

**HIGH BEAM CONTROLLER FOR VEHICLE  
USING IMAGE PROCESSING TECHNIQUES**

By

MD. KHAIRUL AMRI BIN NORDIN

FINAL PROJECT REPORT

Submitted to the Electrical & Electronics Engineering Programme  
in Partial Fulfillment of the Requirements  
for the Degree  
Bachelor of Engineering (Hons)  
(Electrical & Electronics Engineering)

Universiti Teknologi Petronas  
Bandar Seri Iskandar  
31750 Tronoh  
Perak Darul Ridzuan

© Copyright 2007

by

Md. Khairul Amri Bin Nordin, 2007

# **CERTIFICATION OF APPROVAL**

## **HIGH BEAM CONTROLLER FOR VEHICLE USING IMAGE PROCESSING TECHNIQUES**

by

Md. Khairul Amri Bin Nordin

A project dissertation submitted to the  
Electrical & Electronics Engineering Programme  
Universiti Teknologi PETRONAS  
in partial fulfilment of the requirement for the  
Bachelor of Engineering (Hons)  
(Electrical & Electronics Engineering)

Approved:

---

Miss Illani Mohd Nawi  
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS  
TRONOH, PERAK

December 2007

## **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.

---

Md. Khairul Amri Bin Nordin

## **ABSTRACT**

Pattern recognition and image processing techniques has much influence in computer vision. Features have been developing to improve the system in order to solve problems in computer vision. There have been many attempts at applying this technology for human benefits. This project requires the author to improve existing High Beam Controller for Vehicles prepared by Syazwan Hamizin bin Mohd @ Sulaiman and his project partner, Nor Sulyanie bt Sulong in year 2005. The objective of this project is to modify the existing system to allow the detection of approaching vehicle more precise using image processing techniques. The existing system uses Light Dependent Resistors which have contra especially precision of detecting various light sources resulting errors to the system. At the end of this project, a working prototype should be produce which will be performed by the author and his partner. The author is assigned to do the software part which relevant software has been developed and will integrate with hardware part. The scope of study would be on the automotive lighting characteristics, imaging devices, image processing control software simulations, and integrated automotive lamp controller for vehicle lighting circuit. The methodology required for the program structure start with the image acquisition, image enhancement, edge detection, segmentation, feature extraction, recognition, and interpretation. The hardware part which prepared by the author's partner gave a challenging task in order to integrate both software and hardware systems but offered an opportunity to experience how the technology is linked to the outside world.

## **ACKNOWLEDGEMENTS**

First and foremost, I would like to take this opportunity to express my sincere appreciation to my project supervisors, Miss Illani Mohd Nawi for her endless support, comments as well as constructive suggestions through the course of this final year project and preparation of this dissertation.

My project partner who has worked with me for past two semesters, Muhamad Khairudin Bin Ab Wahab, deserves credit for helping extensively with the development of this project. His contributions to this project deserve special recognition.

Besides, I would like to thank engineering lab assistant for providing help when needed without any hesitation subsidizing and providing all the necessary equipment to apply technical skills through this project.

Last but not least, my fellows friends and who gave me useful and precious advice. They have given me a path of objective, knowledge and self-esteem and I wish to express my heartfelt gratitude to all my family for their continued support and encouragement through the studies at Universiti Teknologi PETRONAS.

## TABLE OF CONTENTS

|   |      |
|---|------|
| LIST OF TABLES .....  | viii |
| LIST OF FIGURES .....   | ix   |
| CHAPTER 1 INTRODUCTION .....                                    | 1    |
| 1.1 Background of Study.....                                    | 1    |
| 1.2 Problem Statement .....                                     | 3    |
| 1.3 Significant of the Project.....                             | 3    |
| 1.4 Objective and Scope of Study .....                          | 4    |
| 1.4.1 Objectives .....  | 4    |
| 1.4.2 Scope of Study .....                                      | 4    |
| CHAPTER 2 LITERATURE REVIEW AND THEORY .....                    | 5    |
| 2.1 Software .....  | 5    |
| 2.2 Image Processing Theory .....                               | 5    |
| 2.2.1 Pattern Recognition and Classification as Knowledge ..... | 5    |
| 2.2.2 Image Pre-processing.....                                 | 7    |
| 2.2.3 Edge-based Segmentation.....                              | 7    |
| 2.2.4 Shape Representation and Description .....                | 8    |
| 2.2.5 Recognition and Interpretation .....                      | 8    |
| CHAPTER 3 METHODOLOGY/PROJECT WORK .....                        | 9    |
| 3.1 Project Process Flow .....                                  | 9    |
| 3.1.1 Title Selection .....                                     | 10   |
| 3.1.2 Process and Planning .....                                | 10   |
| 3.1.3 Literature Review .....                                   | 10   |
| 3.1.4 Learning Software Languages .....                         | 11   |
| 3.1.5 Designing Program Structure .....                         | 11   |
| 3.1.6 Simulation.....   | 11   |
| 3.1.7 Integrate Software and Hardware System .....              | 11   |
| 3.1.8 Comb Test.....  | 11   |
| 3.2 Project Work .....  | 12   |
| 3.2.1 Processing Method.....                                    | 12   |
| 3.3 Tool / Equipment.....                                       | 15   |
| 3.3.1 Web Cam .....   | 15   |

|  |    |
|--|----|
| 3.3.2 Image Processing Software.....   | 15 |
| 3.3.3 MATLAB 7.0 .....   | 16 |
| 3.3.4 MICROSOFT VISUAL C++.....  | 16 |
| CHAPTER 4 RESULTS AND DISCUSSION.....  | 17 |
| 4.1 Findings.....  | 17 |
| 4.1.1 Simulation Results.....  | 17 |
| 4.1.2 C++ Testing and Results.....   | 25 |
| a) Samples.....  | 25 |
| b) Program Developed.....  | 25 |
| c) Result of Processed Image.....  | 27 |
| 4.2 Discussion.....  | 29 |
| 4.2.1 Simulation.....  | 29 |
| 4.2.2 Samples.....   | 30 |
| 4.2.3 Methodology of the Project.....  | 30 |
| 4.2.4 Evaluation of Results.....   | 33 |
| 4.2.5 Limitations.....   | 36 |
| CHAPTER 5 CONCLUSION AND RECOMMENDATIONS.....  | 37 |
| 5.1 Conclusions.....   | 37 |
| 5.2 Recommendations and Future Works.....  | 38 |
| REFERENCES.....  | 40 |
| APPENDICES.....  | 42 |
| Appendix A GANTT CHART.....  | 43 |
| Appendix B C++ MAIN PROGRAM.....   | 45 |
| Appendix C CImg.H (IMAGE HEADER FILE) [12].....  | 46 |
| Appendix D High Beam Image Processing Using MATLAB<br>(Simulation).....                      | 49 |
| Appendix E Light Intensity And Image Contour For High Beam And<br>Low Beam Using MATLAB..... | 50 |
| Appendix F MATLAB Source Code.....   | 52 |
| Appendix G MATLAB Image Processing Toolbox.....  | 53 |

## LIST OF TABLES

|   |    |
|---|----|
| <b>Table 1</b> List of car's light distribution.....                | 25 |
| <b>Table 2</b> Output Image.....                                    | 27 |
| <b>Table 3</b> The step by step of the image processing method..... | 30 |
| <b>Table 4</b> The Evaluation of the Result.....                    | 35 |



## LIST OF FIGURES

|  |    |
|--|----|
| <b>Figure 1</b> Traffic Mileage at Night and Accident Consequences [2] ..... | 2  |
| <b>Figure 2</b> Performance of the Human Eye [2] .....                       | 3  |
| <b>Figure 3</b> Project Flow Chart .....                                     | 9  |
| <b>Figure 4</b> The Step of Processing Method .....                          | 12 |
| <b>Figure 5</b> MATLAB Source Code .....                                     | 17 |
| <b>Figure 6</b> Low beam image on the screen.....                            | 18 |
| <b>Figure 7</b> RGB to grayscale image conversion.....                       | 18 |
| <b>Figure 8</b> Edge detection result.....                                   | 19 |
| <b>Figure 9</b> Importing the reference image.....                           | 20 |
| <b>Figure 10</b> Importing the capture image (high beam).....                | 21 |
| <b>Figure 11</b> Binary image conversion using Threshold Tool.....           | 21 |
| <b>Figure 12</b> Morphology filtering using Dilation method.....             | 22 |
| <b>Figure 13</b> Edge detection using Edge Finder Tool.....                  | 23 |
| <b>Figure 14</b> Comparison of the images using Match Tool.....              | 23 |
| <b>Figure 15</b> Complete script (left) and Debugging Process (right).....   | 24 |
| <b>Figure 16</b> The algorithm using Microsoft Visual C++.....               | 26 |
| <b>Figure 17</b> Low Beam Reference Image.....                               | 34 |
| <b>Figure 18</b> High Beam Reference Image.....                              | 34 |
| <b>Figure 19</b> Original Image of High Beam .....                           | 49 |
| <b>Figure 20</b> Grayscale Conversion.....                                   | 49 |
| <b>Figure 21</b> Edge Detection Process .....                                | 49 |
| <b>Figure 22</b> High beam of incoming traffic at 5 m.....                   | 50 |
| <b>Figure 23</b> High beam of incoming traffic at 5 m after processed .....  | 50 |
| <b>Figure 24</b> Low beam of incoming traffic at 5 m .....                   | 51 |
| <b>Figure 25</b> Low beam of incoming traffic at 5 m after processed .....   | 51 |

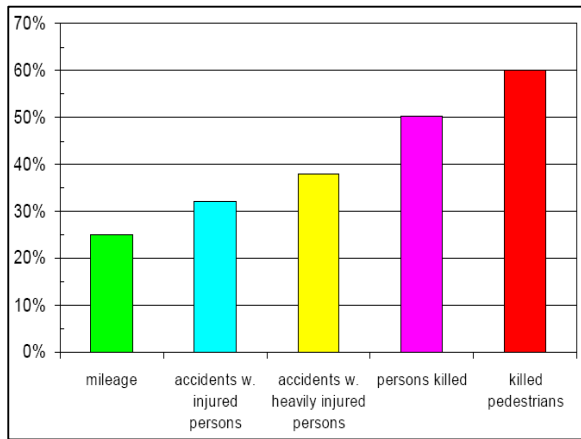
# CHAPTER 1

## INTRODUCTION

### 1.1 Background of Study

The increasing of the number of accidents in recent years in this country is very serious issue to country's road safety, particularly in the area of highway engineering. Every year, the number of fatalities is over 5,500 people and injuring a further 46,400, thus wasting economic cost to the community; over RM 1,500 million a year. [1]

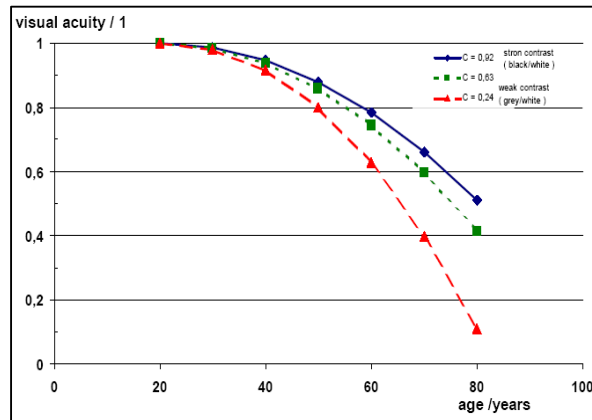
Since the founder Thomas Edison invented the light bulb in 1879, lighting is becoming one of the wide ranges of area such as in automotive applications. Recent years, research and development in this area are highly invested by manufacturers to offer solutions of considerable advantages that can fulfill their customer's technical and quality specifications and improve human visibility especially when driving car as well. A winning automotive design relies on more than just good looks. Many aspect need to considered by the manufacturers and safety of the customers is the most important thing that comes first. Study shows that at certain distance of mileage the most involved in accidents is pedestrians. [2]



**Figure 1** Traffic Mileage at Night and Accident Consequences [2]

The lighting systems of a car provide a variety of challenges from the point of view of illumination science and technology. Automotive lighting requires sources that combine brightness, small size, and energy efficiency. Existing technologies are refined and redesigned for different applications as car themselves are redesigned and as new technologies become options for interior and exterior automotive lighting, roadway lighting, and traffic signals. It wasn't long after automobiles began to be used widely that lighting systems allowing their drivers to see at night began to be developed.

