is cyan, magenta and yellow formed by mixing two primary color of RGB (red, green and blue). The combination of red and blue become magenta, combination of yellow and green become yellow, combination blue and green become cyan and all of the primary color in full intensity makes white. [3]

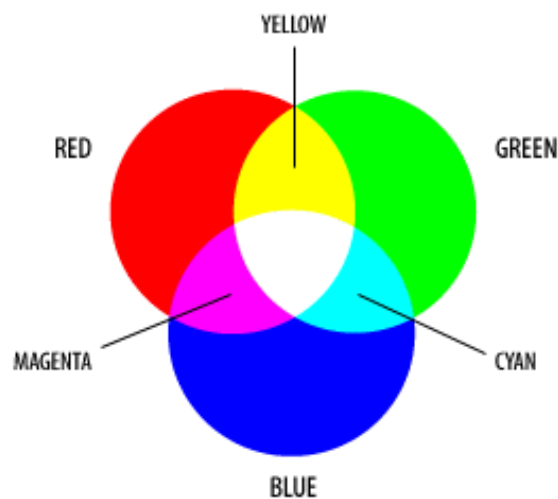


Figure 2.13: Result of combination primary color

2.3 MATLAB 2008

MATLAB is a high-level language and interactive environment that enables you to perform computationally intensive tasks faster than with traditional programming languages such as C, C++, and FORTRAN. This software gives us greatly function for the basic image processing and shape detection which have more application including signal processing, communications, control design, test and measurement, financial modeling and analysis and computational biology. Therefore, MATLAB can use for create our own algorithm and distribute it to other MATLAB users directly as MATLAB code.

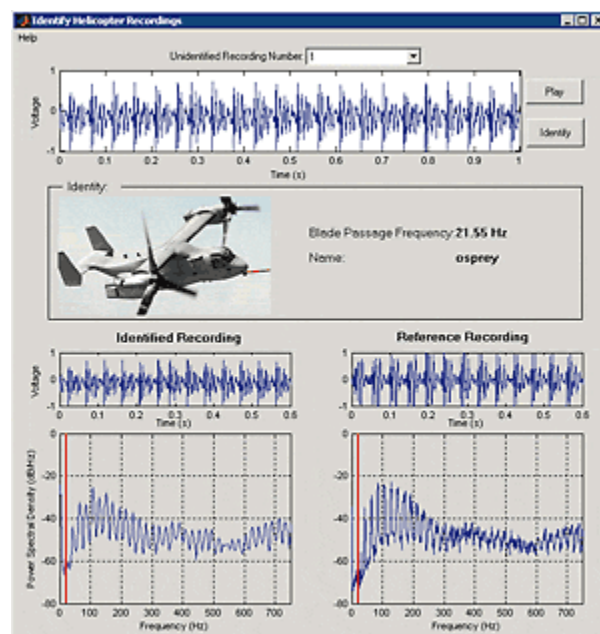


Figure 2.14: The application, developed in MATLAB, directly acquires signals from measurement hardware, performs analysis and plotting, and includes GUI controls.

2.4 Graphical User Interface (GUI)

2.4.1 GUI definition

Graphical user interface (GUI) is a type of user interface that takes advantage of the computer's graphics capabilities to make the program easier to use and also interface item that allows people to interact with programs in more ways the typing such as computers or others devices and household appliances and office commands with images rather than just have the text commands. A GUI give graphical icons and visual indicators to represent text based interfaces to fully present the information and actions available to a user.

Therefore, an icon is a small picture or symbol in a GUI that represents a program (or command), a file, a directory or a device (such as a hard disk or floppy). Icons are used both on the desktop and within application programs. Examples include small rectangles (to represent files), file folders (to represent directories), a trash can (to indicate a place to dispose of unwanted files and directories) and buttons on web browsers (for navigating to previous pages, for reloading the current page, etc.).

A major advantage of GUIs is that they make computer operation more intuitive, and thus easier to learn and use. Adding to this intuitiveness of operation is the fact that GUIs generally provide users with immediate, visual feedback about the effect of each action. In addition, GUIs allow users to take full advantage of the powerful multitasking (the ability for multiple programs and/or multiple instances of single programs to run simultaneously) capabilities of modern operating systems by allowing such multiple programs and/or instances to be displayed simultaneously.

2.4.2 MATLAB GUI

A graphic user interface (GUI) enables a user to perform interactive tasks which GUI is a graphical display that contains devices or components. To perform these tasks, user does not have to create a script or type commands at the command line.

The GUI elements are treated as objects such as pushbutton, checkbox, radio buttons, list boxes, slider (scroll bar) and popup menu just to name a few. They encapsulate data and method. In MATLAB, a GUI can also display in tabular form or as plot and can group related components.

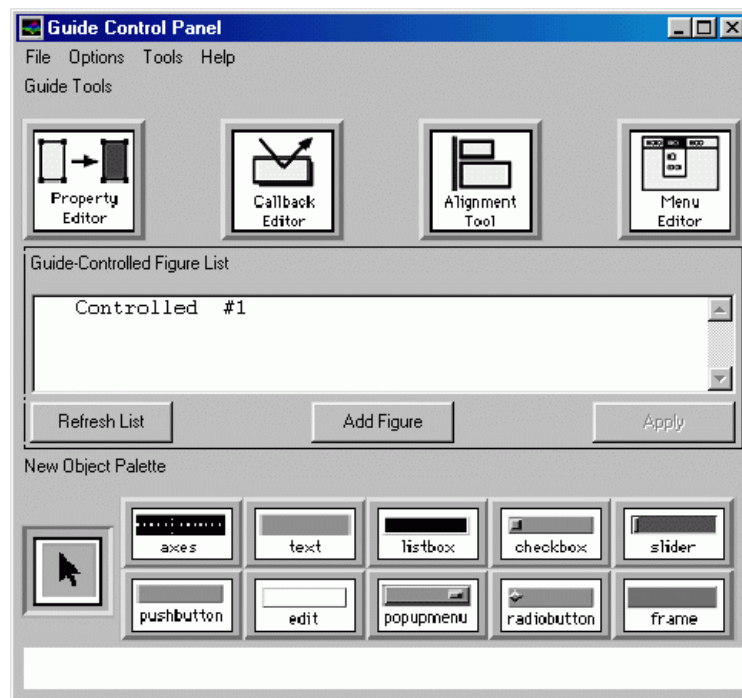


Figure 2.15: Example of GUI elements

These are the two basic tasks in process to implement the GUI:

- i) Laying out the GUI where MATLAB implements GUIs as figure windows containing various styles of uicontrol (user interface) objects.
- ii) Programming the GUI, where each object must be program to perform the intended action when activated by user of GUI.

2.4.3 MATLAB GUIDE

GUIDE, the MATLAB graphical user interface development environment provides a set of tools for creating graphical user interface (GUI). These tools simplify the process of lying out and programming GUIs. GUIDE primarily is a set of layout tools. However, GUIDE also generates an M-file that contains code to handle the initialization and launching of the GUI. This M-file provides a framework for the implementation of the *callbacks* -- the functions that execute when users activate components in the GUI.

2.4.4 GUI operation

The GUI is already associated with one or more user written routines known as callback. A callback is a function that you write and associate with a specific component in the GUI or with the GUI figure itself. The callbacks control GUI or component behavior by performing some action in response to an event for its component. The event can be a mouse click on a push

button, menu selection, key press, etc. This kind of programming is often called *event-driven* programming.

The callback functions you provide control how the GUI responds to events such as button clicks, slider movement, menu item selection, or the creation and deletion of components. There is a set of callbacks for each component and for the GUI figure itself.

The callback routines usually appear in a GUI code file following the initialization code and the creation of the components. When an event occurs for a component, MATLAB software invokes the component callback that is associated with that event. As an example, suppose a GUI has a push button that triggers the plotting of some data. When the user clicks the button, the software calls the callback you associated with clicking that button, and then the callback, which you have programmed, gets the data and plots it.

A component can be any control device such as an axes, push button, list box, or slider. For purposes of programming, it can also be a menu, toolbar tool, or a container such as a panel or button group.

Callbacks:

- i) Routine that execute whenever you activate the uicontrol object.
- ii) Define this routine as a string that is a valid MATLAB expression or the name of an M-file.
- iii) The expression executes in the MATLAB workspace.

Chapter 3

METHODOLOGY

3.1 Introduction

This chapter will present the whole methodology of this project and all the details of each part of software that use in this project. It will describe on how the project is organized and the flows of step by step in complete this project. The methodology is divided into two parts, which is developing the software using MATLAB. The other is developing the programming in offline time using MATLAB GUI.

The block diagram below shown that the basic information about this project which explain about the flow to carry out this project.

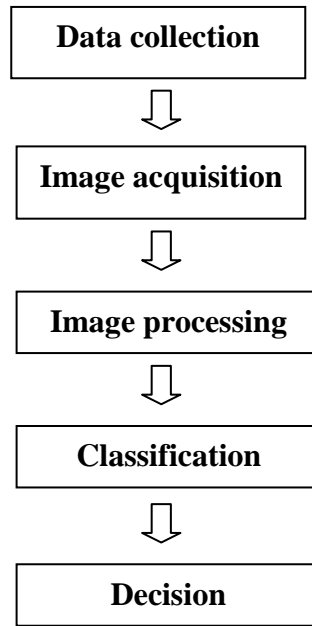


Figure 3.1: Block diagram for the flow of the project

- i) Collecting data which the images consist of different type of shape that is circle, square and rectangular and several color red, green and blue.
- ii) Identify some method that more suitable to use in the application of shape detection and color recognition like edge detection and histogram.
- iii) Image analysis by getting the result to identify the color or shape by development the module of that by using the method that decides.

3.2 Software Development

There are two main methods in order to develop this project. Before the project is developed using MATLAB, it is needed to do the study on MATLAB GUIDE. The flowchart in Figure 3.1 illustrates the sequence of steps for this project. The first method is designing the programming for two tasks that is shape detection and color recognition. Secondly, developing the GUI in MATLAB and programs every GUI component.

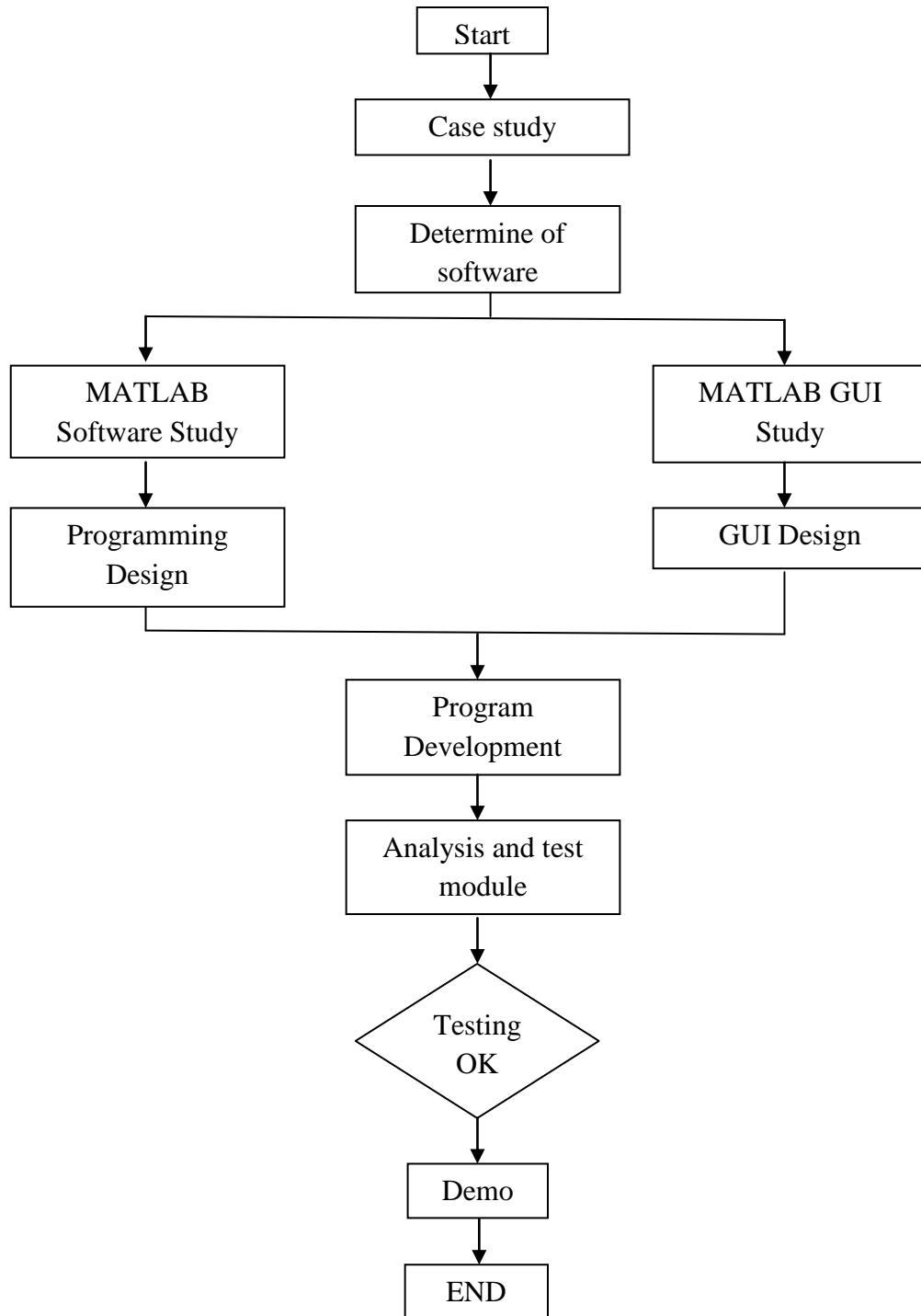


Figure 3.2: Flow Chart

3.2.1 Development MATLAB GUI using MATLAB GUIDE

The MATLAB graphical user interface (GUI) development environment provides a set of tools for creating graphical user interfaces (GUIs). These tools simplify the process of laying out and programming GUIs

There are 5 steps in building the MATLAB GUI. First, use a MATLAB tool called guide (GUI Development Environment) to layout the components that show in Figure 3.2. This tool allows a programmer to layout the GUI, selecting and aligning the GUI component to be placed in it. The basic component of the MATLAB GUI is shown in Table 3.1.

Table 3.1: Basic Matlab GUI component

Element	Created By	Description
Pushbutton	uicontrol	A graphical component that implements a pushbutton. It triggers a callback when clicked with a mouse
Toggle button	uicontrol	A graphical component that implements a toggle button. A toggle button is either on or off and its state changes each time that it is clicked. Each mouse button click also triggers a callback.
Text field	uicontrol	Creates a label which is a text string located at a point on the figure. Text fields never trigger callbacks.
Menu items	uimenu	Creates a menu item. Menu item triggers a callback when a mouse button is released over them
Context menus	uicontextmenu	Creates a context menu, which is a menu that appears over a graphical object when a user right-clicks the mouse on that object.

Axes	axes	Creates a new set of axes to display data on. Axes never trigger callbacks.
------	------	---

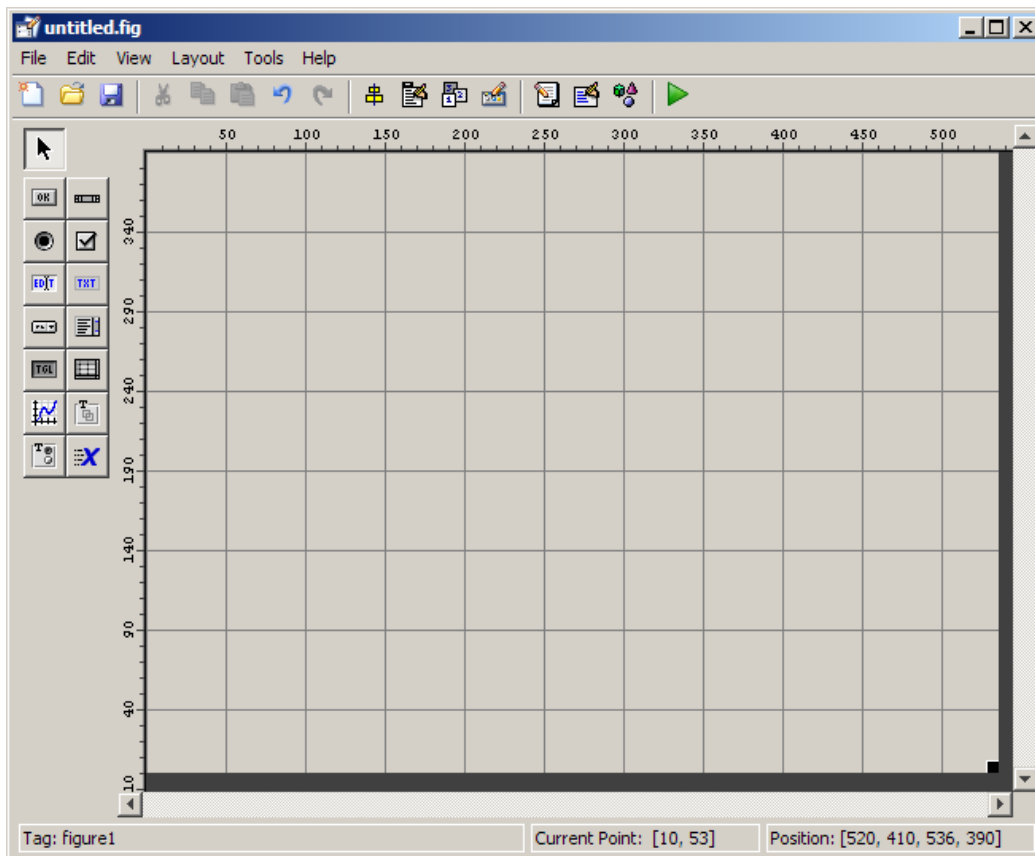


Figure 3.3: MATLAB GUIDE Layouts

Use a MATLAB tool called the Property Inspector (built into guide) to give each component a name (a “tag”) and to set the characteristics of each component, such as its color, the text it displays, the font size of the displays words and so on. After that, we save the figure to the file. When the figure is saved, two files will be created on disk with the same name but difference extents. The fig file contains the actual GUI that has been created and the M-file

contains the code to load the figure and skeleton call backs for each GUI element. These two file usually reside in the same directory. They correspond to the tasks of laying out and programming the GUI. When you lay out the GUI in the Layout Editor, your work is stored in the FIG-file. When you program the GUI, your work is stored in the corresponding M-file.

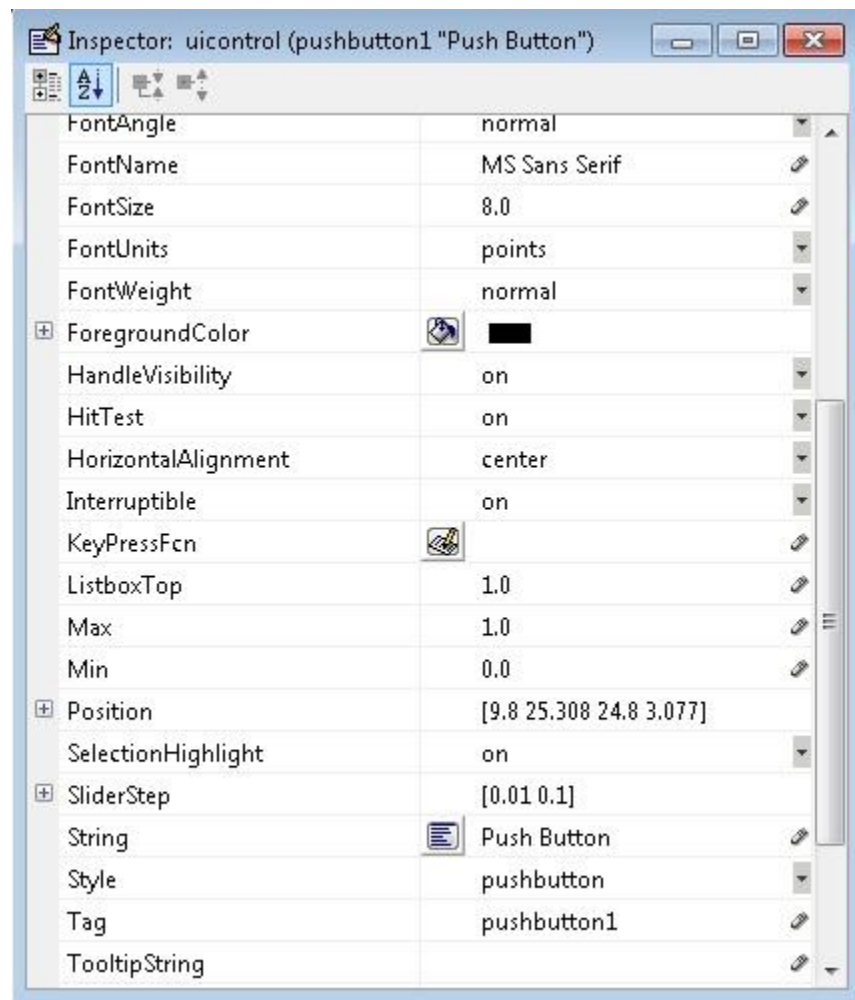


Figure 3.4: Property Inspector

After laying out the GUI component and set the property, the GUI will be look like in Figure 3.4 for example according to the user creativity.