Now, headlights are required equipment on all cars, and these systems must meet specific requirements standardized by the regulation in terms of their distributions of luminous intensity and color. The research and development in the lighting technology is done with main objective to improve the human visibility especially when driving a car at night. The next figure show the human visibility is reducing as our age is increasing. [2]



**Figure 2** Performance of the Human Eye [2]

## 1.2 Problem Statement

Excessive glare from incoming traffic is one of the major factors to caused accident when driving a car at night. Excessive glare or can be known also as blinding phenomenon is basically caused from the approaching driver’s high beam. Generally, the driver forgot to turn off their high beams and this might ‘blind’ or reduces the opposite driver’s vision [3]. Besides that, driver also did not want to turn off their high beam when there is opposite driver especially when they driving the car in the middle of a very dark place because they lack of visibility during night. Furthermore, the slow reaction from the driver to change the light’s controlled switch from high beam to low beam also can cause blinding.

## 1.3 Significant of the Project

This project will give benefits to vehicle’s users especially when they are driving at night because it will be lot more easily to drive without manual switching method. The automatic switching method will improve driver’s visibility for both opposite direction of driver because there is no blind phenomenon occurs. The result of this project will reduce the number of accident occurred especially at night.

## **1.4 Objective and Scope of Study**

### ***1.4.1 Objectives***

The main purpose of this project is to create a working program which able to identify incoming traffic at night and sends signals to hardware systems to switch off high beam lamp, if being used. This is to reduce glairing to opposite drivers. If every vehicle equipped with such system, hopefully number of accidents can be reduces.

### ***1.4.2 Scope of Study***

The scope of study would be on the automotive lighting characteristics, imaging devices, image processing control software simulations, and integrated automotive lamp controller for vehicle lighting circuit. From our findings, most of the drivers are looking for effective, low cost high beam controller for driving satisfaction and safety. The study hopefully can enhance the author's knowledge in order to use the software in order to process and differentiate high beam and low beam using image processing techniques.

## **CHAPTER 2**

### **LITERATURE REVIEW AND THEORY**

#### **2.1 Software**

Received image has been processed using the software according to the stated coding. The proposed software that has been used is Microsoft Visual C++ for application program and MATLAB and Global Lab Image/2 will be using during simulation for the image processing. Image processing is any form of information processing for which both the input and output are images, such as photographs or frames of video. Most image processing techniques involve treating the image as a two-dimensional signal and applying standard signal processing techniques to it.

#### **2.2 Image Processing Theory**

##### ***2.2.1 Pattern Recognition and Classification as Knowledge***

The concept of a pattern is universal in intelligence and discovery of knowledge. In fact, it is the ability to discern patterns that gives human beings the power to perceive information in various form and subsequently allows us to consolidate it into useful; knowledge. Two inherent in any recognition process, are the ability to generalize (likeness) and to discriminate (differences) at the same time.

Not even simplest machine vision tasks can be solved without the help of recognition. Pattern recognition is used for region and object classification, and basic methods of pattern recognition must be understood in order to study more complex machine vision processes. Almost always, when information about an object or region class is available, some pattern recognition method is used.

But no recognition is possible without knowledge. Decisions about classes or groups into which recognized object are classified are based on knowledge – knowledge about objects and their classes gives the necessary information for object classification. Both specific knowledge about the objects being processed and hierarchically higher and more general knowledge about object classes required.

In order to simplify the task of computer vision understanding, two levels are usually distinguished, low-level and high-level image processing. Low-level image processing and high-level image computer vision differs in the data used.

Low-level methods usually use very little knowledge about the content of the images. In the case of the computer knowing image content, it is usually provided by high-level algorithms or directly by a human who knows the problem domain. Low-level methods often include image compression, pre-processing methods for noise filtering, edge extraction and image sharpening. Low-level image processing uses data which resemble the input image. If the image is to be processed using a computer it will be digitized first, after which it may be represented by rectangular matrix with elements corresponding to the brightness at appropriate image locations. Such matrices are the inputs and outputs of low-level processing.

High-level processing is based on knowledge, goals and plans of how to achieve those goals, and artificial intelligence methods are widely applicable. High-level computer vision tries to imitate human cognition and the ability to make decisions according to the information contained in the image.

Computer vision is based on high-level processing, and the recognition process is tightly bound to prior knowledge about the image content. An image is mapped into a formalized model of the world, but this model does not remain unchanged. Although the initial model may consist of some general a priori knowledge, high-level processing constantly extracts new information from the images and updates and clarifies the knowledge.

Experience shows that a good knowledge representation design is the most important part of solving the understanding problem. Moreover, a small number of relatively simple control strategies are often sufficient for AI (Artificial Intelligence) Systems to show complex behavior, assuming an appropriate complex knowledge base is available. In other words, a high degree of control sophistication is not required for intelligent behavior, but a rich, well-structured representation of a large set of priori data hypotheses is needed. [6]

### ***2.2.2 Image Pre-processing***

Pre-processing is the name used for operations on images at the lowest level of abstraction – both input are, pre-intensity images. Preprocessing does not increase image information content. From the information-theoretic viewpoint it can thus be concluded that the best preprocessing is no pre-processing, and without question, the best way to avoid pre-processing is to concentrate on high-quality image acquisition.

Nevertheless, pre-processing is very useful in a variety of situations since it helps to suppress information that is not relevant to the specific image processing or analysis task. Therefore, the aim for pre-processing is an improvement of the image data that suppress undesired distortions and enhances some image features important for further processing.

Image processing method uses the considerable redundancy in images. Neighboring pixels, corresponding to one object in real images have essentially the same or similar brightness value, so if a distorted pixel can be picked out from the image, it can usually be restored as an average value of neighboring pixels. [6]

### ***2.2.3 Edge-based Segmentation***

Edge-based segmentation represents a large group of methods based on information about edges in the image; it is one of the earliest segmentation approaches and still remains very important. Edge-based segmentation relies on the edges found in an image by edge detecting operators - these edges mark image locations of discontinuities in gray-level, color, texture, etc. Normally the image resulting from edge detection cannot be used as the segmentation result.



Supplementary processing steps must follow to combine edges into edge chains that correspond better with borders in the image. The final aim is to reach at least partial segmentation-that is, to group local edges into an image where only edge chains with a correspondence to existing objects or image parts are present. [6]

#### ***2.2.4 Shape Representation and Description***

Defining the shape of an object can prove to be very difficult. Shape is usually represented verbally or in figures, and people use terms such as elongated, rounded, with sharp edge, etc. The computer era has introduced the necessity to describe even very complicated shape precisely.

Sharp description generates a numeric feature vector or a non-numeric syntactic description world, which characterize properties of the describe region. In many task, it is important to represent classes of shapes properly, e.g, shape classes of cars, buses, lorries and vans. Shape classes represent the generic shapes of the objects belonging to the class and emphasize shape differences among classes. The influence of shape variations within the classes should not be reflected in the class description. [6]

#### ***2.2.5 Recognition and Interpretation***

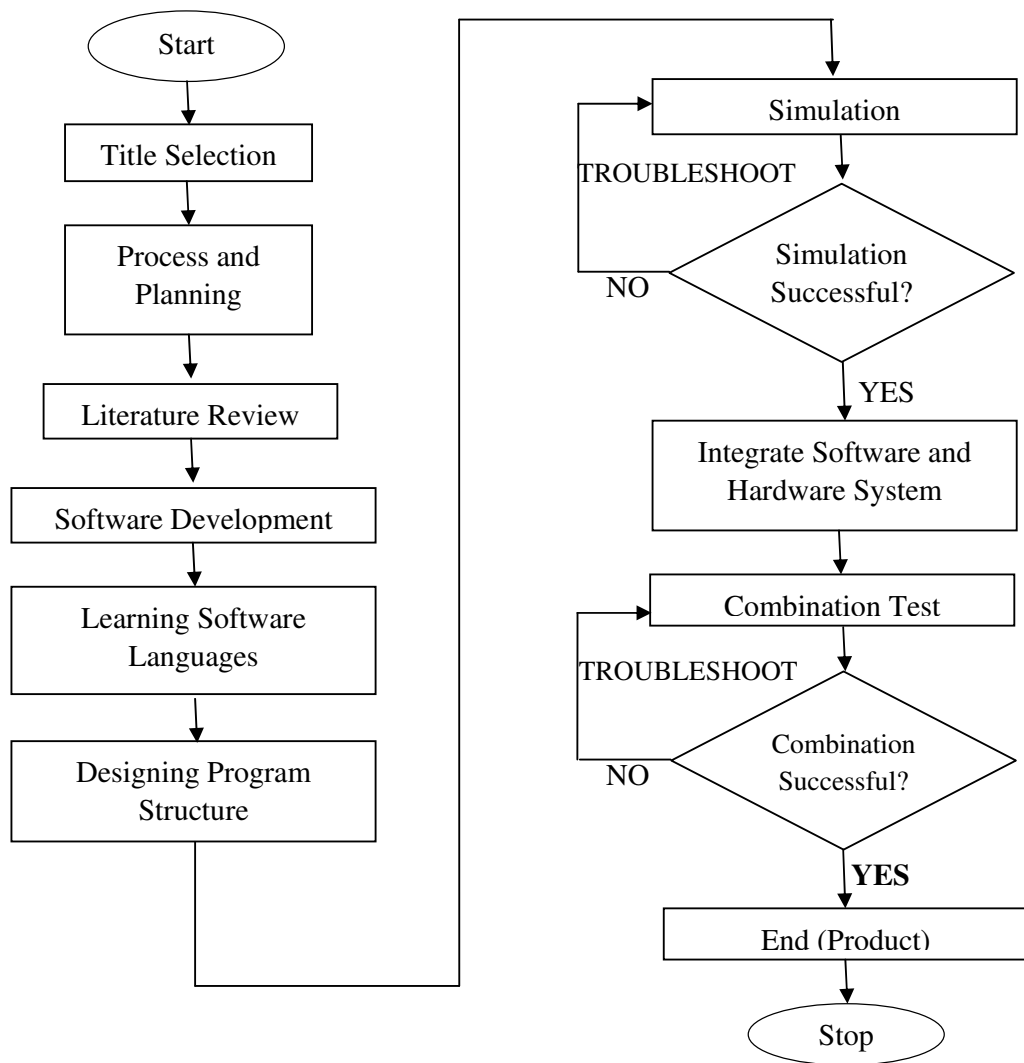
Pattern refers to a collection of features (descriptors) derived from unknown objects. Pattern classification is typically the final step in the development of computer vision algorithm. It refers simply to the process whereby an unknown object within an image is identified as belonging to one particular group.

Recognition is the process of correctly assigning unknown patterns to their respective pattern class based on the information provided by the descriptor. The recognition process should be completely automatic and error free in vision applications. But in practice, it is not always a simple task. [6]

# CHAPTER 3

## METHODOLOGY/PROJECT WORK

### 3.1 Project Process Flow



**Figure 3** Project Flow Chart

In order to achieve the final complete project from the piece of concept need a lot of procedures. This section will describes more detail each procedures and will becoming the prime guide in order to come out with the final project. Thus, one of the important steps that need to be taken is debugging at every step in this procedures. This involve with the correction, which is done to meet the specification of each step. Besides that, all procedures need to be well-planned in details and systematic to avoid problem in the following procedures.

### ***3.1.1 Title Selection***

The title is chosen either proposed by the lecturer or own topic proposed by FYP students. The final topic for the author is ‘High Beam Controller Using Image Processing Techniques’ and being supervised under Miss Illani Mohd Nawi.

### ***3.1.2 Process and Planning***

The author has completed to identify all procedures need to be taken in order to achieve the objective of this project. The procedures can be referred in Figure 3 which summarizes by the author. All the procedures are well planned in details and systematic manner to prevent problem during the completion of the project.

### ***3.1.3 Literature Review***

It is the study of the related parts of this project. It includes research and thorough study to understand the key concepts and the relationships between them. Literature review gives the author basic understanding to manage the project well as planned. It also includes the problem definition of the project and objective need to be achieved by the author. Although this part involved at early stage, it doesn’t mean that the learning process of the theory is stop. Improvements need to be taken from time to time in order to enhance the knowledge and more information can be collected to achieve the objective of this project. So, it will be a continuous process of study.

### ***3.1.4 Learning Software Languages***

Generally the most important part is processing the high-beam image. MATLAB, C++, and Global Lab Image/2 will be used during this project as the command languages. Author need to study all the possible command and understand each command before apply it to this project.