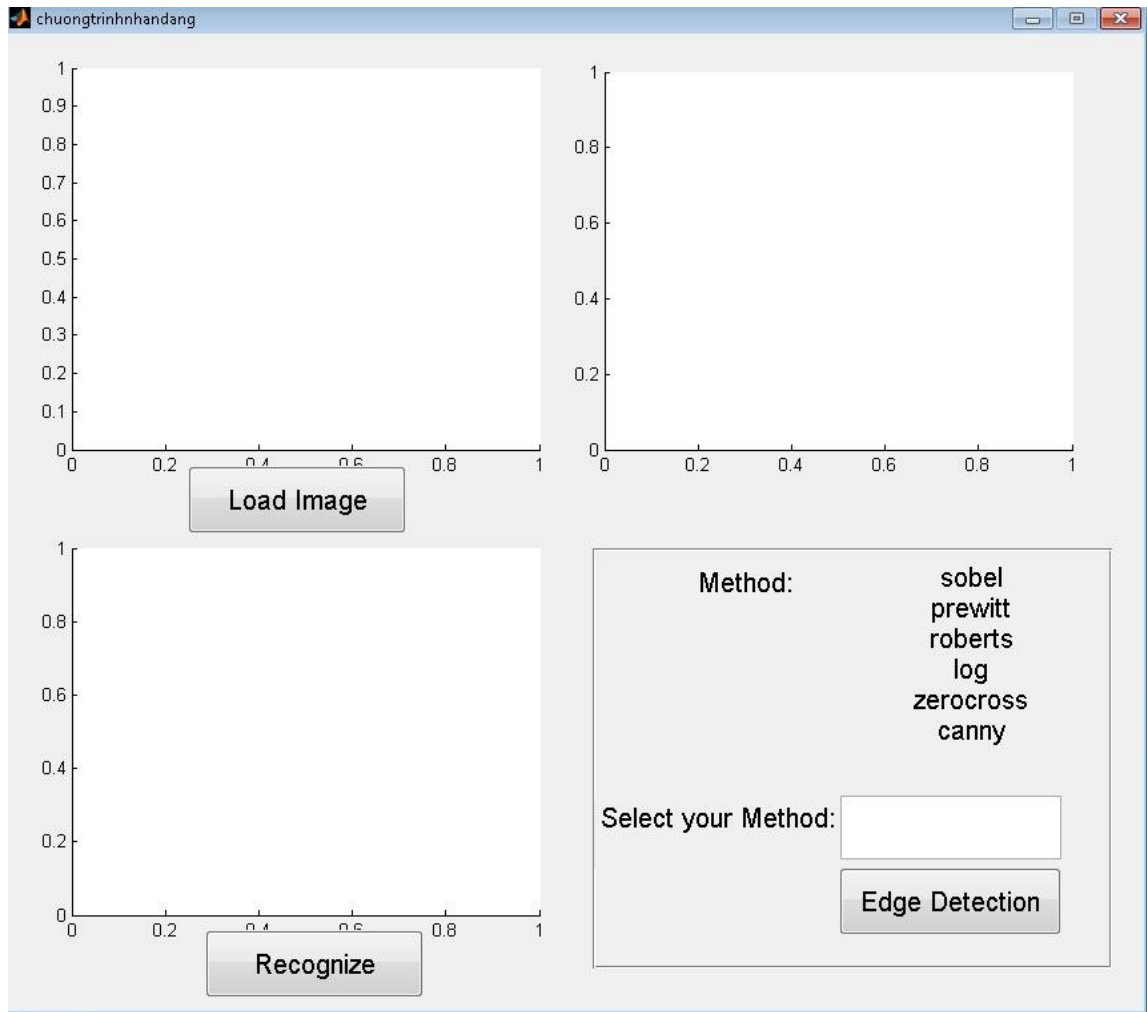


Figure 3.5: Example GUI

Finally, Figure 3.5 shown that the coding use to implement the behavior associated with each callback function in m-files. A callback is a function that writes and associates with the specifies GUI component or with GUI figure. It controls GUI by performing some action in response to an event for its component. Thus kind of programming is often called event-driven programming. The last step is the difficult part and has to make an extra reading on how to write the coding before the GUI component can perform some task that we need.

The image shows a MATLAB Editor window with the following code:

```

72 % Get default command line output from handles structure
73 varargout{1} = handles.output;
74
75
76 % --- Executes on button press in Load_Image.
77 function Load Image Callback(hObject, eventdata, handles)
78 [filename, pathname] = uigetfile({'*.bmp'; '*.jpg'; '*.gif'; '*.png'}, 'Pick an Image File');
79 S = imread([pathname,filename]);
80 axes(handles.axes1);
81 imshow(S);
82
83 handles.S = S;
84 guidata(hObject, handles);
85 % hObject    handle to Load Image (see GCBO)
86 % eventdata  reserved - to be defined in a future version of MATLAB
87 % handles    structure with handles and user data (see GUIDATA)
88 % --- Executes on button press in Edge_Detection.
89 function Edge Detection Callback(hObject, eventdata, handles)
90 b=get(handles.edit1, 'String');
91 S = handles.S;
92 axes(handles.axes2);
93 a=rgb2gray(S);
94 bw=edge(a,b);
95 bw = bwareaopen(bw,30);
96 se = strel('disk',2);
97 bw = imclose(bw,se);

```

The window title is "Editor - C:\Users\Jason\Documents\MATLAB\shape\shape\chuongtrinhnhandang\chuongtrinhnhandang.m". The stack is "Base". The file explorer on the right shows the file structure.

Figure 3.6: Example m-files for GUI

3.2.2 Build MATLAB Programming

After layout the GUI, it needs to program its behavior. The coding is writing to controls how the GUI responds to events such as push button, slider movement, menu item selection, axes or the creation and deletion components. This programming takes the form of set functions, called callbacks, for each of the components and for the GUI figure itself.

A callback is a function that writes and associates with a specific component in the GUI or with the GUI figure itself. The callbacks control GUI or component behavior by performing some action in response to an event for its component. The event can be a mouse click on a push button, menu selection, key press, etc. This kind of programming is often called event-driven programming.

The GUI figure and each type of component have specific kinds of callbacks with which you can associate it. The callbacks that are available for each component are defined as properties of that component. For example, a push button has five callback properties: `ButtonDownFcn`, `Callback`, `CreateFcn`, `DeleteFcn`, and `KeyPressFcn`. A panel has four callback properties: `ButtonDownFcn`, `CreateFcn`, `DeleteFcn`, and `ResizeFcn`. It are not required to, create a callback function for each of these properties. The GUI itself, which is a figure, also has certain kinds of callbacks with which it can be associated.

Each kind of callback has a triggering mechanism or event that causes it to be called. The following Table 3.2 lists the callback properties that are available, their triggering events, and the components to which they apply.

Table 3.2: Various kind of Callbacks

Callback Property	Triggering Event	Components
ButtonDownFcn	Executes when the user presses a mouse button while the pointer is on or within five pixels of a component or figure.	Axes, figure, button group, panel, user interface controls
Callback	Control action. Executes, for example, when a user clicks a push button or selects a menu item.	Context menu, menu user interface controls
CellEditCallback	Reports any edit made to a value in a table with editable cells; uses event data.	uitable
CellSelectionCallback	Reports indices of cells selected by mouse gesture in a table; uses event data.	uitable
ClickedCallback	Control action. Executes when the push tool or toggle tool is clicked. For the toggle tool, this is independent of its state.	Push tool, toggle tool
CloseRequestFcn	Executes when the figure closes.	Figure
CreateFcn	Initializes the component when it is created. It executes after the component or figure is created, but before it is displayed.	Axes, button group, context menu, figure, menu, panel, push tool, toggle tool, toolbar, user interface controls
DeleteFcn	Performs cleanup operations just before the component or	Axes, button group, context menu, figure, menu, panel,

	figure is destroyed.	push tool, toggle tool, toolbar, user interface controls
KeyPressFcn	Executes when the user presses a keyboard key and the callback's component or figure has focus.	Figure, user interface controls
KeyReleaseFcn	Executes when the user releases a keyboard key and the figure has focus.	Figure
OffCallback	Control action. Executes when the State of a toggle tool is changed to off.	Toggle tool
OnCallback	Control action. Executes when the State of a toggle tool is changed to on.	Toggle tool
ResizeFcn	Executes when a user resizes a panel, button group, or figure whose figure Resize property is set to On.	Figure, button group, panel
SelectionChangeFcn	Executes when a user selects a different radio button or toggle button in a button group component.	Button group
WindowButtonDownFcn	Executes when you press a mouse button while the pointer is in the figure window.	Figure
WindowButtonMotionFcn	Executes when you move the pointer within the figure window.	Figure

WindowButtonUpFcn	Executes when you release a mouse button.	Figure
WindowKeyPressFcn	Executes when you press a key when the figure or any of its child objects has focus.	Figure
WindowKeyReleaseFcn	Executes when you release a key when the figure or any of its child objects has focus.	Figure
WindowScrollWheelFcn	Executes when the mouse wheel is scrolled while the figure has focus.	Figure

The GUI M-files that GUIDE generates is a function file. The name of the main function is the same as the name of the M-files. Each callback in the file is a sub function of the main function. When GUIDE generates as M-files, it automatically includes templates for the most commonly used callbacks for each component. The major sections of the GUI M-file are shown in Table 3.3 below.

Table 3.3: Major Sections of the GUI M-file

Section	Description
Comments	Displayed at the command line in response to the help command. Edit these as necessary for GUI
Initialization	GUIDE initialization tasks. Do not edit this code
Opening function	Perform the initialization tasks before the user has access to the GUI
Output function	Returns outputs to the MATLAB command line after the opening function returns control and before control returns to the command lines.
Component and figure callbacks	Control the behavior of the GUI figures and of individual components. MATLAB calls a callback in response to a particular event for a component or for the figure itself.
Utility/helper functions	Performs miscellaneous functions not directly associated with an event for the figure or a component.

GUIDE automatically includes two callbacks, the opening function and the output function, in every GUI M-file it creates.

Chapter 4

RESULTS AND DISCUSSIONS

4.1 Introduction

The methodology had come out with some result in order to complete the development of shape detection and color recognitions module. These are the result of step by step methods done in the methodology development during the training and execution phase within its problems.

i) Training phase

Training phase will produce a development of shape detection and color recognitions coding which is in the m-file format. The purpose of produce the m-file in MATLAB is to avoid error in coding and determine the accurate coding for the two applications above. The most important part is the result had given the decision on screen.

ii) Execution phase

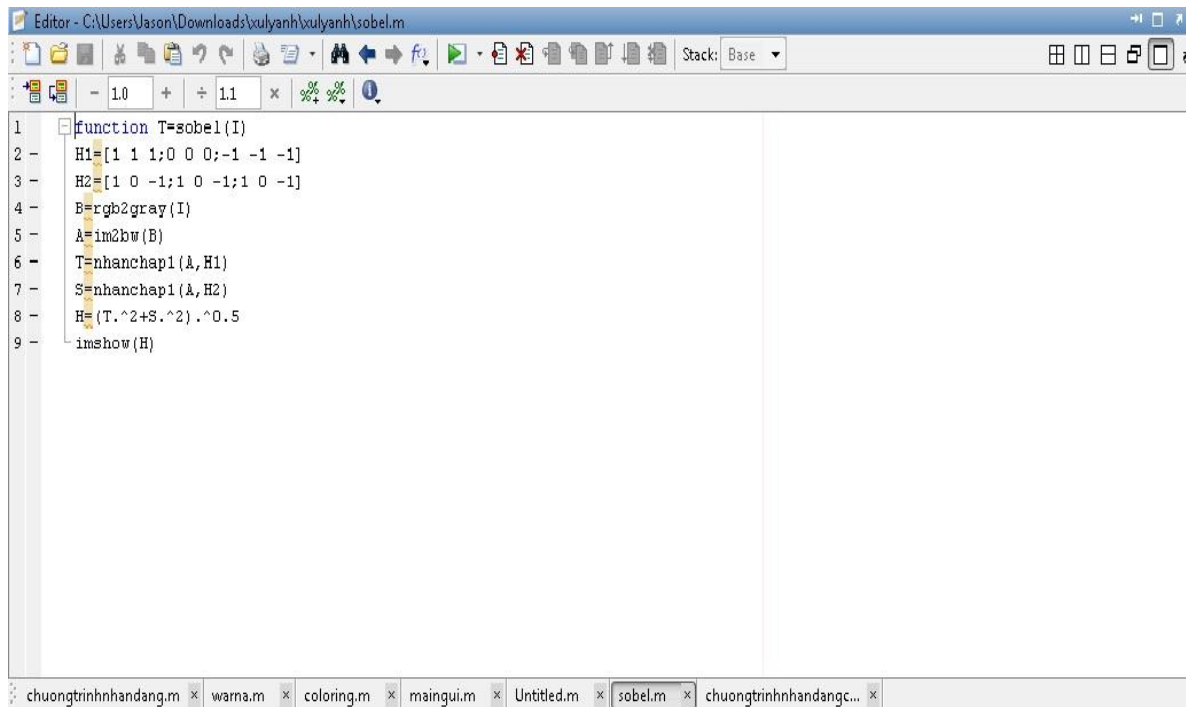
Execution phase is the process to convert the coding in m-file MATLAB to the MATLAB GUI application for the two application of image processing which is the shape detection and color recognitions. The simple module for the trainers is developed by using MATLAB GUI.

4.2 Training phase result

In the training phase, there are divided into two parts according to the two applications that need in this project. There are few methods taking part and the result of the methods can be explained as result below. The problem of some steps during the training phase is described.