### ***3.1.5 Designing Program Structure***

Before design the program structure, it will be easier for the author to draft the program algorithm which is the flow of the image processing process start from the receiving of the image from the camera until completely process to identify the pattern of the receive image. An algorithm is a play safe procedure which can detect error beforehand. It is an operation procedure that is written in a simple and easy to understand language that describes the flow of the actual program.

### ***3.1.6 Simulation***

A complete program structure needs to test its functionality. Then it is come to the simulation test to check the system capability to process the receive image. If the simulation not succeed, troubleshooting the program is necessary in order to acquire the actual output.

### ***3.1.7 Integrate Software and Hardware System***

Both hardware and software is combined into one system.

### ***3.1.8 Comb Test***

Combination test can be performing at practical situation. These mean that the combination unit or end product will be testing during the real situation. The product should operate accordingly to the requirements after setup it in the vehicle.

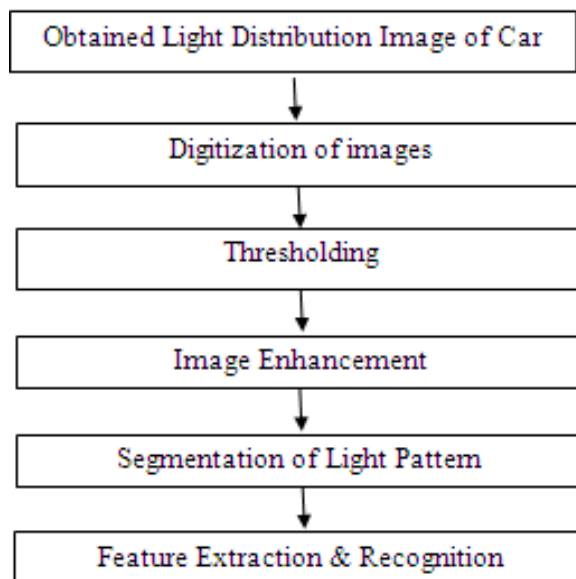
## 3.2 Project Work

The image database used in this project comprises car's light distribution of low beam mode and high beam mode obtained manually captured from the digital camera in certain condition. There are 3 different manufactures of car with each car had two lighting mode.

The project work is basically followed the flow shown in Figure 3. In order to achieve the objective to produce an output image that is clear and able to detect the lighting mode, a method using morphological operation was created. The method consists of threshold and edge detection.

### 3.2.1 Processing Method

The step involved in this method basically shown in Figure 4. After the car light distribution was obtained, the first stage is digitized the image. The digitized image is then was enhanced using the edge detection technique to detect the current lighting mode. The third stage is applying the segmentation in order to further highlight the region of interest. After the current switching mode is clearly detected, the light circuit will automatically trigger according to the user setting.



**Figure 4** The Step of Processing Method

### **Stage 1: Digitization of Images**

Total of 6 lighting images from 3 different manufacturer obtained were digitized using a digital camera to capture the light distribution. The digital camera is the substitute of the web cam used in this project. The digitized image is stored in the workstation in the BMP format. The compression of the file might causes losses, but the affect is acceptable. For testing purpose, digital camera replaces webcam.

### **Stage 2: Thresholding**

The acquire image is further processed by segmentation method that used the threshold technique. The step is to segment the image so that all potential objects will be given an equal value and everything else is given a value of zero. In order to automate Thresholding, it is necessary to determine some statistic of an image so that mathematical method can be used to compute a threshold value. This method uses a histogram of the data (number of pixels vs. intensity) and iteratively selects a value which divides the histogram so its two halves are 'balanced' around the chosen value.

Applying this value to the image yields a segmentation result in which interest area are masked with a value of 1 and everything else in the image is black (value of 0) that is equivalent to 0 or 255 pixels. The choice of threshold varied manually until interested area has been detected. When we specify the threshold value, let say 20, all the pixel value lower than that is converted to 0 (black) and the value higher than that is converted to 1 (white).

### **Stage 3: Image Enhancement**

This operation is applied to the images to eliminate the other light source because it is necessary to enhance the visibility and detect ability of light distribution. A suitable morphological filtering used to remove the slow rate of variation of the image intensity values and to enhance the image contrast. In morphology, filtering is performed using kernel. The local neighborhood with respect to the operations are performed is defined by the shape and size of the kernel that is employed. In this stage, erosion is choosing to remove the noise in the binary image which is the uninvolved white pixels.

#### **Stage 4: Segmentation of light pattern**

After the procedure above, the edge detection technique is carried out. Edge detection technique is to make edges of the image more visually prominent. A digital edge may be defined as the boundary between two regions that appears when brightness values of two regions are 'significantly different'. The type of edges detected in the images is the step edge. Step edge separate two regions in an image like the boundary between an object and the background. Edges are characterized by changes in gray values and can be detected by observing value of derivatives of the image function and the finite differences for digital image.

For this project, the edge detection is done for a binary image, which after Thresholding and enhancement. Thus the edge is detected by the difference between '0' and '1'. Consider a region of size 3x3 with  $i$  as the central pixel. The gray values in the region can be represented by  $\mathbf{Z} = (a,b,c\dots i)$  while the weighting coefficient vector represented by  $\mathbf{W} = (w_1,w_2,\dots w_n)$ . Edges can be detected by fixing a threshold on the value of  $\mathbf{S}$ , given by;

$$\mathbf{S} = \mathbf{Z}'\mathbf{W}$$

The threshold value of these operators can be select by users as required [8]. Lower threshold value yields thick edges and higher threshold value yield thin edges. After getting the edge of the light distribution, the image is highlighted with the 'white' markers by multiplying with pixel value '255'. The resulting image is the overlay back on the original image by adding the two images. From here, the output image that shows the interested area with each one is mark by white edges is obtained.

#### **Stage 5: Feature extraction and recognition**

From the output image obtained above, an analysis of the images can be performed to conclude that the lighting mode is low beam or high beam. In this project, this stage are not performed practically, instead it is done theoretically due to unfortunate circumstances. The potential of using numerical analysis of light distribution can distinguish between low beam or high beam is explored. The shape property of the light distribution is determined in reaching the conclusion of the switching mode.

### **3.3 Tool / Equipment**

The selection of the need for and the implementation of a project especially for the image processing system is a very important key in order to achieve a good result. Many important factors need to be considered such as compatibility between hardware and software, limitations, and etc because it will affect the system's performance. Following are the components involved in this project.

#### ***3.3.1 Web Cam***

Instead of using a 2.0 Megapixel Nikon digital camera, the selection of web cam as an imaging device involves three main criteria which were color acquisition, object motion and image resolution. Factors that lead in using the web cam as the imaging device is the price is cheaper and its advantage in size when place in the car compare to the digital camera. Web-accessible cameras typically involve a digital camera which uploads images to a web server, either continuously or at regular intervals. This may be achieved by a camera attached to a PC, or by dedicated hardware. The type of web cam to be used is Creative web cam.

#### ***3.3.2 Image Processing Software***

The software which is installed in the host computer (laptop) should be compatible with the web cam. The software will initiate the web cam to capture the image from the motion of approaching car. It converts the analogue image data from the camera into digital data. The software should be user-friendly and supports multiple programming. Image processing software such as Global Lab Image/2 will has been used for simulation before the system is implemented using C++ program structure. The simulation will give an early overview about the image processing process start from the acquiring the light image till the comparison of the detected light source. Simulation will give advantage to the author as it will give problem that might occur during application. This will give time to the author to solve the problem and improve the system as well.



### **3.3.3 *MATLAB 7.0***

Using the MATLAB 7.0 is another preliminary step besides using C++. The reason of using different software is to compare the results. It is to determine which software will give the best image processing. But it is important to remind that the algorithm or the coding must be considered in order to get the best result. MATLAB 7.0 has set of Toolboxes which are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others. MATLAB can produce many functions in order to process the image. The image processing coding can be simply found using the toolbox provides by the MATLAB. The list of command in the MATLAB Image Processing Toolbox can be referred in Appendix G.

### **3.3.4 *MICROSOFT VISUAL C++***

The algorithms developed in this project are all presented in C++ code. This language has been chosen because of its wide acceptance by the digital processing community. This increases C++ code readability and, at the same time, reduces software robustness, especially with respect to inappropriate subroutine parameter passing at subroutine calls. The algorithms described are compatible with each other. [4]

## CHAPTER 4

### RESULTS AND DISCUSSION

#### 4.1 Findings

##### 4.1.1 Simulation Results

At early stage of the project, the main purpose is to create a simulation program for a proposed system to allow the detection of approaching vehicle more precise using image processing techniques. After studies in all aspect related to this project, the author has simulated the system using MATLAB and Global Lab Image/2. Image processing can be achieved using both simulations' software. MATLAB already have its own image processing coding that can be found in the Toolboxes. *Toolboxes* which are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems [9].

Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others. MATLAB can produce many functions in order to process the image. Both simulations were based on the designed program structure in C++ language.

#### i. MATLAB 7.0

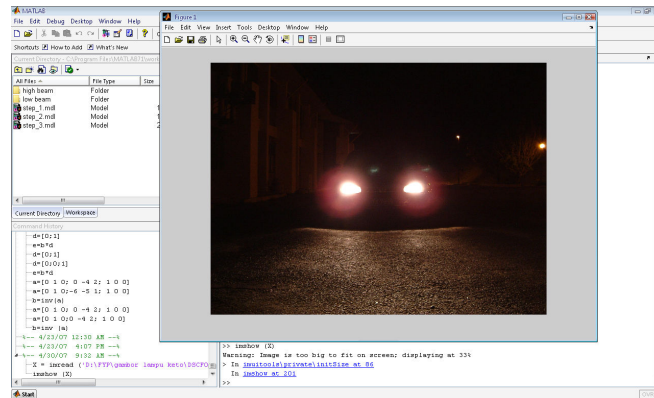
- *Program Developed as referred to Appendix F*

```
X = imread('D:\FYP\gambar lampu keto\DSCF0001.JPG') %import image from directory%
imshow(X) %show the original image%
L=im2bw(X,0.8) %convert image RGB to grayscale%
figure,imshow(L) %show the grayscale image%
L1=edge(L,'sobel') %process the image using 'sobel' edge detector's method%
figure,imshow(L1) %show the results image%
L2=edge(L,'canny') %process the image using 'canny' edge detector's method%
figure,imshow(L1) %show the results image%
```

**Figure 5** MATLAB Source Code

- **Image Acquisition**

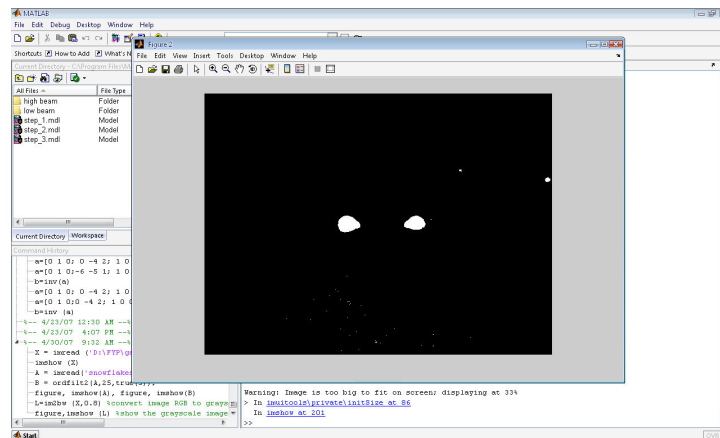
In simulation, the image is imported manually from the local folder of the host computer. The simulation will use JPG as the file format for the image taken. By using the import command provide in the Toolboxes, MATLAB will read the image file from the stated destination folder. After that, the simulation will show the imported picture on the screen.



**Figure 6** Low beam image on the screen

- **RGB to Grayscale Conversion**

The image then will convert in grayscale image before proceed to the next step. The next step which is edge detection requires the image to be in the grayscale image. Figure below show the resulting image of the grayscale conversion process.