4.2.1 Result of shape detection development

For the development of shape detection coding, Sobel, Prewitt, and Roberts methods and also Laplacian of Gaussian, zero-cross and canny methods was used. Figure below show that the coding of sobel method completely design for the first step. Sobel is one of the edge-detection methods that use for the shape detection which it's finds edges using the Sobel approximation to the derivative. It returns edges at those points where the gradient of I is maximum.



```

1 function T=sobel(I)
2     H1=[1 1 1;0 0 0;-1 -1 -1]
3     H2=[1 0 -1;1 0 -1;1 0 -1]
4     B=rgb2gray(I)
5     A=im2bw(B)
6     T=nhanchap1(A,H1)
7     S=nhanchap1(A,H2)
8     H=(T.^2+S.^2).^0.5
9     imshow(H)

```

Figure 4.1: Coding for sobel method

After complete design the coding for the each methods that will be used in the application, its need to design the coding for the system can be detect the image that we load from the data to give the decision by calculate the centroid, area and perimeter of the image which the result is circle, square or reactangle.

```

12 - L = bwlabel(bw);
13 - s = regionprops(L, 'centroid');
14 - dt = regionprops(L, 'area');
15 - cv = regionprops(L, 'perimeter');
16 - dim = size(s)
17 - BW_filled = imfill(bw, 'holes');
18 - boundaries = bwboundaries(BW_filled);
19 - imshow(bw);
20 - figure; imshow(i);
21 - hold on;
22 - for k=1:dim(1)
23 -     b= boundaries(k);
24 -     dim = size(b)
25 -     for i=1:dim(1)
26 -         khoangcach(k)(1,i) = sqrt ( ( b(i,2) - s(k).Centroid(1) )^2 + ( b(i,1) - s(k).Centroid(2) )^2 )
27 -     end
28 -     a=max(khoangcach(k));
29 -     b=min(khoangcach(k));
30 -     c=dt(k).Area;
31 -     dolech=a-b;
32 -     vuong = c/ (4*b^2)
33 -     chunhat=c/ (4*b*(a^2-b^2)^0.5);
34 -     tanggiacdeu=(c*3^0.5)/((a+b)^2);
35 -     elip =c/ (a*b*pi);
36 -     thoi= (c*( a^2 - b^2 )^0.5) / (2*a^2*b)
37 -     if dolech < 10
38 -         text(s(k).Centroid(1)-20,s(k).Centroid(2),'circle')
39 -     elseif (vuong < 1.05 ) & (vuong > 0.95 )
40 -         text(s(k).Centroid(1)-20,s(k).Centroid(2),'square')
41 -     elseif (elip < 1.05 ) & (elip > 0.95 )
42 -         text(s(k).Centroid(1)-20,s(k).Centroid(2),'ellipse')

```

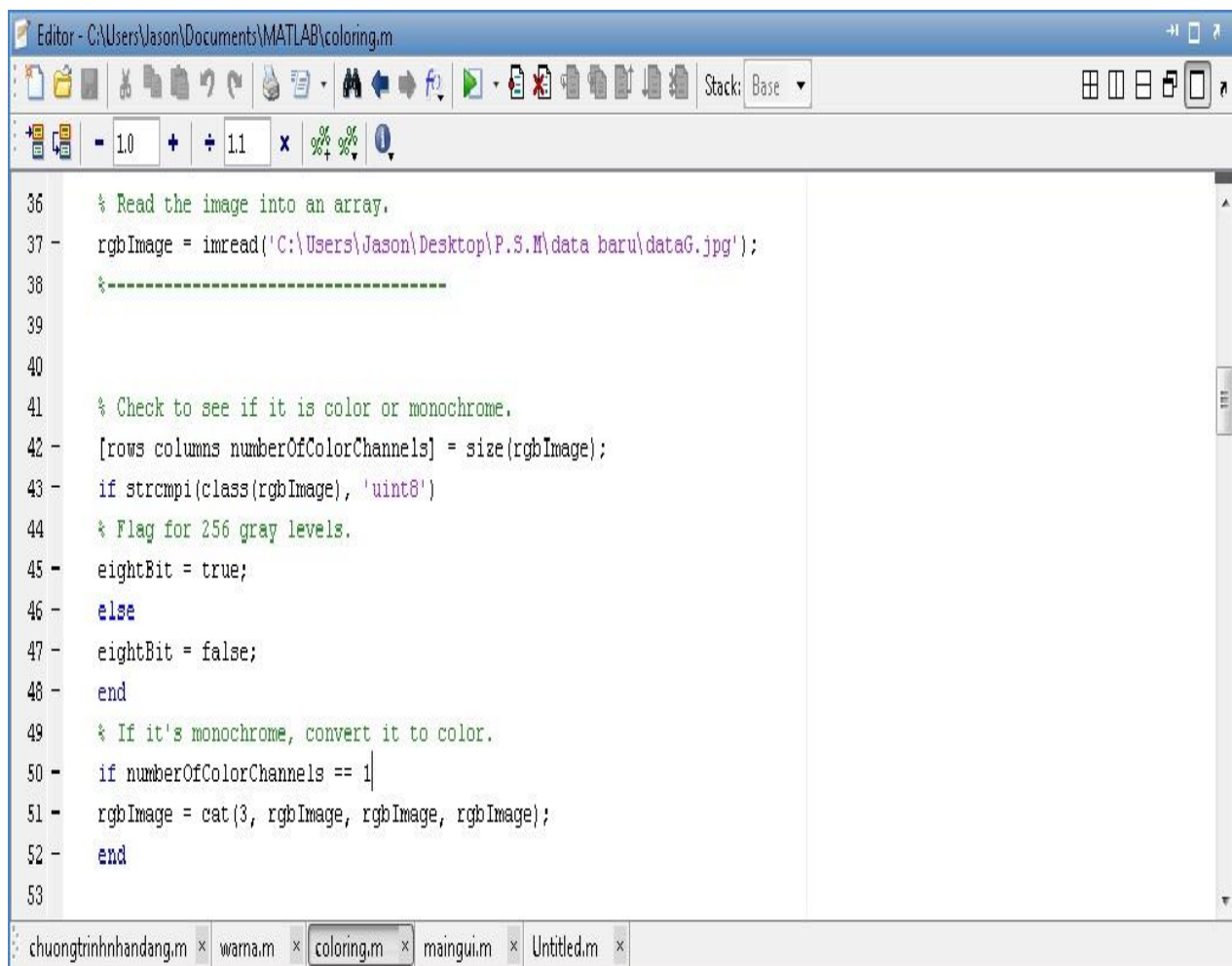
chuongtrinhnhandangcachinh.m x warna.m x coloring.m x maingui.m x Untitled.m x sobel.m x chuongtrinhnhandangcachinh.m x

Start script Ln 1 Col 1

Figure 4.2: Coding shown decision of detect image

4.2.2 Result of color recognition development

For the development of color recognition, the method that used is histogram. For the beginning to design the coding the system must call the imaged from the data and check the image is color or monochrome so if the image is monochrome then it will be convert to color.



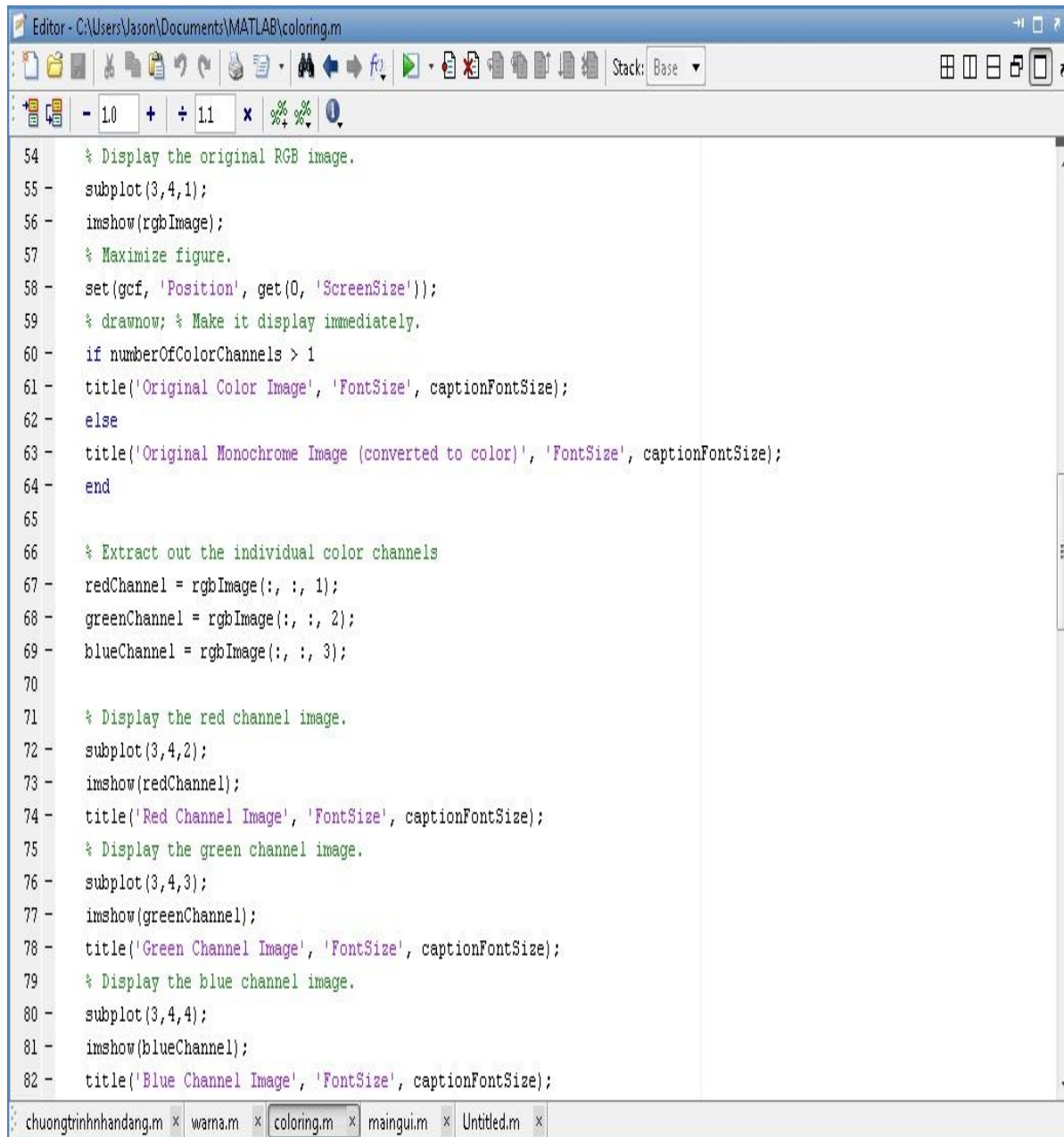
```

Editor - C:\Users\Jason\Documents\MATLAB\coloring.m
36 % Read the image into an array.
37 - rgbImage = imread('C:\Users\Jason\Desktop\P.S.M\data baru\dataG.jpg');
38 %-----
39
40
41 % Check to see if it is color or monochrome.
42 - [rows columns numberOfColorChannels] = size(rgbImage);
43 - if strcmpi(class(rgbImage), 'uint8')
44 % Flag for 256 gray levels.
45 - eightBit = true;
46 - else
47 - eightBit = false;
48 - end
49 % If it's monochrome, convert it to color.
50 - if numberOfColorChannels == 1
51 - rgbImage = cat(3, rgbImage, rgbImage, rgbImage);
52 - end
53
chuongtrinhnhandang.m x warna.m x coloring.m x maingui.m x Untitled.m x

```

Figure 4.3: Coding of read and check the image

Then, display the original RGB image as the reference for the user which image have been choose for this application. After that, it will be extract out the color image to the individual color channels which in Red, Green and blue channel image.



```

54 % Display the original RGB image.
55 - subplot(3,4,1);
56 - imshow(rgbImage);
57 % Maximize figure.
58 - set(gcf, 'Position', get(0, 'ScreenSize'));
59 % drawnow; % Make it display immediately.
60 - if numberOfColorChannels > 1
61 - title('Original Color Image', 'FontSize', captionFontSize);
62 - else
63 - title('Original Monochrome Image (converted to color)', 'FontSize', captionFontSize);
64 - end
65
66 % Extract out the individual color channels
67 - redChannel = rgbImage(:, :, 1);
68 - greenChannel = rgbImage(:, :, 2);
69 - blueChannel = rgbImage(:, :, 3);
70
71 % Display the red channel image.
72 - subplot(3,4,2);
73 - imshow(redChannel);
74 - title('Red Channel Image', 'FontSize', captionFontSize);
75 % Display the green channel image.
76 - subplot(3,4,3);
77 - imshow(greenChannel);
78 - title('Green Channel Image', 'FontSize', captionFontSize);
79 % Display the blue channel image.
80 - subplot(3,4,4);
81 - imshow(blueChannel);
82 - title('Blue Channel Image', 'FontSize', captionFontSize);

```

Figure 4.4: Coding of extract out the color image to the individual color channels

To recognize the image we used histogram method which calculates the pixel value of each RGB color and then plot in the graph separately according to the RGB pixel value and plot all into one graph to compare the pixel value. By comparing the pixel value, which color pixel value is highest then that is the result color of the image.

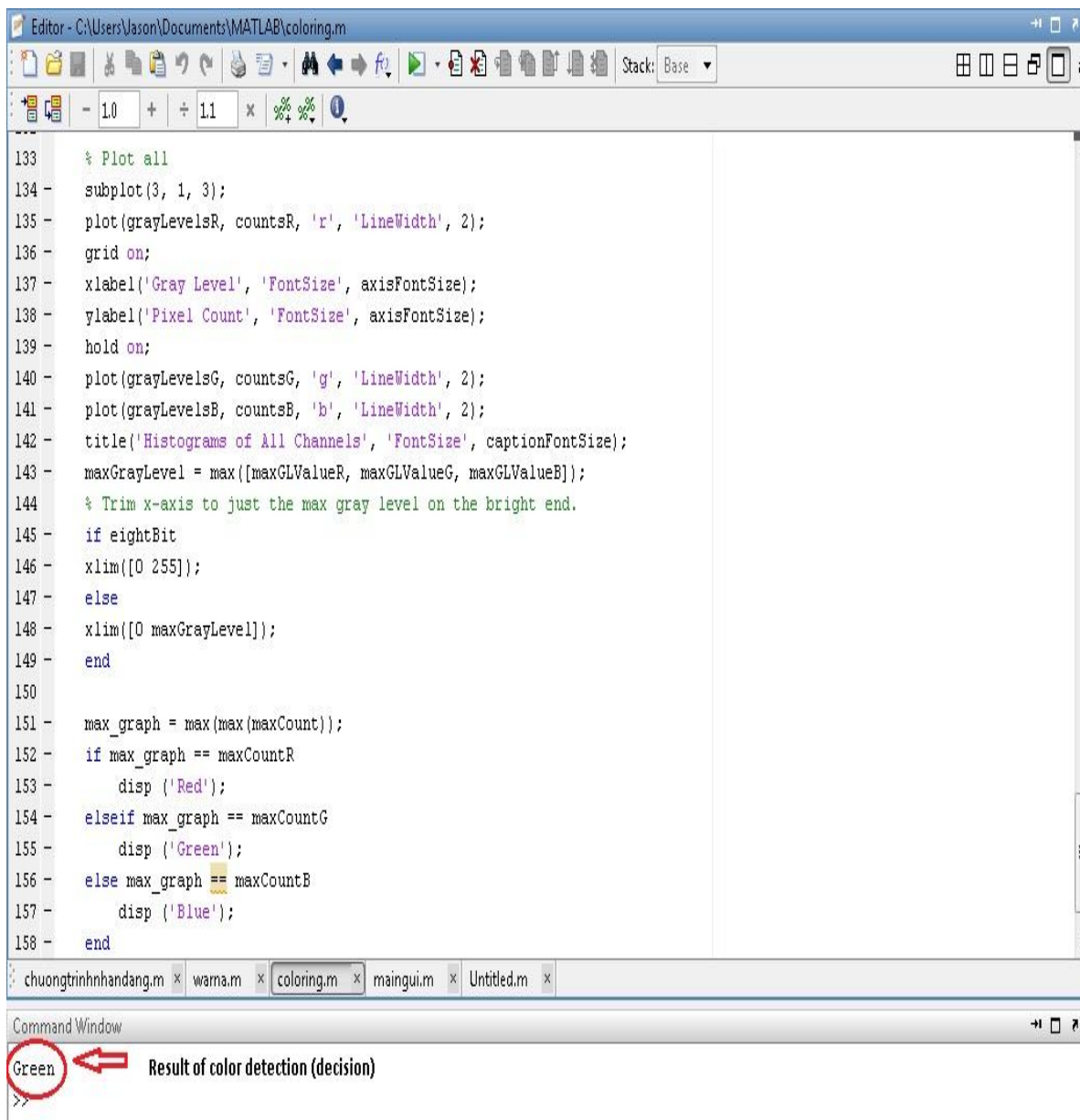


Figure 4.5: Coding of after calculate the pixel value and plot in a graph

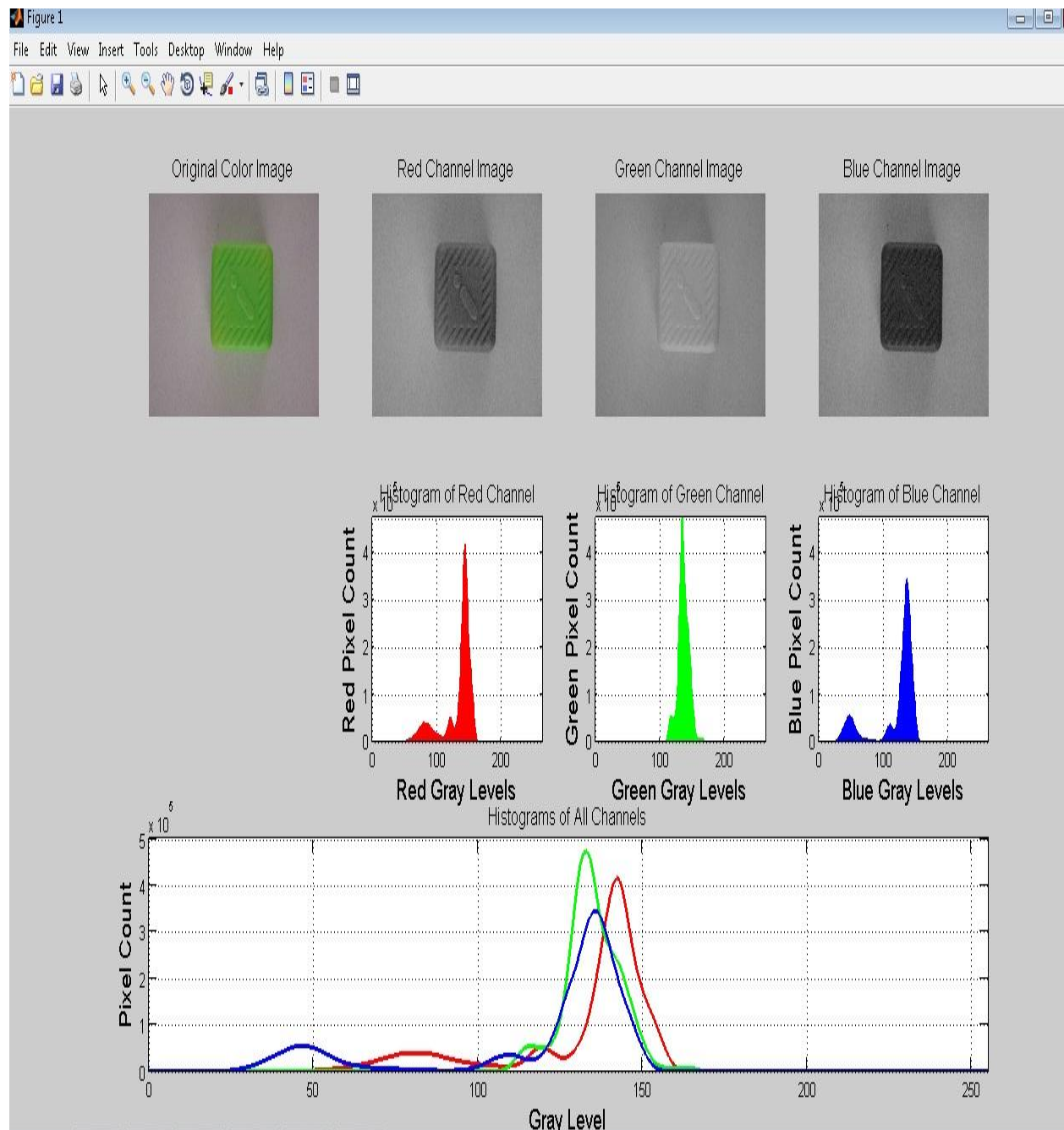


Figure 4.6: Final result

4.3 Execution phase result

The main menu of execution phase is the GUI in this project which contains two main buttons which link to the two application selection which the user of this module can choose one of the applications there. The main menu of the GUI and the functions of the applications are shown in Figure 4.7.

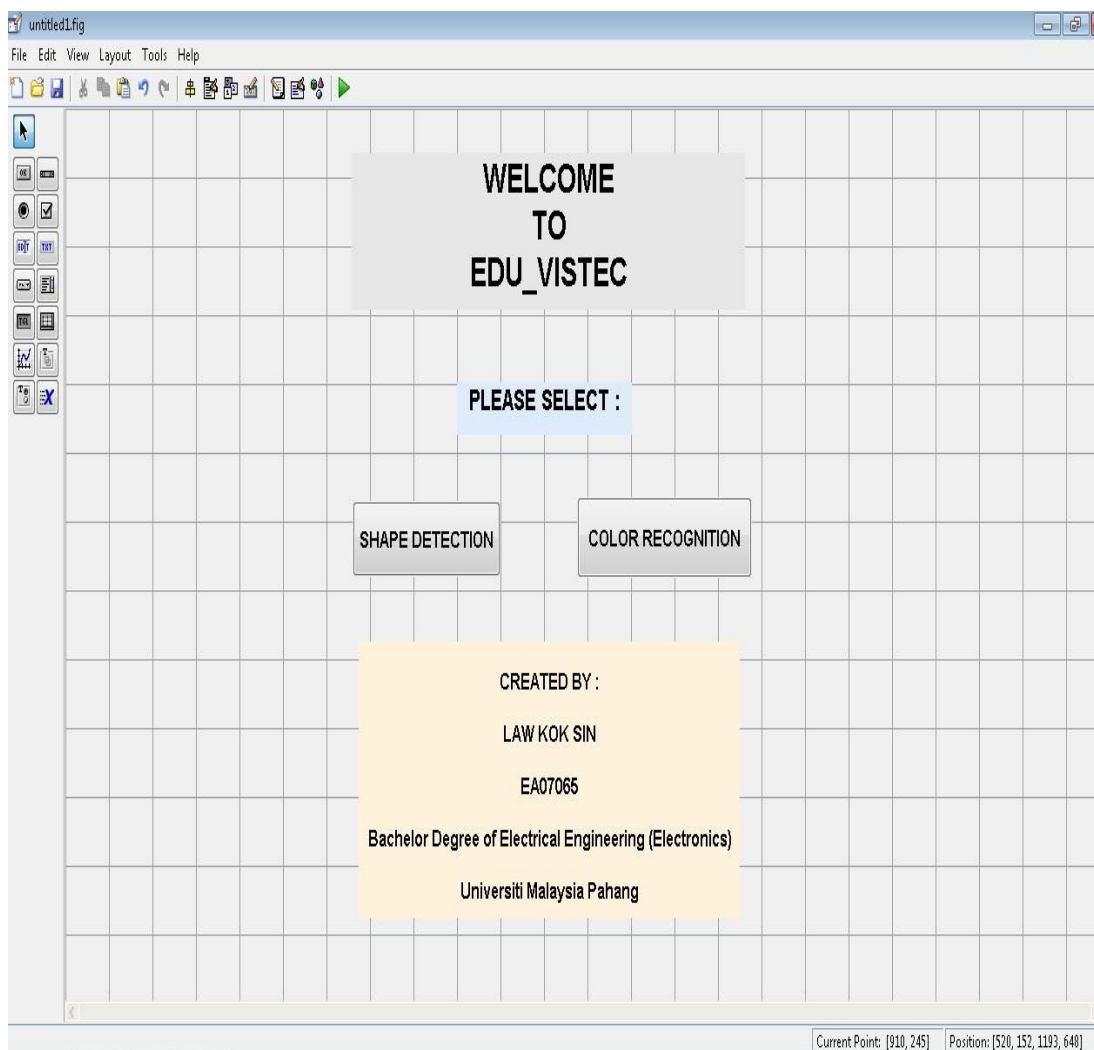


Figure 4.7: Main Menu GUI

When the user select the first button which is the shape detection, then there will pop-up the GUI application for that item. From there the user can start to learn the process of shape detection start from load image, select edge-detection methods and then analysis the image and finally get the result.