**Figure 7** RGB to grayscale image conversion

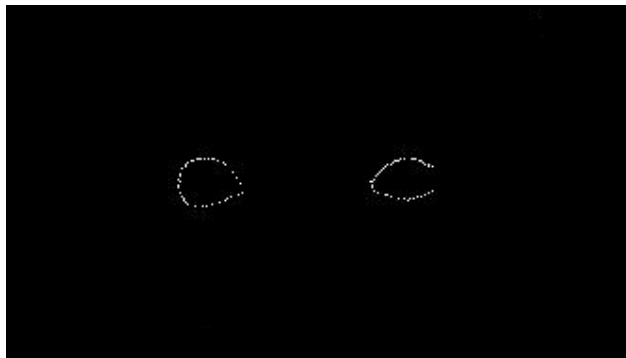
- Edge Detection

The next step of the simulation is edge detection process. The image will be analyzed about its structure. In an image, an edge is a curve that follows a path of rapid change in image intensity. Edges are often associated with the boundaries of objects in a scene. Edge detection is used to identify the edges in an image. MATLAB provide the user edge function to find edges. This function looks for places in the image where the intensity changes rapidly, using one of these two criteria:

- Places where the first derivative of the intensity is larger in magnitude than some threshold
- Places where the second derivative of the intensity has a zero crossing

Edge provides a number of derivative estimators, each of which implements one of the definitions above. For some of these estimators, user can specify whether the operation should be sensitive to horizontal edges, vertical edges, or both. Edge returns a binary image containing 1's where edges are found and 0's elsewhere.

In this simulation, the author use *Sobel* method to detect edges. Figure below shows the result.



**Figure 8** Edge detection result

High beam also has been tested using the same procedure above starting from the image acquisition till the edge detection. The result for the high beam can be referred at the Appendix D.

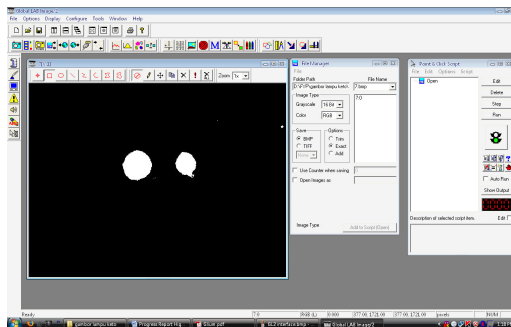
The author also used MATLAB's contour command to check the light intensity and light pattern of the low beam and high beam at 5 meter distance. The result which attached in Appendix E shows that the pattern is not the same as the theory when using the digital camera as capture device because the capture device cannot detect the light intensity. Besides, others factor that lead such phenomenon is the combination of the light distribution, right and left headlamp give the different light pattern.

## ii. Global Lab Image/2

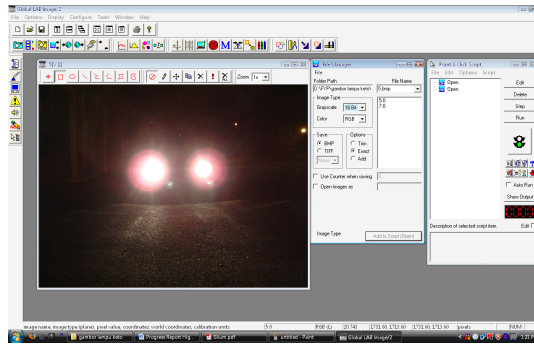
The proposed system also has been tested using Global Lab Image/2 as the simulation tool for the image processing process. The picture of an actual car's light consist of low beam and high beam was taken during night without traffic and others light source such as reflection from road sign. The simulation will be run based on the designed program structure in C++ language. As mention in the methodology part, the process start with the acquiring of the car's light. The process of the simulation will be described below.

- Image Acquisition

In simulation, the image is imported manually from the local folder of the host computer. The simulation will use BMP as the file format for the image taken. By using File Manager Tool which allows you to open image from local folder of host computer into the software. The sequences of the simulation start with the importing the reference image. The reference image will be the image that will be use to compare with the processed image. Then, using the File Manager Tool again, import the second image which is the capture image.



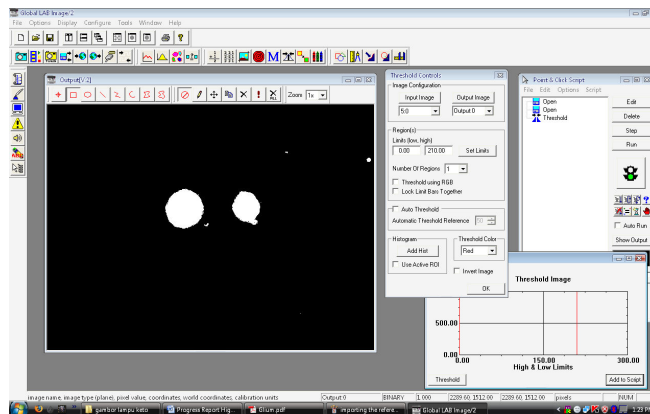
**Figure 9** Importing the reference image



**Figure 10** Importing the capture image (high beam)

- Threshold

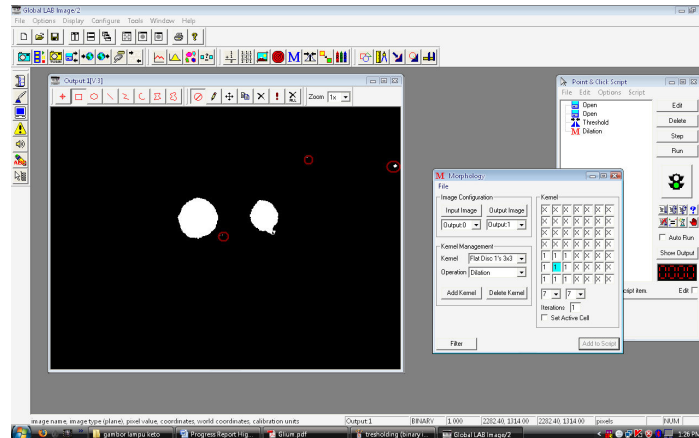
The Threshold Tools is use to convert a RGB image to a binary image. A binary image is an image with all pixel values set to either 1 (foreground/black) or 0 (background/white). The software provide the user to threshold the image by manually or automatically ways. The author has decided to choose manual threshold by specifying the high and low threshold values in the threshold display graph or in the Threshold Control dialog box. By manually setting the value of the threshold, the author can easily set the value as it C++ requires the fixed value of the threshold limit. So, by choosing manually the author can find a suitable value and set it as fixed limit values of threshold.



**Figure 11** Binary image conversion using Threshold Tool

- Morphology

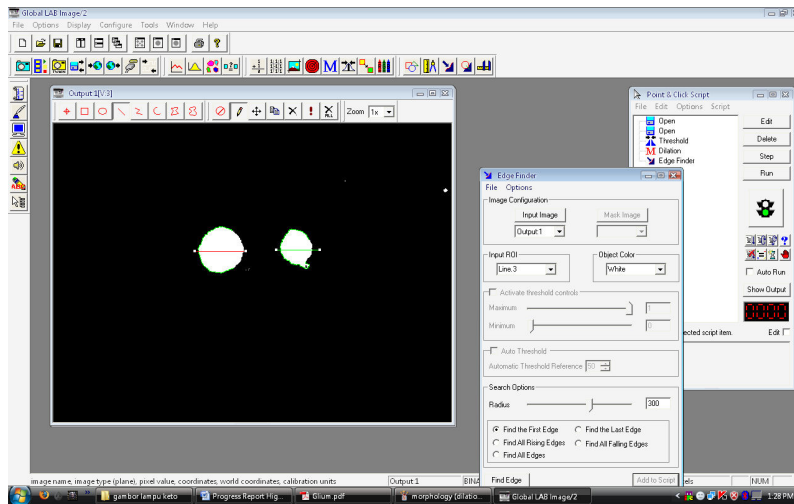
The binary image will be filter using morphology techniques. The purpose of this step is to provide us with clearly defined picture. This software consists of six different of morphological operations to perform on binary images. There are Erosion, Dilation, Opening, Closing, Skeletonization, and Watershed. Different types of morphological operations give different of result. As for the simulation, Erosion will choose as the operation for the resulting binary image. Erosion will erode a foreground object by removing pixels that touch the background. This tends to shrink and smooth foreground particles. It is the opposite of Dilation. The improvement of the image can be seen clearly before and after the process. The uninvolved white particles are removed denoted by the red circle.



**Figure 12** Morphology filtering using Dilataion method

- Edge Finder

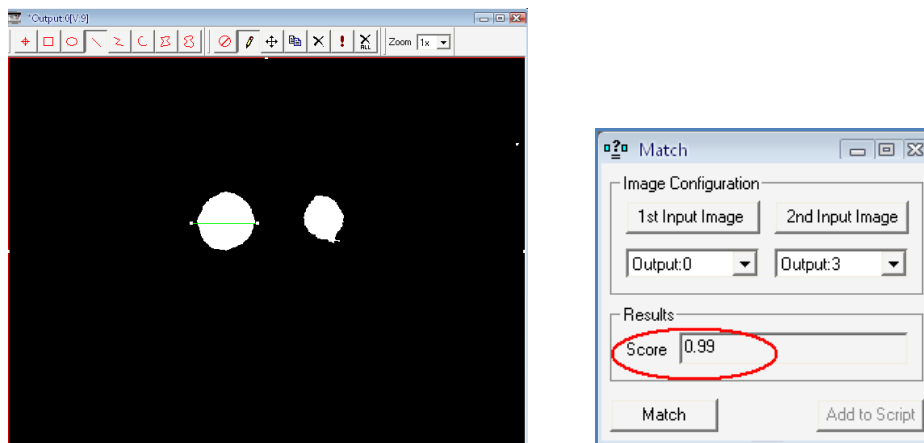
Laplacian Filter commonly used to detect edges within an image. But for the Global Lab Image/2, Laplacian Filtering cannot be used a method in order to detect edge because the sequence of this process involved binary image. The Filter Tool consist of kernel did not compatible with binary image. The simulation of the filtering is replaced with the Edge Finder Tools which it allows the author to extract points, edges, or contours from a binary image. Region of Interest (ROI) lines are used in order to find the edges.



**Figure 13** Edge detection using Edge Finder Tool

- Matching

The final step of the simulation is the matching of the processed image with the reference output image. This is the recognition part when the actual process is happened during C++ program. The Match Tool will compare two input images which is the Input 1 as the reference image and Input 2 as the result of current image processing process. The result of the comparison based on scoring points which 1 means actually the same but in this case the light source will be affected by other source light such as road's light. So, the scoring will be set within in a range such from 0.95 till 1.0 is the accepted values to trigger the circuit. The scoring will be input to trigger the automotive circuit.

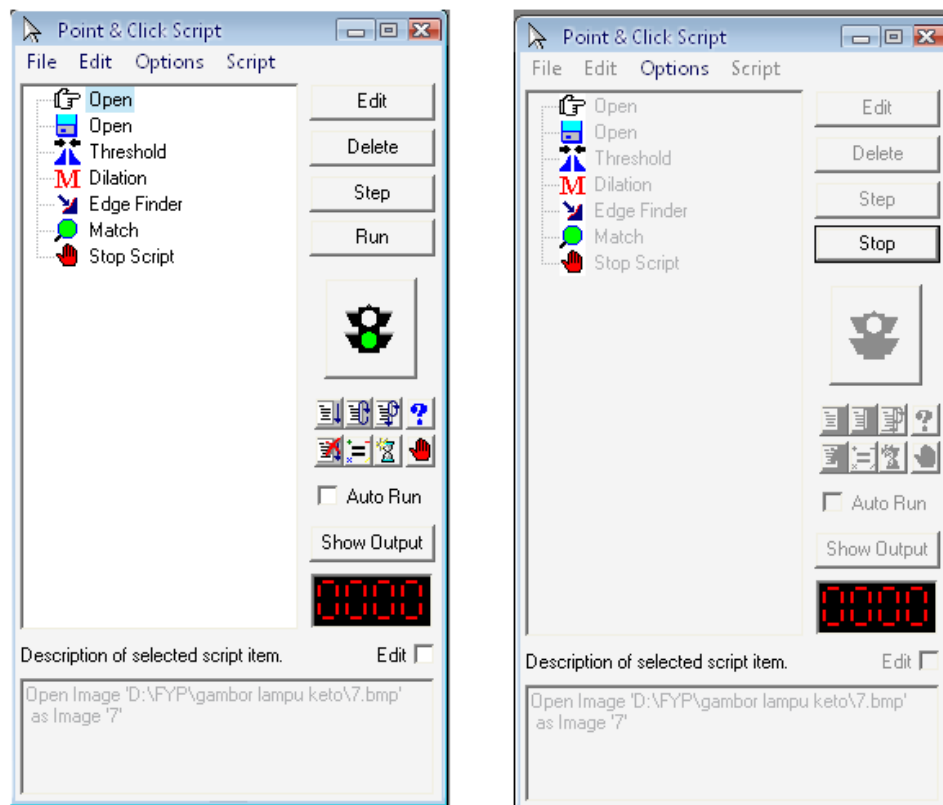


**Figure 14** Comparison of the images using Match Tool



- Finishing and Debugging The Script

Finally, to finish the program, use the STOP script by clicking it at the selected point. This program will be stop after matching stage. A stop script item stops the script when encountered during a script run and is typically used an error branch to stop execution of a script in an error condition. Save the current script setting by choosing the File in the script menu bar and click Save As Script. Then it will be the complete script of the program. To run the script again to show the simulation, click at the beginning of the script item which in this case is the open using File Manager Tool. The selected item will be confirmed by showing the hand (show) icon. After that, click Run.



**Figure 15** Complete script (left) and Debugging Process (right)

#### 4.1.2 C++ Testing and Results

The result obtained from the project is presented in four different sections which are samples, the program developed, and the output image after processed.

##### a) Samples

6 samples of car's light distribution images are processed using the above program. The samples are listed as follows:

**Table 1** List of car's light distribution

| No. | Car Brand                | Mode                 |
|-----|--------------------------|----------------------|
| 1   | Brand A (Proton Wira)    | Low Beam & High Beam |
| 2   | Brand B (Proton Satria)  | Low Beam & High Beam |
| 3   | Brand C (Perodua Kancil) | Low Beam & High Beam |

Table 1 shows the list of the different type of brand car's light distribution tested in this project. The name of the car's brand is kept confidential to follow the car company policy. The image of samples is viewed from the front car. For each brand, image was taken for in two light's mode, low beam and high beam.

##### b) Program Developed

The algorithm in the previous chapter is realized by using Microsoft Visual C++ software. The program is described below:

```

#include "Cimg.h"
using namespace cimg_library;
int main()
{
    printf("*****\n\r");
    printf("      THIS PROGRAM WILL BE ABLE TO IDENTIFY LOW BEAM OR HIGH BEAM *\n\r");
    printf("      USING IMAGE PROCESSING TECHNIQUES *\n\r");
    printf("*****\n\r");
    printf("\n\n");

    //(1)loading the image
    CImg<float> img("input.bmp"); // Load your image file.
    const int x=100, y=50;
    const float R = img(x,y,0), G = img(x,y,1), B = img(x,y,2);
    img.save("input_image.bmp");
    CImgDisplay displ(img,"Input Image");

    //(2)binary conversion
    CImg<>binary = img.norm_pointwise().normalize(0,255);
    binary.threshold(200); //set the value here
    binary.save("binary_image_result.bmp");
    ;

    //(3)erosion process
    //CImg<> img("lowbeam_binary_image.bmp");
    CImg<>erosion=binary.erode(3);
    erosion.save("erosion_image_result.bmp");

    //(4)edge detection
    const float sigma = 1; // smoothing strength
    CImg<> gx = erosion.get_deriche(sigma,1,'x');
    CImg<> gy = erosion.get_deriche(sigma,1,'y');
    CImg<> grad = ( gx.get_pow(2) + gy.get_pow(2) ).sqrt();
    grad.save("edge_detection_image_result.bmp");
    grad.display("Edge Detection");

    CImg<>list[4];
    list[0] = CImg<>("lowbeam.bmp");
    list[1] = CImg<>("lowbeam_binary_image.bmp");
    list[2] = CImg<>("lowbeam_dilatation_image.bmp");
    list[2] = CImg<>("lowbeam_edge_detection.bmp");

    return 0;
}

```

**Figure 16** The algorithm using Microsoft Visual C++




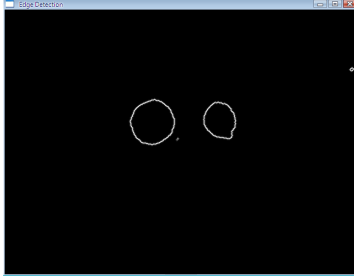
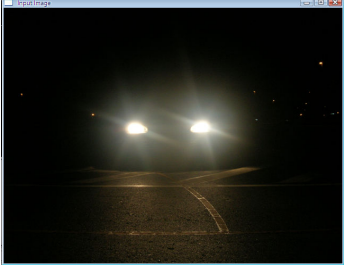
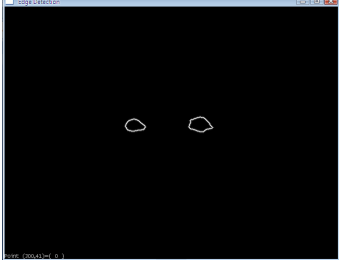
The program shown in figure above was arranged according to the step discussed in the previous chapter. Every sample in BMP format is processed using the above program.

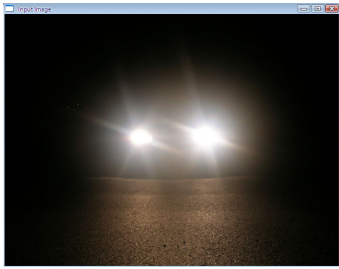
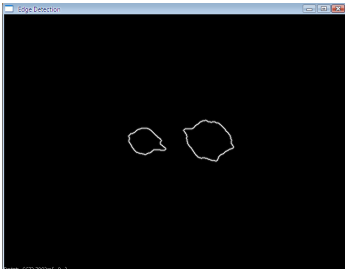


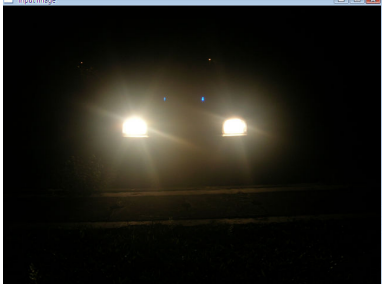

Cimg.h header file can be referred to Appendix C.

c) **Result of Processed Image**

Table 1.2 shows the final output image of each brand car's light distribution for both low beam and high beam. Original image is show together with the output image in order to differentiate before and after the image processing process.

**Table 2** Output Image

| Brand   | Lighting Mode | Original Image  | Output Image  |
|---------|---------------|---|---|
| Brand A | Low Beam      |   |    |
|         | High Beam     |  |  |
| Brand B | Low Beam      |  |  |

|         |           |  |   |
|---------|-----------|--|---|
|         | High Beam |   |  |
| Brand C | Low Beam  |   |  |
|         | High Beam |  |  |

From the table above, the input image or the original image and the output after processed by the proposed method is displayed. The output image show the interested area is marked by the white edges.

## 4.2 Discussion

### 4.2.1 Simulation

The simulation using MATLAB and Global Lab Image/2 will be the preliminary step to show the actual process of the image processing techniques. The simulation is executed based on the proposed system that will be using in the actual application. Both simulations would be a good reference for the author to improve the program design. After comparing both methods of simulation, Global Lab Image/2 is found to be good simulation software than MATLAB. It is because the time taken to process an image using Global Lab image/2 is more faster and accurate compare to MATLAB.

MATLAB also did not provide the matching or comparing image process which can compare the process image with the reference image like Global Lab Image/2 did. Besides that, MATLAB cannot simulate directly like Global Lab Image/2 which provides the user a program script to run the structure program. MATLAB can compare an image with another but have limited features such as it only can compare within a small set of array. As an example an image consist of [3x3] array. MATLAB can easily trace the difference in each array. But using the 24-bit image, the display of the array in MATLAB is out of range. User cannot interpret the differences among the array.

During simulation using Global Lab Image, the output image is compared with the reference image (high beam) to produce a score. This score is a value within a range set by the user to trigger the lamp circuit. As in the simulation the range of the value is set from 0 to 1. The score will trigger the circuit if the score is above 0.9 which means the input image is same as the reference image.


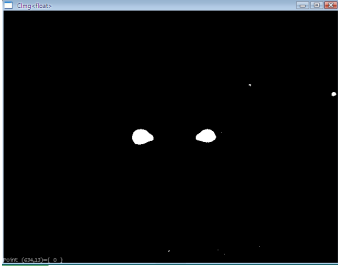
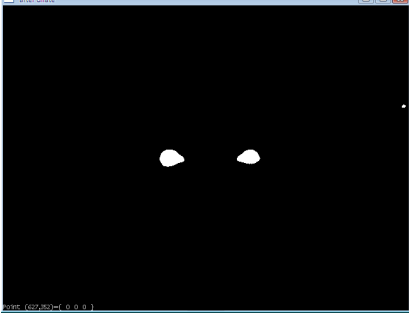

### 4.2.2 Samples

A digital camera used to capture the input image (samples) during low beam mode and high beam mode. The digital camera will be replaced by the web cam as the capture device in order to reduce the cost and space saving. Besides, digital camera saved the capture image in the memory card while web cam can save image directly to the host computer.

### 4.2.3 Methodology of the Project

The system basically showed the input and the output image after processed. Table below shows the step by step image by applying the image processing techniques proposed in the system. The sample of Brand A is taken for the discussion.

**Table 3** The step by step of the image processing method

|   |  |
|---|--|
|   |   |
| a) Input Image  | b) Thresholding Image  |
|  |  |
| c) Erosion Image  | d) Edge Detection Image  |

### **Figure (a): Read Input Image**

The input image is read from its file by using the CImg image loading command. CImg is the open-source C++ toolkit for image processing. It consist in a single header file 'CImg.h' providing a set of C++ classes and functions that can be used in the author's sources, to load/save, process and display image. It is very portable (Unix/X11, Windows, MacOS X, FreeBSD), efficient, easy to use, it's a pleasant toolkit for coding image processing stuffs in C++.

The type of the file must be specified and in this project the BMP file format used. Its sometimes called bitmap or DIB file format (for *device-independent bitmap*), is an image file format used to store bitmap digital images, especially on Microsoft Windows and OS/2 operating systems. Both input and output image is stored in BMP format. The input image will be loading manually from local folder. Assuming the capture device sent the capture image to the selected local folder.

### **Figure (b): Thresholding Image**

The input image is converted from RGB image to binary image. A binary image is an image with all pixel values set to either 1 (white) or 0 (black). The choice of threshold level is set to 200 as show in the main program. If the pixel value lower than the specified threshold value (200) is converted to 0 (black) and the higher value than that is converted to 1 (white).

The threshold value should be fixed to avoid much difference in region of interest after the processed is done. As an example if the value is set too small, the binary image from low beam RGB will contain more white pixels and maybe the program will detected as high beam binary image. In the end, the final output image will be interpreted by the program is the high beam mode.



**Figure (c): Erosion Image**

The binary image will be filter using morphology techniques. The purpose of this step is to provide us with clearly defined picture. This software consists of six different of morphological operations to perform on binary images. There are Erosion, Dilation, Opening, Closing, Skeletonization, and Watershed. Different types of morphological operations give different of result.

By perform the erosion to the image; the pixel value at the centre of the kernel is replaced by the minimum value of the neighborhood pixels. This operation reduces regions of high signal intensity in the digitized image. By observed the result before and after this step, the image show the improvement which it eliminates the uninvolved white particle.

**Figure (d): Edge Detection Image**

Finding edges of the detected objects in an image is considered to be an important process in many artificial visions systems. But after complicated analyzing from the views of accuracy and speed, some questions are retained more or less with all of the edge detectors. In my study, while both the cost of computations and the accuracy of detection about the objects are matter of primary importance, the author had detected the image using the Canny-Deriche Filter.

The selection of this filter is made because it can filter out high frequency noise and pixelization from the image by linking adjacent edges into long, smooth, continuous contours. This allows the edge map to reflect the dominant structures in the image. Besides, it is an optimal edge detector, in term of having probability of false or missing edges and in producing accurate edge position.

#### ***4.2.4 Evaluation of Results***

The proposed system has been tested on images which are captured through digital camera as discussed above. Each of pictures is taken from the constant parameters of the camera and the surrounding. The test and stored images are taken under similar lighting, distance, and background. Table 2 shows that almost most of the tested image is successfully verified using the proposed system. Currently the proposed system only available until matching processed only which the score from the matching will be using to or not to trigger the light circuit.