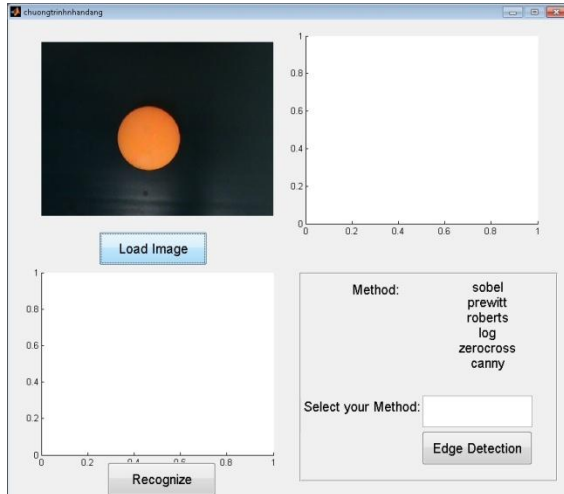


Figure 4.8: Load image

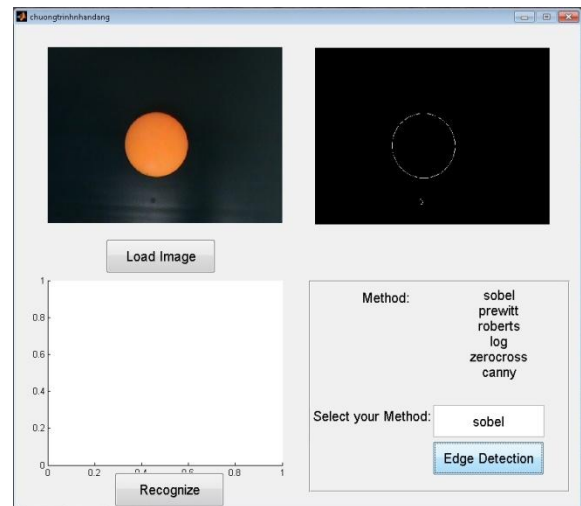


Figure 4.9: Select edge-detection method

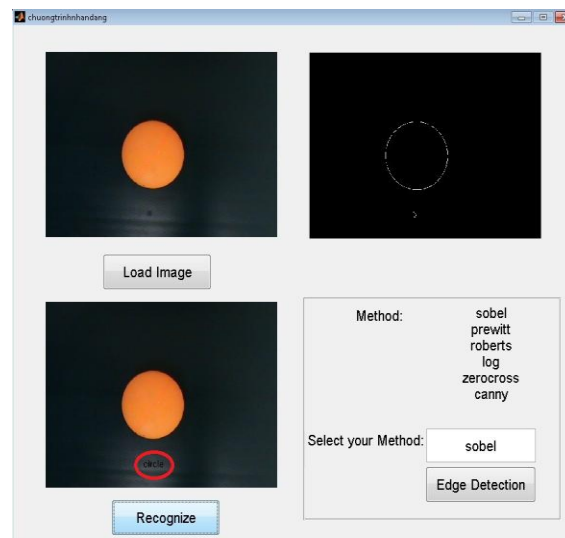


Figure 4.10: Recognize and get result

Besides, the user for this module can also select for the second button which contain the color recognition. The pop-up GUI will show to the user the explanation for the whole process to recognize the color by using calculating the histogram and the maximum of pixel value means that is the answer. The result will show in command window.

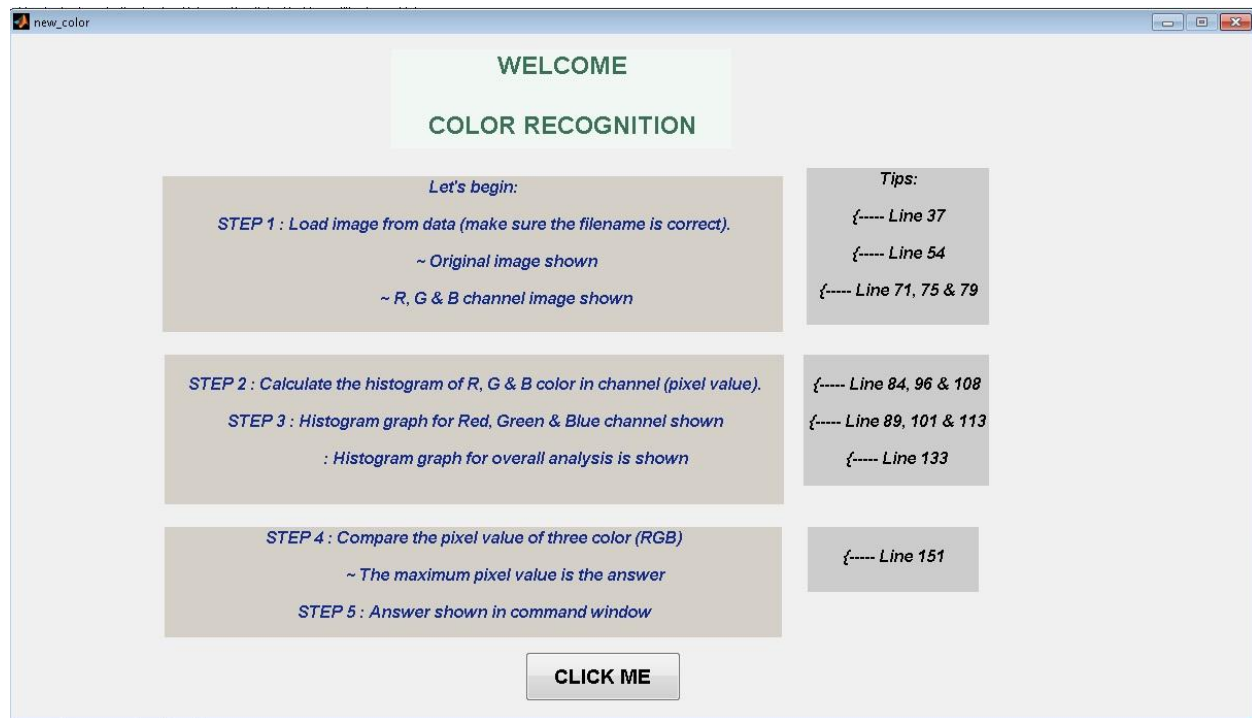


Figure 4.11: GUI for the color recognition (Part I)

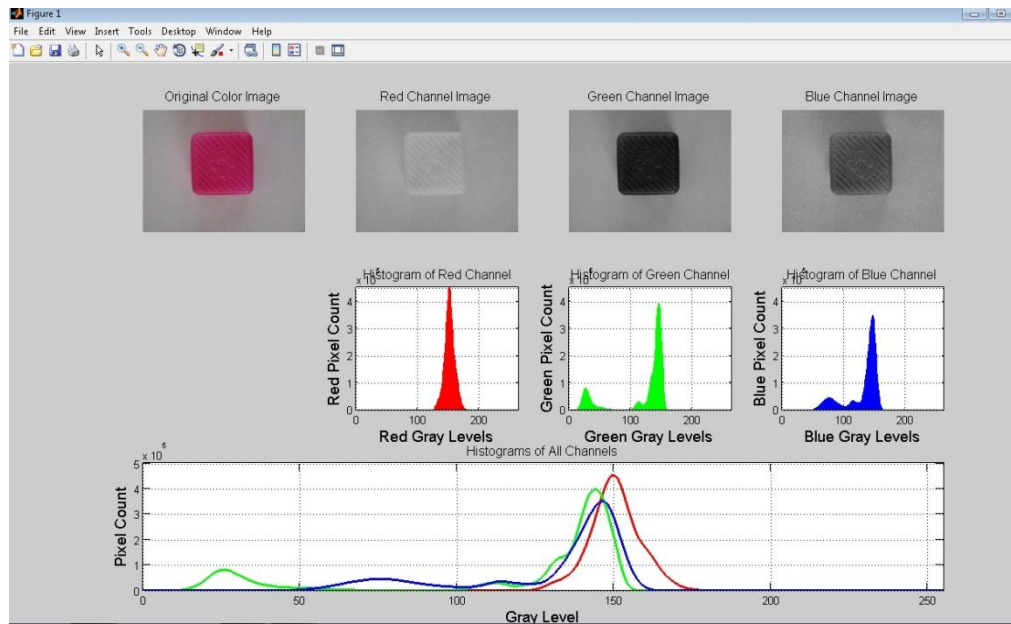


Figure 4.12: GUI for the color recognition (Part II)

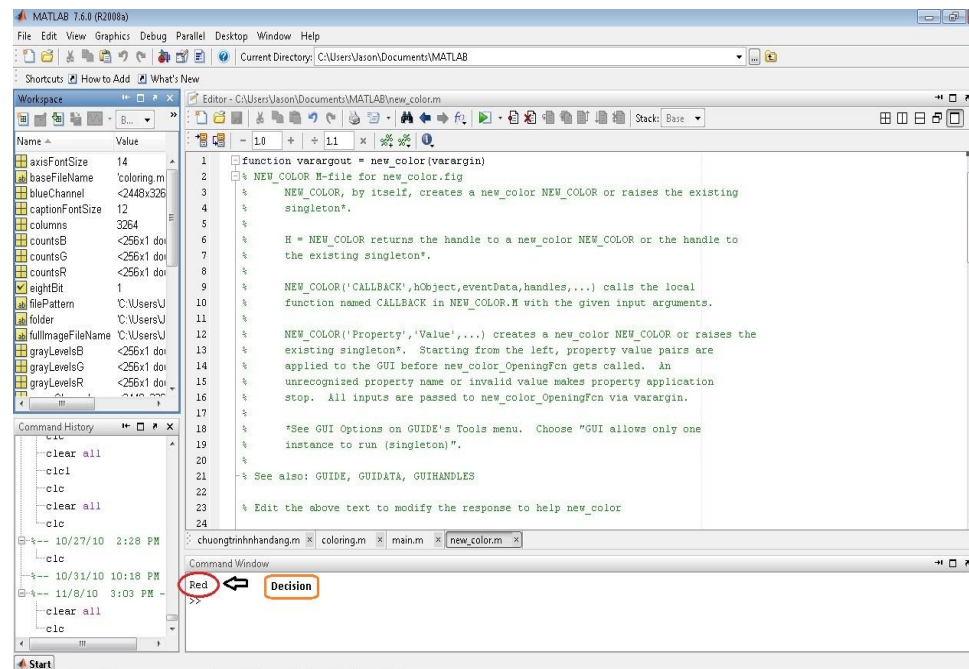


Figure 4.13: GUI for the color recognition (Part III)

4.4 Discussion

The MATLAB software function implemented is successfully developed the image processing application module. However, there are few aspects that need to be improved. When user need to load the image, the image use as the data must be very clear because of the method that used for the application is quite sensitive. So make sure when the collecting data process the image must try to avoid had shadow because shadow of image can affect the final result of the analysis.

Nevertheless, this problem can be overcome by not taking image under the lighting and also take the image frequently to get as much as possible for the data so that at the end of the project can get the accurately result.

Besides, the MATLAB software also can be categorized as sensitive software in the part of coding and also coding name. Be careful when save file make sure that the filename will not a number but use any word.

Chapter 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

The GUI design and its implementation has fully developed to achieve the entire objectives. The development of image processing application which covers about the shape detection and color recognition using MATLAB GUI in MATLAB GUIDE was done after detailed study and analysis of the methods of those applications. Through the development of this project, it has been concluded that the MATLAB GUI can be used for many types of applications for image processing but not just for these simple applications.

The main objectives of this project are to create a module about the application of image processing which uses the MATLAB software as the main tool and use the application Graphics User Interface (GUI) skill is fully achieved. The benefits for the student which complete this project is which can more understand about the method and technique that can be used for image processing by studying further and analyzing it.

5.2 Future Recommendation

For the future recommendation, to improve this project, others technique or added more technique to the image processing application so that the user for the module can learn more than the current module. To make it more advance maybe this module use for the real time which the image not load from the file but taken directly from the webcam or camera.

For others recommendation, to learn more about the software that can be used for the image processing application these kind of project can try to replace MATLAB tool with others like Open CV and Visual Basic for the same purpose. The color recognition just only for RGB basic color can add to HSV color. Same to the shape detection can add on more difference type of shape like diamond or triangle.

REFERENCES

- [1] J.P. Wang, Y.T. Lin, Y.S. Tsai, Journal of Materials Processing Technology 68 (1997) 246 -250
- [2] Jianping Fan, Walid G. Aref, Mohand-Said Hacid, Ahmed K.Elmagarmid, Pattern Recognition Letters 22 (2001) 1419 – 1429
- [3] G.D Finlayson, G.Y Tian, Color normalization for the color recognition, Internat J. Pattern Recog. ArtiDcial Intell. 13(8) (1999) 1271-1285
- [4] http://www.manifold.net/doc/image_threshold.htm
- [5] Canny, J., “A Computational Approach to Edge Detection”, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, November 1986
- [6] <http://www.mathworks.com/help/toolbox/vipblks/ref/edgedetection.html>

APPENDIX A

```

function varargout = chuongtrinhnhandang(varargin)
% CHUONGTRINHNHANDANG M-file for chuongtrinhnhandang.fig
%   CHUONGTRINHNHANDANG, by itself, creates a new CHUONGTRINHNHANDANG
or raises the existing
%   singleton*.
%
%   H = CHUONGTRINHNHANDANG returns the handle to a new
CHUONGTRINHNHANDANG or the handle to
%   the existing singleton*.
%
%   CHUONGTRINHNHANDANG('CALLBACK',hObject,eventData,handles,...) calls the
local
%   function named CALLBACK in CHUONGTRINHNHANDANG.M with the given input
arguments.
%
%   CHUONGTRINHNHANDANG('Property','Value',...) creates a new
CHUONGTRINHNHANDANG or raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before chuongtrinhnhandang_OpeningFunction gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to chuongtrinhnhandang_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 chuongtrinhnhandang

% Last Modified by GUIDE v2.5 30-Sep-2010 15:52:28

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

```

```

end

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

% --- Executes just before chuongtrinhnhandang is made visible.
function chuongtrinhnhandang_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 chuongtrinhnhandang (see VARARGIN)

% Choose default command line output for chuongtrinhnhandang
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes chuongtrinhnhandang wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = chuongtrinhnhandang_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)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in Load_Image.
function Load_Image_Callback(hObject, eventdata, handles)
[filename, pathname] = uigetfile({'*.bmp'; '*.jpg'; '*.gif'; '*.*'}, 'Pick an Image File');
S = imread([pathname, filename]);
axes(handles.axes1);
imshow(S);

```

```

handles.S = S;
guidata(hObject, handles);
% hObject    handle to Load_Image (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 Edge_Detection.
function Edge_Detection_Callback(hObject, eventdata, handles)
b=get(handles.edit1,'String');
S = handles.S;
axes(handles.axes2);
a=rgb2gray(S);
bw=edge(a,b);
bw = bwareaopen(bw,30);
se = strel('disk',2);
bw = imclose(bw,se);
bw = imfill(bw,'holes');
imshow(bw);
handles.bw = bw;

% hObject    handle to Edge_Detection (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 Recognize.
function Recognize_Callback(hObject, eventdata, handles)
global i;
b=get(handles.edit1,'String');
S = handles.S;
axes(handles.axes3);
format long;
a=rgb2gray(S);
bw=edge(a,b);
bw = bwareaopen(bw,30);
se = strel('disk',2);
bw = imclose(bw,se);
bw = imfill(bw,'holes');
L = bwlabel(bw);
s = regionprops(L, 'centroid');
dt = regionprops(L, 'area');
dim = size(s);
BW_filled = imfill(bw,'holes');
boundaries = bwboundaries(BW_filled);
imshow(S);
hold on;
for k=1:dim(1)
    b= boundaries{k};
    dim = size(b);

```

```

for i=1:dim(1)
    khoangcach{k}(1,i) = sqrt ( ( b(i,2) - s(k).Centroid(1) )^2 + ( b(i,1) - s(k).Centroid(2) )^2 );
end
a=max(khoangcach{k});
b=min(khoangcach{k});
c=dt(k).Area;
dolech=a-b;
vuong = c/(4*b^2);
chunhat=c/(4*b*(a^2-b^2)^0.5);
tamgiacdeu=(c*3^0.5)/((a+b)^2);
elip =c/(a*b*pi);
thoi= (c*( a^2 - b^2 )^0.5) / (2*a^2*b);
if dolech < 10
    text(s(k).Centroid(1)-20,s(k).Centroid(2),'circle');
elseif (vuong < 1.05 ) & (vuong > 0.95 )
    text(s(k).Centroid(1)-20,s(k).Centroid(2),'square');
elseif (elip < 1.05 ) & (elip > 0.95 )
    text(s(k).Centroid(1)-20,s(k).Centroid(2),'ellipse');
elseif (thoi < 1.05 ) & (thoi > 0.95 )
    text(s(k).Centroid(1)-20,s(k).Centroid(2),'diamond');
elseif ((chunhat <1.05) & (chunhat >0.95))
    text(s(k).Centroid(1)-20,s(k).Centroid(2),'rectangle');
elseif (tamgiacdeu < 1.05 ) & (tamgiacdeu > 0.95 )
    text(s(k).Centroid(1)-20,s(k).Centroid(2),'triangle');
end
end

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

function edit1_Callback(hObject, eventdata, handles)
% hObject   handle to edit1 (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 edit1 as text
%         str2double(get(hObject,'String')) returns contents of edit1 as a double
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject   handle to edit1 (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.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

```

```
    set(hObject,'BackgroundColor','white');  
end
```

```
% --- Executes during object creation, after setting all properties.  
function axes1_CreateFcn(hObject, eventdata, handles)  
% hObject    handle to axes1 (see GCBO)  
% eventdata  reserved - to be defined in a future version of MATLAB  
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: place code in OpeningFcn to populate axes1
```

```
% --- Executes on button press in pushbutton6.  
function pushbutton6_Callback(hObject, eventdata, handles)  
% hObject    handle to pushbutton6 (see GCBO)  
% eventdata  reserved - to be defined in a future version of MATLAB  
% handles    structure with handles and user data (see GUIDATA)
```


APPENDIX B

```
%-----
% Calculates and displays histograms of a color image.
% Each color channel (red, green, blue) is extracted and its histogram calculated and displayed.
% Written by Image Analyst, Jan. 2010.

clc;
clear;
close all;
workspace;

% Change the current folder to the folder of this m-file.
% (The line of code below is from Brett Shoelson of The Mathworks.)
if(~isdeployed)
cd(fileparts(which(mfilename)));
end

% Set up sizes for the captions and text we will show on the figure.
captionFontSize = 12; % Font size for headers/captions of images.
axisFontSize = 14; % Font size for axes of the histogram plots.

% Set the initial default directory to the one containing the
% standard demo images for the MATLAB Image Processing Toolbox.
folder = 'C:\Program Files\MATLAB\R2010a\toolbox\images\imdemos';
if ~exist(folder, 'dir')
folder = pwd;
end
filePattern = fullfile(folder, '*.*');

% Ask user to browse for the image file.
[baseFileName, folder] = uigetfile(filePattern, 'Specify an image file');
% Construct the full file name.
fullImageFileName = fullfile(folder, baseFileName);

%-----
% Read the image into an array.
rgbImage = imread('C:\Users\Jason\Desktop\P.S.M\data baru\dataG.jpg');
%-----
```

```

% Check to see if it is color or monochrome.
[rows columns numberOfColorChannels] = size(rgbImage);
if strcmpi(class(rgbImage), 'uint8')
% Flag for 256 gray levels.
eightBit = true;
else
eightBit = false;
end
% If it's monochrome, convert it to color.
if numberOfColorChannels == 1
rgbImage = cat(3, rgbImage, rgbImage, rgbImage);
end

% Display the original RGB image.
subplot(3,4,1);
imshow(rgbImage);
% Maximize figure.
set(gcf, 'Position', get(0, 'ScreenSize'));
% drawnow; % Make it display immediately.
if numberOfColorChannels > 1
title('Original Color Image', 'FontSize', captionFontSize);
else
title('Original Monochrome Image (converted to color)', 'FontSize', captionFontSize);
end

% Extract out the individual color channels
redChannel = rgbImage(:, :, 1);
greenChannel = rgbImage(:, :, 2);
blueChannel = rgbImage(:, :, 3);

% Display the red channel image.
subplot(3,4,2);
imshow(redChannel);
title('Red Channel Image', 'FontSize', captionFontSize);
% Display the green channel image.
subplot(3,4,3);
imshow(greenChannel);
title('Green Channel Image', 'FontSize', captionFontSize);
% Display the blue channel image.
subplot(3,4,4);
imshow(blueChannel);
title('Blue Channel Image', 'FontSize', captionFontSize);

% Calculate the histogram of the red channel.
hR = subplot(3, 4, 6);

```

```

[countsR, grayLevelsR] = imhist(redChannel);
maxGLValueR = find(countsR > 0, 1, 'last');
maxCountR = max(countsR);
% Plot the histogram of the red channel.
bar(countsR, 'r');
grid on;
xlabel('Red Gray Levels', 'FontSize', axisFontSize);
ylabel('Red Pixel Count', 'FontSize', axisFontSize);
title('Histogram of Red Channel', 'FontSize', captionFontSize);

% Calculate the histogram of the green channel.
hG = subplot(3, 4, 7);
[countsG, grayLevelsG] = imhist(greenChannel);
maxGLValueG = find(countsG > 0, 1, 'last');
maxCountG = max(countsG);
% Plot the histogram of the green channel.
bar(countsG, 'g');
grid on;
xlabel('Green Gray Levels', 'FontSize', axisFontSize);
ylabel('Green Pixel Count', 'FontSize', axisFontSize);
title('Histogram of Green Channel', 'FontSize', captionFontSize);

% Calculate the histogram of the blue channel.
hB = subplot(3, 4, 8);
[countsB, grayLevelsB] = imhist(blueChannel);
maxGLValueB = find(countsB > 0, 1, 'last');
maxCountB = max(countsB);
% Plot the histogram of the blue channel.
bar(countsB, 'b');
grid on;
xlabel('Blue Gray Levels', 'FontSize', axisFontSize);
ylabel('Blue Pixel Count', 'FontSize', axisFontSize);
title('Histogram of Blue Channel', 'FontSize', captionFontSize);

% Set all axes to be the same height.
maxCount = max([maxCountR, maxCountG, maxCountB]);
% Set all axes to be the same width.
maxGL = max([maxGLValueR, maxGLValueG, maxGLValueB]);
if eightBit
maxGL = 255;
end
% If there's a big spike at the last bin, it can be hard to see because
% of the box around the axes, so make the box a few bins wider.
% This will make it easier to see spikes in the last bin.
maxGL = maxGL + 8;
axis([hR hG hB], [0 maxGL 0 maxCount]);