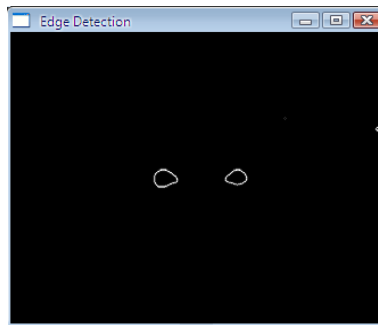
As referred to the Table 2, the output image of the low beam mode and high beam mode for three different car's manufacturers after being processed by the proposed system shows slight difference from each other using the comparison of same lighting mode. The slight difference came from the different light distribution from the variety design of headlamp. But it is clearly can be said that, if one of the output images is taking as a reference for comparison to get a score point, the segmentation or recognition process can be done. This can be shown from the all output images produced to have almost same area of white marked edges.

In this project, the area of the white marked edges is known as the region of interest. This region of interest can be use as the numerical analysis to be included in the proposed system. Basically, the area contain white edges in low beam output image is smaller in term of pixel size compare to the high beam mode. This pixels value can be used in the C++ main program as the factor to determine whether it's a low beam or high beam. The proposed system should calculate the black and white pixels in the binary image (output image) and this can value can be used as a range of comparison in order to trigger the automotive circuit. But in this project, this procedure is still in progress because of some unfortunate circumstances. Further studies on area calculation need to done in order to produce more accurate result and to complete the whole system automatically without the human supervision.

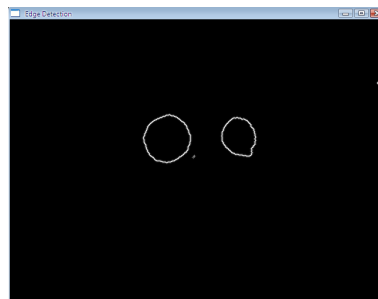
The system has been found to work faster as compared to previous system which used the LDR to detect approaching vehicle. This is because LDR detect any existence of light directly. Even a small portion of light will be detected by a LDR depending on its resistance value. Imagine the system using LDR; even torchlight can trigger the system to turn on the automotive light circuit.

In case of processing time, the system only take approximately 5 seconds to complete the process. This processing time is faster compared to previous LDR system because of the simple main program which required a small of time to perform the whole process. But that time is still not suitable for application in case of detecting high-speed car. The studies on the processing time should be taking into consideration when improving the system.

The proposed system was applied to 3 car manufacturers. The resultant image obtained was compared manually in term interested area size from acquisition of the image till edge detection using a reference image, one for the low beam, and one for the high beam. The reference image is shown below.



**Figure 17** Low Beam Reference Image



**Figure 18** High Beam Reference Image

For the time being, the result is evaluated using visual inspection. If the system is complete, evaluation based on the white pixels value need to be carried out to determine the system accuracy and effectiveness. Classifying image s into three different groups as described; provides a conclusion summarized in Table 4.

**Below Expectation:** The output image from proposed system applied completely did not have any similarities as compared with the reference image that is visually identified by normal human supervision during the visual inspection.

**Meet Expectation:** The output images from the proposed system applied have slightly different as compared with the reference image that is visually identified by normal human supervision during the visual inspection.

**Above Expectation:** The output images from the proposed system applied have similarities as compared with the reference image that is visually identified by normal human supervision during the visual inspection.

**Table 4** The Evaluation of the Result

| <b>Brand</b> | <b>Lighting Mode</b> | <b>Evaluation</b> |
|--------------|----------------------|-------------------|
| <b>A</b>     | Low Beam             | Above Expectation |
|              | High Beam            | Above Expectation |
| <b>B</b>     | Low Beam             | Above Expectation |
|              | High Beam            | Meet Expectation  |
| <b>C</b>     | Low Beam             | Above Expectation |
|              | High Beam            | Meet Expectation  |

From the Table 4, it shows that there are no output images that is classified under the below expectation, 4 are above expectation and 2 had recorded to be meet expectation. From the evaluation above, it can be said that this intelligent system can automatically determined the lighting mode and can be applied in the car to perform automatic light circuit switching scheme with further modifications and

improvements on comparison stage. The comparison stage would provide the output to be used for controlling the light beam.

#### ***4.2.5 Limitations***

Currently, the proposed system still has several limitations that have yet to be overcome. The simulation only shows the actual ways how the system will work. Limitations of the system are explained below:

- i. Due to the inconsistency of clarifying the light pattern, the detection should be set at certain distance to avoid errors. The distance must be choosing corresponding to the system setup by the author. This will give a good solution in order for the system to detect and compare the light.
- ii. The proposed system would be able to process the image that have been taken during no other's light source such as road's light, light reflection on the sign board, and light from the building or houses. The image of the car light is taken at a completely dark area.
- iii. The system also unable to capture the high speed car's light. In the simulation, the used image is taken from an unmoving car. The light distribution at a high speed will affect the image processing.
- iv. The proposed system would be able to recognize shapes of light pattern from in front view only. The system is unable to recognize the light patterns from other views.

## **CHAPTER 5**

### **CONCLUSION AND RECOMMENDATIONS**

#### **5.1 Conclusions**

Overall, the principal objective of this project has been achieved within the time frame given. The author has successfully developed a car's light pattern detection which would be able to recognize a car's low beam or high beam. The sample of different brand car's light distribution was able to be digitized and processed using the available image processing software and the C++ program. The method that is suitable to apply images in getting a clearer image to show the lighting mode is recognized.

This report provides an overview on how the research on the final year project progressed throughout the two semesters. It serves as a framework, which will enhance the student's skill in the process of applying knowledge, expanding thoughts, solving problems independently, and presenting findings through minimum guidance and supervision.

Throughout two semesters, the project has exhibited a tremendous growth. A multitude of algorithms from preprocessing, data reductions to feature analysis have been presented. This project not only allows the author to explore computer imaging concepts, at the same time provides the opportunity for the author to acquire skills in C++ programming.

To apply the image processing concepts to an image, it is necessary for a detailed understanding of the algorithms described. In some programs, there are relatively simple math involved, but more advanced methods require justification on

the algorithms. The author has to select references that provide a range of approaches, from detailed and complex mathematical analysis to clear and concise exposition. This has let the author gain insight into various computer imaging algorithms and concepts when developing the required processing tools.

## **5.2 Recommendations and Future Works**

The author would like to suggest few recommendations for future enhancement. The focus can be placed on the limitations of the current proposed system. Limitations of the proposed system still encourage the author to find the necessary algorithms that would provide fully automated vehicle identification system. Areas which still need improvements in the future are:

- Final touch of the proposed system which to tell the user whether the processed image is low beam or high beam using GUI.
- The system can recognize the light pattern like the speed of normal human eye when driving. Hopefully, some day the system work even faster to process the image.
- Successfully integrated with hardware part. The proposed system work fully automatically without human handling and can trigger the light circuit to suitable light mode (low beam or high beam) according to the situation.
- The proposed system has good recognition process even with high speed moving vehicle and can detect all type of vehicles such as lorry, bus, and van.
- The proposed system should be able to work in various types of environments and conditions around vehicle such as in accommodation area where other type of lights affect the previous system.
- Modification need to be done on the current system which enables to calculate the numbers of white pixel to be used in the comparison stage in order to control the light circuit. The circuit should be trigger or not to trigger based on the pixel values set by the author in the main program. As an example, the complete system will trigger the automotive circuit with the condition of both car using this system and in the high beam mode, if the number of white pixel consist in the output image in the range of 80-100, the system will defined the output image as high beam and score 1 will be produce by the system to

trigger the circuit on to change from high beam mode to low beam mode. If the number of white pixel detected by the system less than 40-80, the system will defined the output image as low beam mode and score 1 will be produce by the system to trigger the circuit on to change from high beam mode to low beam mode. So, this mean that, in that condition, if the system detects low beam and high beam from approaching vehicle automotive circuit will triggered. But, when the number of white pixel is less than 20, the system will defined the output image consist of no approaching vehicle or the vehicle is at the dark surrounding. This will produce a 0 score which tell the system to not trigger the circuit and the car is remains using high beam mode on the road.

- Instead of using erosion to remove the noise especially uninvolved white particle in the binary image, opening and closing operation is the most suitable in the morphological noise removal. Opening removes small objects, while closing removes small holes. The main program of the erosion part shall be replaced with the opening and closing operation.
- Besides, the main program should be added with a region of interest step after the conversion to binary image step and before the morphological filtering. The purpose of this new step is to cut and take only the interest region of the image only. At the end of the processing stage, the system will give a better output image because uninvolved white pixels or noise is removed at the early stage of the processing steps.



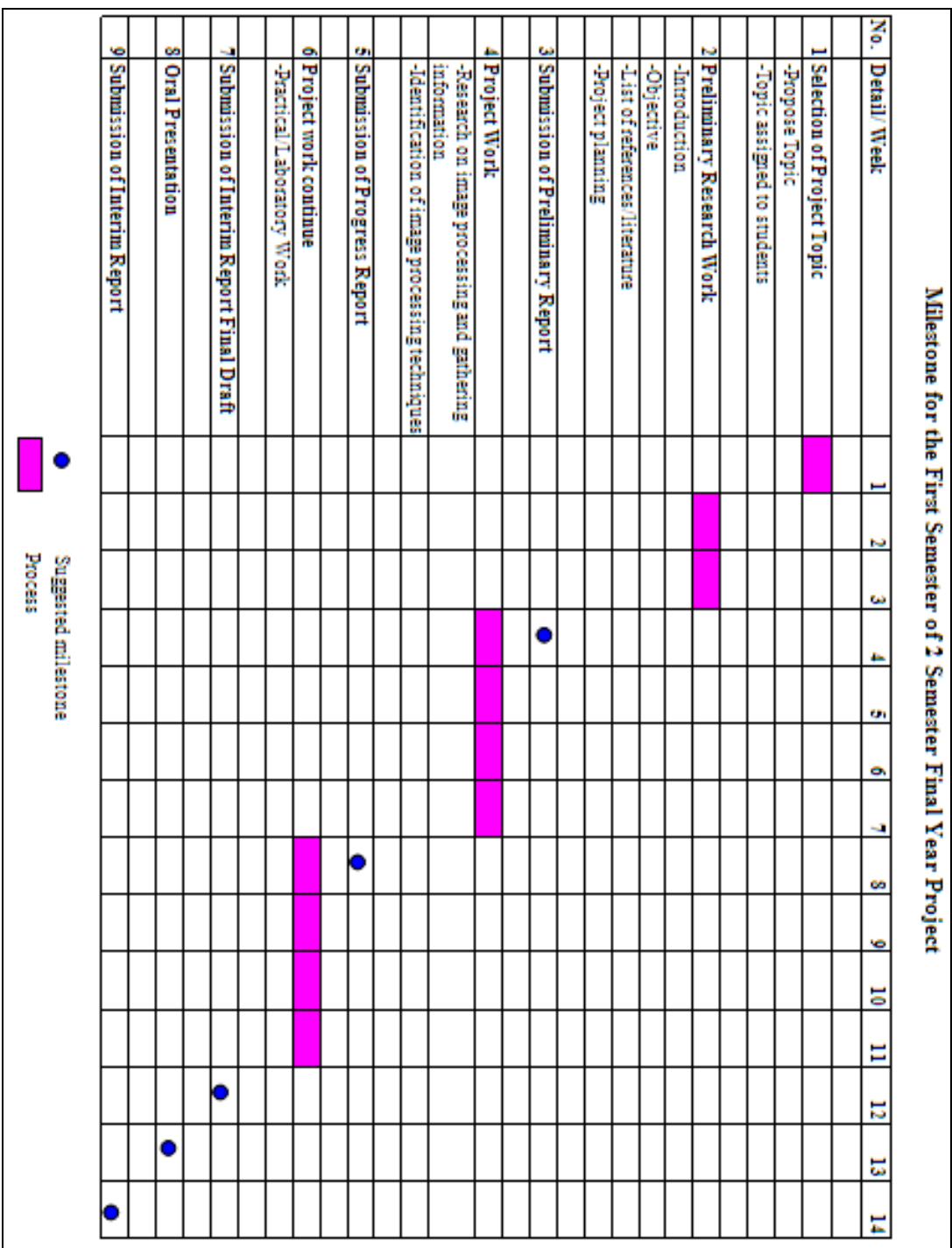
## REFERENCES

- [1] Transportation Engineers and Highway Planners for the Government of Malaysia, “Road Safety Section,” <http://www.kkr.gov.my/bpj/safety/safety1.htm>.
- [2] Md. Khairul Amri Bin Nordin, “Final Report”, Industrial Internship Training Programme, Universiti Teknologi Petronas, Jan 2007, pg 14-16.
- [3] Nor Sulyanie bt Sulong, “High Beam Controller for Vehicles”, Degree Thesis, Universiti Teknologi Petronas, Tronoh, 2005.
- [4] Chew Siew Kheng, “Vehicle Identification Using Image Processing Techniques”, Degree Thesis, Universiti Teknologi Petronas, Tronoh, 2002.
- [5] ‘Wikipedia, the Free Encyclopaedia’, Wikimedia Foundation, Inc. [http://en.wikipedia.org/wiki/Programmable\\_logic\\_controller](http://en.wikipedia.org/wiki/Programmable_logic_controller)
- [6] Barry Dorr, “Autronic Eye,” [http://www.dorrengeering.com/autronic\\_eye.htm](http://www.dorrengeering.com/autronic_eye.htm).
- [7] Bruce Batchelor and Federick Waltz, 2001, *Intelligent Machine Vision. Techniques, Implementations and Applications*, Springer.
- [8] Umi Kalthum Ngah, 1995, *An Expert System With Integrated Image Processing Features For The Diagnoses of Breast Diseases*, Thesis for Master.
- [9] ‘What Is MATLAB’, The MathWorks, Inc. <http://www.mathworks.com/>
- [10] James M. SLack, 2000, *Programming and Problem Solving with JAVA*, Brooks/Cole.
- [11] Howard E. Burdick, 1997, *Digital Imaging: Theory and Applications*, McGrawHill.