```

```

% Plot all
subplot(3, 1, 3);
plot(grayLevelsR, countsR, 'r', 'LineWidth', 2);
grid on;
xlabel('Gray Level', 'FontSize', axisFontSize);
ylabel('Pixel Count', 'FontSize', axisFontSize);
hold on;
plot(grayLevelsG, countsG, 'g', 'LineWidth', 2);
plot(grayLevelsB, countsB, 'b', 'LineWidth', 2);
title('Histograms of All Channels', 'FontSize', captionFontSize);
maxGrayLevel = max([maxGLValueR, maxGLValueG, maxGLValueB]);
% Trim x-axis to just the max gray level on the bright end.
if eightBit
    xlim([0 255]);
else
    xlim([0 maxGrayLevel]);
end

max_graph = max(max(maxCount));
if max_graph == maxCountR
    disp ('Red');
elseif max_graph == maxCountG
    disp ('Green');
else max_graph == maxCountB
    disp ('Blue');
end

```

APPENDIX C

```

function varargout = main(varargin)
% MAIN M-file for main.fig
%   MAIN, by itself, creates a new MAIN or raises the existing
%   singleton*.
%
%   H = MAIN returns the handle to a new MAIN or the handle to
%   the existing singleton*.
%
%   MAIN('CALLBACK', hObject,eventData,handles,...) calls the local
%   function named CALLBACK in MAIN.M with the given input arguments.
%
%   MAIN('Property','Value',...) creates a new MAIN or raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before main_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to main_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 main

% Last Modified by GUIDE v2.5 23-Oct-2010 20:52:11

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
    'gui_Singleton', gui_Singleton, ...
    'gui_OpeningFcn', @main_OpeningFcn, ...
    'gui_OutputFcn', @main_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 main is made visible.
function main_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 main (see VARARGIN)

% Choose default command line output for main
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes main wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = main_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)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
RUN chuongtrinhnhandang;

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
RUN new;

```

APPENDIX D

```

function varargout = new_color(varargin)
% NEW_COLOR M-file for new_color.fig
%   NEW_COLOR, by itself, creates a new_color NEW_COLOR or raises the existing
%   singleton*.
%
%   H = NEW_COLOR returns the handle to a new_color NEW_COLOR or the handle to
%   the existing singleton*.
%
%   NEW_COLOR('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in NEW_COLOR.M with the given input arguments.
%
%   NEW_COLOR('Property','Value',...) creates a new_color NEW_COLOR or raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before new_color_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to new_color_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 new_color

% Last Modified by GUIDE v2.5 08-Nov-2010 15:48:00

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
    'gui_Singleton', gui_Singleton, ...
    'gui_OpeningFcn', @new_color_OpeningFcn, ...
    'gui_OutputFcn', @new_color_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 new_color is made visible.
function new_color_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 new_color (see VARARGIN)

% Choose default command line output for new_color
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes new_color wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = new_color_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)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in plotAxes1_pushbutton.
function plotAxes1_pushbutton_Callback(hObject, eventdata, handles)
% hObject    handle to plotAxes1_pushbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
RUN coloring;

% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over text1.

```


**“I hereby acknowledge that the scope and quality of this thesis is qualified for the
award of Bachelor Degree of Electrical Engineering (Electronics)”**

Signature : _____

Name : _____

Date : _____

**EDU_VISTEC: A SOFTWARE FOR COMPUTER VISION EDUCATIONAL
TRAINER USING MATLAB**

LAW KOK SIN

**This thesis is submitted as partial fulfilment of the requirement for the award of
the**

Bachelor Degree of Electrical Engineering (Electronic)

Faculty of Electrical & Electronic Engineering

Universiti Malaysia Pahang

NOVEMBER 2010

“All the trademark and copyrights use herein are property of their respective owner. References of information from other sources are quoted accordingly; otherwise the information presented in this report is solely work of the author.”

Signature : _____

Name : **LAW KOK SIN**

Date : _____

To

My beloved parents

and my siblings

“who offered me unconditional love and support

throughout the course of this thesis”

Acknowledgement

First and foremost, I am heartily thankful to my supervisor of this work, Dr. Kamarul Hawari bin Ghazali, whose encouragement, guidance and advice from the initial to the final level throughout the course of my graduate studies. He inspired me greatly to work in this project. This work would not be possible without his willingness to motivate me contributed tremendously to my work.

Besides, I wish to express my gratitude to the authority of Faculty of Electrical and Electronic for providing me with the good environment and facilities to complete this work. I would like to extend my gratitude to those who gave me the possibilities for me to complete this work especially my beloved parents and my siblings for giving such a great motivation and financial support.

Finally, an honourable mention goes to my graduate friends especially to those group members under Dr. Kamarul supervision whose sharing all the literature and skills. Special thanks to the administrator who help me to find the research material easily.

Abstract

Nowadays, from time to time technologies become more advances that bring much advantage for our human life. But, it needs time for us to learn especially the knowledge about image processing because it uses for many field such as medical or aerospace. So, this project is a study that mains in the basic applications of image processing. It will introduce to the new learner about the image processing. MATLAB software is the main tools that will be use for the application of image processing. MATLAB software has been chosen because of this tool is a universal and all well known tool. Then, MATLAB GUI is one of the parts that use to create the module of those applications. The main outcome of this project is to create a module about the application of image processing for learning purpose. In this module which will cover two parts, which is shape detection and color recognition (RGB). The method that will be use for the shape detection is edge detection. A few edge-finding methods will be used for this application such as Sobel, Prewitt, Roberts, Laplacian of Gaussian (Log), zero-cross and also canny technique. Therefore, the user can learn much in this module. Then, for the application of color part used histogram to recognize the 3 basic colors, that are red, green and blue. To complete this project it need to further study about the MATLAB tools briefly and also all the technique that use to create that module. Finally, this module can give the benefit for the new learner of image processing to more understand about the useful applications. This module was designed to perform basic applications effectively.

Abstrak

Pada masa kini, kemajuan teknologi dari semasa ke semasa telah membawa banyak manfaat dalam kehidupan harian manusia. Tapi, ia memerlukan masa bagi kita untuk belajar terutama pengetahuan tentang pemprosesan imej kerana kegunaannya dalam pelbagai bidang yang amat meluas seperti perubatan atau aerospace. Jadi, projek ini merupakan kajian yang berkaitan dengan aplikasi pemprosesan imej. Ini akan memperkenalkan kepada mereka yang baru belajar tentang pemprosesan imej. MATLAB adalah alat utama yang akan digunakan untuk aplikasi pemprosesan imej. MATLAB telah dipilih kerana alat ini adalah universal dan semua amat mengenalinya dalam bidang kejuruteraan ini. Kemudian, MATLAB GUI merupakan salah satu bahagian yang digunakan untuk membuat modul dari aplikasi tersebut. Objektif utama dari projek ini adalah menghasilkan modul tentang aplikasi pemprosesan imej untuk tujuan belajar. Dalam modul ini akan merangkumi dua bahagian, iaitu pengenalanpastian bentuk dan warna (RGB). Kaedah yang akan digunakan untuk pengesanan bentuk adalah pengesanan tepi. Teknik ini merangkumi beberapa kaedah yang akan digunakan untuk aplikasi seperti teknik Sobel, Prewitt, Roberts, Laplacian dari Gaussian (Log), zero-cross dan juga canny. Oleh kerana itu, pengguna boleh banyak belajar di dalam modul ini. Maka, untuk aplikasi bahagian warna yang digunakan adalah histogram untuk mengenali 3 warna iaitu merah, hijau dan biru. Demi menyiapkan projek ini perlu mempelajari lebih lanjut mengenai alat MATLAB terlebih dahulu dan juga semua teknik yang digunakan untuk membuat modul itu. Akhirnya, modul ini dapat memberikan manfaat bagi mereka baru ingin belajar pemprosesan imej untuk lebih memahami tentang aplikasi bermanfaat yang ada. Modul ini direka dengan tujuan mampu menonjolkan aplikasi asas dengan berkesan.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
	ORGANIZATION OF THE THESIS	
	TITLE PAGE	I
	DECLARATION	II
	DEDICATION	III
	ACKNOWLEDGEMENT	IV
	ABSTRACT	V
	ABSTRACT	VI
	TABLE OF CONTENTS	VII
	LIST OF FIGURE	VIII
	LIST OF TABLE	XI
	LIST OF ABBREVIATIONS	XII
	LIST OF APPENDICES	XIII
1.	INTRODUCTION	1
	1.1 Problem Statements	2
	1.2 Objectives	2
	1.3 Scope of project	2
	1.4 Thesis Outline	3

2. LITERATURE REVIEW

2.1	Image processing	4
2.1.1	Threshold image	7
2.1.2	Edge Detection	10
2.1.3	Histogram Method	12
2.2	RGB	15
2.3	MATLAB 2008	16
2.4	Graphical User Interface (GUI)	17
2.4.1	GUI definition	17
2.4.2	MATLAB GUI	18
2.4.3	MATLAB GUIDE	19
2.4.4	GUI operation	19

3. METHODOLOGY

3.1	Introduction	21
3.2	Software Development	23
3.2.1	Development MATLAB GUI Using MATLAB GUIDE	25
3.2.2	Build MATLAB Programming	30

4. RESULT AND DISCUSSIONS

4.1	Introduction	35
4.2	Training phase result	36
4.2.1	Result of shape detection development	36
4.2.2	Result of color recognition development	39
4.3	Execution phase result	43
4.4	Discussion	47

5. CONCLUSION AND RECOMMENDATION

5.1	Conclusion	48
5.2	Future Recommendation	49

REFERENCES	50
-------------------	-----------

APPENDICES	51
-------------------	-----------

LIST OF FIGURES

FIGURE	TITLE	PAGE
2.1	Example of restoration image	5
2.2	Example of enhancement image	5
2.3	Example of understanding images	6
2.4	Original image	8
2.5	RGB to gray scale	8
2.6	Threshold to binary image	8
2.7	Inverse binary image	8
2.8	Prewitt horizontal and vertical operators	11
2.9	Example of image histogram	13
2.10	Example of histogram show pixel value	13
2.11	Low key image with the majority of pixels to the left of center of the graph	14
2.12	High key images with the majority of the pixels to the right of center of the graph	14
2.13	Result of combination primary color	15
2.14	The application, developed in MATLAB, directly acquires signals from measurement hardware, performs analysis and plotting, and includes GUI controls.	16
2.15	Example of GUI elements	18
3.1	Block diagram for the flow of the project	22
3.2	Flow Chart	24

3.3	MATLAB GUIDE Layouts	26
3.4	Property Inspector	27
3.5	Example GUI	28
3.6	Example m-files for GUI	29
4.1	Coding for sobel method	37
4.2	Coding shown decision of detect image	38
4.3	Coding of read and check the image	39
4.4	Coding of extract out the color image to the individual color channels	40
4.5	Coding of after calculate the pixel value and plot in a graph	41
4.6	Final result	42
4.7	Main Menu GUI	43
4.8	Load Image	44
4.9	Select edge-detection method	44
4.10	Recognize and get result	44
4.11	GUI for the color recognition (Part I)	45
4.12	GUI for the color recognition (Part II)	46
4.13	GUI for the color recognition (Part III)	46

LIST OF TABLE

TABLE PAGE	TITLE	
3.1	Basic Matlab GUI component	25
3.2	Various kind of Callbacks	31
3.3	Major Sections of the GUI M-file	34

LIST OF ABBREVIATIONS

GUI	-	Graphical User Interface
RGB	-	Red, Green and Blue
HSV	-	hue, saturation, and value

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
A	Coding GUI Shape Detection	51
B	Coding Color Recognition	56
C	Coding Main GUI	60
D	Coding Color GUI	62