- [12] 'C++ Template Image Processing Library', 2004, David Tschumperlé,  
<http://cimg.souceforge.net/>
  
- [13] 'Visual C++ and C++ Programming Forum', 2000, Jupitermedia Corporation,  
<http://www.codeguru.com/forum>
  
- [14] Kenneth R. Castleman, 1996, *Digital Image Processing*, Prentice Hall.
  
- [15] Khairul Nisak Md Hasan, "Detection of Microcalcification Using Mammograms", Degree Thesis, Universiti Teknologi Petronas, Tronoh, 2004.

## APPENDICES

## APPENDIX A GANTT CHART



### Milestone for the Second Semester of 2 Semester Final Year Project

| No. Detail Week                                | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |  |
|--|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|--|
| <b>1 Development of Program</b>                |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |  |
| - Designation of stages                        |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |  |
| - Thresholding                                 |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |  |
| - Image enhancement                            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |  |
| - Segmentation                                 |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |  |
| - Feature extraction and recognition           |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |  |
| <b>2 Evaluation of Result</b>                  |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |  |
| - Analysis                                     |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |  |
| <b>3 Submission of First Progress Report</b>   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |  |
| <b>4 Submission of Second Progress Report</b>  |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |  |
| <b>5 Pre-EDV</b>                               |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |  |
| <b>6 Submission of Draft Report</b>            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |  |
| Submission of Final Report                     |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |  |
| Submission of Technical Report                 |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |  |
| <b>8 Final Exam</b>                            |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |  |
| <b>9 Oral Presentation</b>                     |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |  |
| <b>10 Submission of Hardbound Dissertation</b> |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |  |

 Suggested milestones  
 Process

## APPENDIX B

### C++ MAIN PROGRAM

```
#include "CImg.h"
using namespace cimg_library;

int main()
{
    printf("*****\n\r");
    printf("* THIS PROGRAM WILL BE ABLE TO IDENTIFY LOW BEAM OR HIGH BEAM *\n\r");
    printf("          USING IMAGE PROCESSING TECHNIQUES *\n\r");
    printf("*****\n\r");
    printf("\n\n");

    //(1)loading the image
    CImg<float> img("input.bmp"); // Load your image file.
    const int x=100, y=50;
    const float R = img(x,y,0), G = img(x,y,1), B = img(x,y,2);
    img.save("input_image.bmp");
    CImgDisplay disp1(img,"Input Image");

    //(2)binary conversion
    CImg<>binary = img.norm_pointwise().normalize(0,255);
    binary.threshold(200); //set the value here
    binary.save("binary_image_result.bmp");

    //(3)erosion process
    CImg<>erosion=binary.erode(3);
    erosion.save("dilatation_image_result.bmp");

    //(4)edge detection
    const float sigma = 1; // smoothing strength
    CImg<> gx = erosion.get_deriche(sigma,1,'x');
    CImg<> gy = erosion.get_deriche(sigma,1,'y');
    CImg<> grad = ( gx.get_pow(2) + gy.get_pow(2) ).sqrt();
    grad.save("edge_detection_image_result.bmp");
    grad.display("Edge Detection");

    return 0;
}
```

## APPENDIX C

### CIMG.H (IMAGE HEADER FILE) [12]

```
#ifndef cimg_version
#define cimg_version 1.24

// Detect Microsoft VC++ 6.0 compiler to get some workarounds afterwards.
#if defined(_MSC_VER) && _MSC_VER<1300
#define cimg_use_visualcpp6
#endif

// Avoid strange 'deprecated' warning messages with Visual C++ .NET.
#if defined(_MSC_VER) && _MSC_VER>=1300
#define _CRT_SECURE_NO_DEPRECATED 1
#define _CRT_NONSTDC_NO_DEPRECATED 1
#endif

// Standard C++ includes.
#include <cstdio>
#include <cstdlib>
#include <cstdlibarg>
#include <cstring>
#include <cmath>
#include <ctime>

/*
#
# Set CImg configuration flags.
#
# If compilation flags are not adapted to your system,
# you may override their values, before including
# the header file "CImg.h" (use the #define directive).
#
*/

// Try to detect the current system and set value of 'cimg_OS'.
#ifndef cimg_OS
// Unix-like (Linux, Solaris, BSD, MacOSX, Irix,...).
#if defined(unix) || defined(__unix) || defined(__unix__) \
|| defined(linux) || defined(__linux) || defined(__linux__) \
|| defined(sun) || defined(__sun) \
|| defined(BSD) || defined(__OpenBSD__) || defined(__NetBSD__) || defined(__FreeBSD__) || defined
__DragonFly__ \
|| defined(__MACOSX__) || defined(__APPLE__) \
|| defined(sgi) || defined(__sgi) \
|| defined(__CYGWIN__)
#define cimg_OS 1
#endif
#ifndef cimg_display_type
#define cimg_display_type 1
#endif
#ifndef cimg_color_terminal
#define cimg_color_terminal
#endif
// Windows.
#elif defined(_MSC_VER) || defined(WIN32) || defined(_WIN32) || defined(__WIN32__) \
|| defined(WIN64) || defined(_WIN64) || defined(__WIN64__)
#define cimg_OS 2
#endif
#ifndef cimg_display_type
#define cimg_display_type 2
#endif
// Unknown configuration : ask for minimal dependencies (no display).
#else
#define cimg_OS 0
#endif
```

```

#ifndef cimg_display_type
#define cimg_display_type 0
#endif
#endif
#endif

// Debug configuration.
//
// Set 'cimg_debug' to : 0 to remove debug messages (exceptions are still thrown anyway).
//           1 to display debug messages on standard error output (console).
//           2 to display debug messages with modal windows (default behavior).
//           3 to do as 2 + add extra memory access warnings (may slow down the code)
#ifndef cimg_debug
#define cimg_debug 2
#endif

// Architecture-dependent includes.
#if cimg_OS==1
#include <sys/time.h>
#include <unistd.h>
#elif cimg_OS==2
#include <windows.h>
#ifndef _WIN32_IE
#define _WIN32_IE 0x0400
#endif
#include <shlobj.h>
#ifdef cimg_use_visualcpp6
#define std
#endif
#endif

// Test if min/max or PI macros are defined.
#ifdef PI
#error -----
#error The macro value 'PI' has been defined prior to the #include "CImg.h" directive.
#error The CImg Library does not compile with such a macro value defined.
#error Please (re)define this macro *after* including "CImg.h" if really necessary.
#error Following error messages are most probably related to this problem.
#error -----
#endif
#ifdef min
#undef min
#define cimg_redefine_min
#endif
#ifdef max
#undef max
#define cimg_redefine_max
#endif

// Display-dependent includes.
#if cimg_display_type==1
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/keysym.h>
#include <pthread.h>
#ifdef cimg_use_xshm
#include <sys/ipc.h>
#include <sys/shm.h>
#include <X11/extensions/XShm.h>
#endif
#ifdef cimg_use_xrandr
#include <X11/extensions/Xrandr.h>
#endif
#endif

// Configuration for using extra libraries
//
// Define 'cimg_use_png', 'cimg_use_jpeg' or 'cimg_use_tiff' to enable native PNG, JPEG or TIFF files support.

```



```

// This requires you link your code with the zlib/png, jpeg or tiff libraries.
// Without these libraries, PNG,JPEG and TIFF support will be done by the Image Magick's 'convert' tool,
// or byt the GraphicsMagick 'gm' tool if installed
// (this is the case on most unix plateforms).
#ifdef cimg_use_png
extern "C" {
#include "png.h"
}
#endif
#ifdef cimg_use_jpeg
extern "C" {
#include "jpeglib.h"
}
#endif
#ifdef cimg_use_tiff
extern "C" {
#include "tiffio.h"
}
#endif
#ifdef cimg_use_magick
#include "Magick++.h"
#endif
#ifdef cimg_use_fftw3
extern "C" {
#include "fftw3.h"
}
#endif
#ifdef cimg_use_lapack
extern "C" {
extern void sgetrf_(int*, int*, float*, int*, int*, int*);
extern void sgetri_(int*, float*, int*, int*, float*, int*, int*);
extern void sgets_(char*, int*, int*, float*, int*, float*, int*, int*);
extern void sgesvd_(char*, char*, int*, int*, float*, int*, float*, float*, int*, float*, int*, float*, int*, int*);
extern void ssyev_(char*, char*, int*, float*, int*, float*, float*, int*, int*);
extern void dgetrf_(int*, int*, double*, int*, int*, int*);
extern void dgetri_(int*, double*, int*, int*, double*, int*, int*);
extern void dgets_(char*, int*, int*, double*, int*, int*, double*, int*, int*);
extern void dgesvd_(char*, char*, int*, int*, double*, int*, double*, double*, int*, double*, int*, double*, int*,
int*);
extern void dsyev_(char*, char*, int*, double*, int*, double*, double*, int*, int*);
}
#endif

// Macros used to describe the program usage, and retrieve command line arguments
// (See corresponding module 'Retrieving command line arguments' in the generated documentation).
#define cimg_usage(usage) cimg_library::cimg::option((char*)0,argc,argv,(char*)0,usage)
#define cimg_help(str) cimg_library::cimg::option((char*)0,argc,argv,str,(char*)0)
#define cimg_option(name,default,usage) cimg_library::cimg::option(name,argc,argv,default,usage)

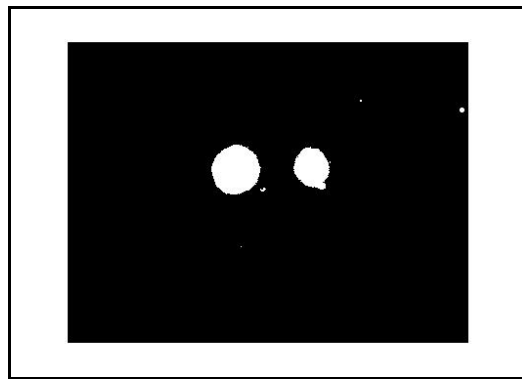
// Macros used for neighborhood definitions and manipulations.
// (see module 'Using Image Loops' in the generated documentation).
#define CImg_2(I,T) T I##cc,I##nc=0
#define CImg_2x2(I,T) T I##cc,I##nc=0,I##cn,I##nn=0
#define CImg_3(I,T) T I##pp,I##cp,I##np=0
#define CImg_3x3(I,T) T I##pp,I##cp,I##np=0,I##pc,I##cc,I##nc=0,I##pn,I##cn,I##nn=0
#define CImg_4(I,T) T I##pp,I##cp,I##np=0,I##ap=0
#define CImg_4x4(I,T) T I##pp,I##cp,I##np=0,I##ap=0, \
I##pc,I##cc,I##nc=0,I##ac=0, \
I##pn,I##cn,I##nn=0,I##an=0, \
I##pa,I##ca,I##na=0,I##aa=0
*****CONTINUE

```

**APPENDIX D**  
**HIGH BEAM IMAGE PROCESSING USING MATLAB**  
**(SIMULATION)**



**Figure 19** Original Image of High Beam



**Figure 20** Grayscale Conversion

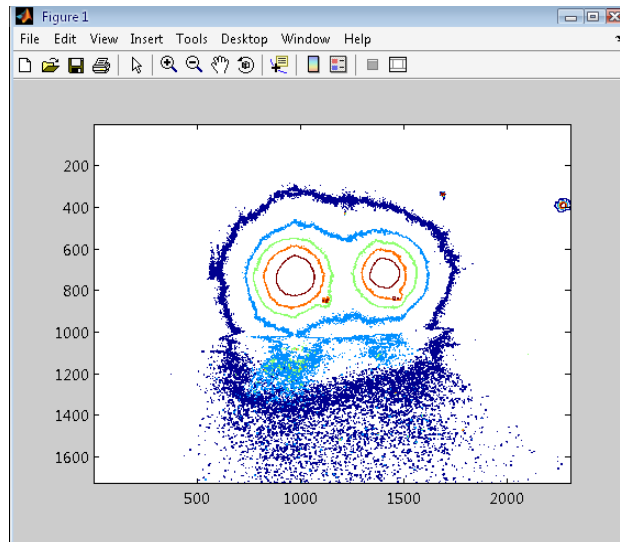


**Figure 21** Edge Detection Process

**APPENDIX E**  
**LIGHT INTENSITY AND IMAGE CONTOUR FOR HIGH BEAM**  
**AND LOW BEAM USING MATLAB**



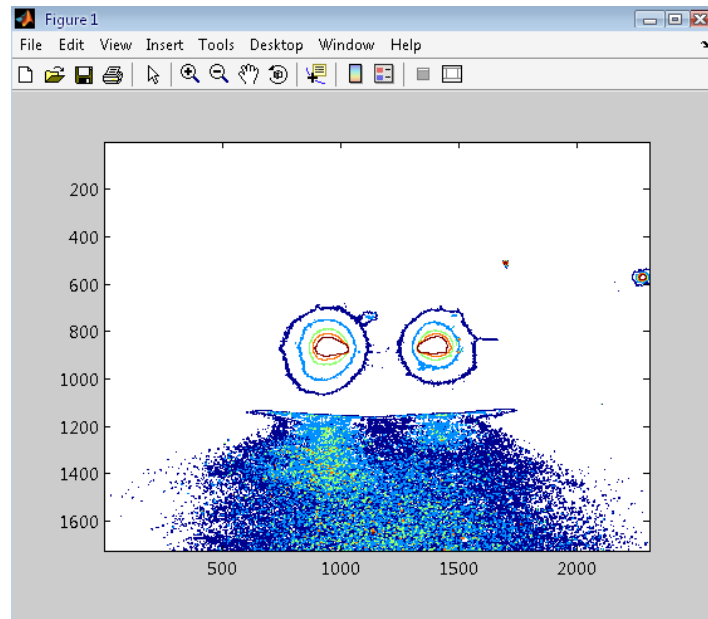
**Figure 22** High beam of incoming traffic at 5 m



**Figure 23** High beam of incoming traffic at 5 m after processed



**Figure 24** Low beam of incoming traffic at 5 m



**Figure 25** Low beam of incoming traffic at 5 m after processed

## APPENDIX F

### MATLAB SOURCE CODE

- Converting RGB to Grayscale Image and processing using edge detector's command

```
X = imread ('D:\FYP\gambor lampu keto\DSCF0001.JPG') %import image from
directory%
imshow (X) %show the original image%
L=im2bw (X,0.8) %convert image RGB to grayscale%
figure,imshow (L) %show the grayscale image%
L1=edge (L,'sobel') %process the image using 'sobel' edge detector's method%
figure,imshow (L1) %show the results image%
L2=edge (L,'canny') %process the image using 'canny' edge detector's method%
figure,imshow (L1) %show the results image%
```

- Contour analysis

```
RGB = imread('D:\FYP\gambor lampu keto\DSCF0001.JPG'); %import image
from directory%
gray1 = rgb2gray (RGB); %convert image RGB to grayscale%
imcontour(gray1) %analysis the contour's image%
imshow (gray1) %show the results image%
```

# APPENDIX G

## MATLAB IMAGE PROCESSING TOOLBOX

Version 3.1 (R12.1) 18-May-2001

### Release information.

images/Readme - Display information about current and previous versions.

### Image display.

colorbar - Display colorbar (MATLAB Toolbox).  
getimage - Get image data from axes.  
image - Create and display image object (MATLAB Toolbox).  
imagesc - Scale data and display as image (MATLAB Toolbox).  
immovie - Make movie from multiframe image.  
imshow - Display image.  
montage - Display multiple image frames as rectangular montage.  
movie - Play recorded movie frames.  
subimage - Display multiple images in single figure.  
truecolor - Adjust display size of image.  
warped - Display image as texture-mapped surface.

### Image file I/O.

dicominfo - Read metadata from a DICOM message.  
dicomread - Read a DICOM image.  
dicomdict.txt - Text file containing DICOM data dictionary.  
imfinfo - Return information about image file (MATLAB Toolbox).  
imread - Read image file (MATLAB Toolbox).  
imwrite - Write image file (MATLAB Toolbox).

### Image arithmetic.

imabsdiff - Compute absolute difference of two images.  
imadd - Add two images, or add constant to image.  
imcomplement - Complement image.  
imdivide - Divide two images, or divide image by constant.  
imlincomb - Compute linear combination of images.  
immultiply - Multiply two images, or multiply image by constant.  
imsubtract - Subtract two images, or subtract constant from image.

### Geometric transformations.

checkerboard - Create checkerboard image.  
findbounds - Find output bounds for geometric transformation.  
fliptform - Flip the input and output roles of a TFORM struct.  
imcrop - Crop image.  
imresize - Resize image.  
imrotate - Rotate image.  
intransform - Apply geometric transformation to image.  
makeresampler - Create resampler structure.  
maketform - Create geometric transformation structure (TFORM).  
tformarray - Apply geometric transformation to N-D array.  
tformfwd - Apply forward geometric transformation.  
tforminv - Apply inverse geometric transformation.

### Image registration.

cpstruct2pairs - Convert CPSTRUCT to valid pairs of control points.  
cp2tform - Infer geometric transformation from control point pairs.  
cpcorr - Tune control point locations using cross-correlation.  
cpselect - Control point selection tool.  
normxcorr2 - Normalized two-dimensional cross-correlation.

### Pixel values and statistics.

corr2 - Compute 2-D correlation coefficient.  
imcontour - Create contour plot of image data.  
imhist - Display histogram of image data.  
impixel - Determine pixel color values.  
improfile - Compute pixel-value cross-sections along line segments.  
mean2 - Compute mean of matrix elements.  
pixmap - Display information about image pixels.  
regionprops - Measure properties of image regions.  
std2 - Compute standard deviation of matrix elements.

### Image analysis.

edge - Find edges in intensity image.

qtdecomp - Perform quadtree decomposition.  
 qtgetblk - Get block values in quadtree decomposition.  
 qtsetblk - Set block values in quadtree decomposition.

**Image enhancement.**  
 histeq - Enhance contrast using histogram equalization.  
 imadjust - Adjust image intensity values or colormap.  
 imnoise - Add noise to an image.  
 medfilt2 - Perform 2-D median filtering.  
 ordfilt2 - Perform 2-D order-statistic filtering.  
 stretchlim - Find limits to contrast stretch an image.  
 wiener2 - Perform 2-D adaptive noise-removal filtering.

**Linear filtering.**  
 convmtx2 - Compute 2-D convolution matrix.  
 fspecial - Create predefined filters.  
 imfilter - Filter 2-D and N-D images.

**Linear 2-D filter design.**  
 freqspace - Determine 2-D frequency response spacing (MATLAB Toolbox).  
 freqz2 - Compute 2-D frequency response.  
 fsamp2 - Design 2-D FIR filter using frequency sampling.  
 ftrans2 - Design 2-D FIR filter using frequency transformation.  
 fwind1 - Design 2-D FIR filter using 1-D window method.  
 fwind2 - Design 2-D FIR filter using 2-D window method.

**Image deblurring.**  
 deconvblind - Deblur image using blind deconvolution.  
 deconvlucy - Deblur image using Lucy-Richardson method.  
 deconvreg - Deblur image using regularized filter.  
 deconvwnr - Deblur image using Wiener filter.  
 edgetaper - Taper edges using point-spread function.  
 otftpsf - Optical transfer function to point-spread function.  
 psf2otf - Point-spread function to optical transfer function.

**Image transforms.**  
 dct2 - 2-D discrete cosine transform.  
 dctmtx - Discrete cosine transform matrix.  
 fft2 - 2-D fast Fourier transform (MATLAB Toolbox).  
 fftn - N-D fast Fourier transform (MATLAB Toolbox).  
 fftshift - Reverse quadrants of output of FFT (MATLAB Toolbox).  
 idct2 - 2-D inverse discrete cosine transform.  
 ifft2 - 2-D inverse fast Fourier transform (MATLAB Toolbox).  
 ifftn - N-D inverse fast Fourier transform (MATLAB Toolbox).  
 iradon - Compute inverse Radon transform.  
 phantom - Generate a head phantom image.  
 radon - Compute Radon transform.

**Neighborhood and block processing.**  
 bestblk - Choose block size for block processing.  
 blkproc - Implement distinct block processing for image.  
 col2im - Rearrange matrix columns into blocks.  
 colfilt - Columnwise neighborhood operations.  
 im2col - Rearrange image blocks into columns.  
 nlfilt - Perform general sliding-neighborhood operations.

**Morphological operations (intensity and binary images).**  
 conndef - Default connectivity.  
 imbothat - Perform bottom-hat filtering.  
 imclearborder - Suppress light structures connected to image border.  
 imclose - Close image.  
 imdilate - Dilate image.  
 imerode - Erode image.  
 imextendedmax - Extended-maxima transform.  
 imextendedmin - Extended-minima transform.  
 imfill - Fill image regions and holes.  
 imhmax - H-maxima transform.  
 imhmin - H-minima transform.  
 imimposemin - Impose minima.  
 imopen - Open image.  
 imreconstruct - Morphological reconstruction.  
 imregionalmax - Regional maxima.  
 imregionalmin - Regional minima.  
 imtophat - Perform tophat filtering.  
 watershed - Watershed transform.

**Morphological operations (binary images)**  
 applylut - Perform neighborhood operations using lookup tables.

**bwarea** - Compute area of objects in binary image.  
**bwareaopen** - Binary area open (remove small objects).  
**bwdist** - Compute distance transform of binary image.  
**bweuler** - Compute Euler number of binary image.  
**bwhitmiss** - Binary hit-miss operation.  
**bwlabel** - Label connected components in 2-D binary image.  
**bwlabeln** - Label connected components in N-D binary image.  
**bwmorph** - Perform morphological operations on binary image.  
**bwpack** - Pack binary image.  
**bwperim** - Determine perimeter of objects in binary image.  
**bwselect** - Select objects in binary image.  
**bwulterode** - Ultimate erosion.  
**bwunpack** - Unpack binary image.  
**makelut** - Construct lookup table for use with applylut.

**Structuring element (STREL) creation and manipulation.**  
**getheight** - Get strel height.  
**getneighbors** - Get offset location and height of strel neighbors  
**getnhood** - Get strel neighborhood.  
**getsequence** - Get sequence of decomposed strels.  
**isflat** - Return true for flat strels.  
**reflect** - Reflect strel about its center.  
**strel** - Create morphological structuring element.  
**translate** - Translate strel.

**Region-based processing.**  
**roicolor** - Select region of interest, based on color.  
**roifill** - Smoothly interpolate within arbitrary region.  
**roifilt2** - Filter a region of interest.  
**roipoly** - Select polygonal region of interest.

**Colormap manipulation.**  
**brighten** - Brighten or darken colormap (MATLAB Toolbox).  
**cmpermute** - Rearrange colors in colormap.  
**cmunique** - Find unique colormap colors and corresponding image.  
**colormap** - Set or get color lookup table (MATLAB Toolbox).  
**imapprox** - Approximate indexed image by one with fewer colors.  
**rgbplot** - Plot RGB colormap components (MATLAB Toolbox).

**Color space conversions.**  
**hsv2rgb** - Convert HSV values to RGB color space (MATLAB Toolbox).  
**ntsc2rgb** - Convert NTSC values to RGB color space.  
**rgb2hsv** - Convert RGB values to HSV color space (MATLAB Toolbox).  
**rgb2ntsc** - Convert RGB values to NTSC color space.  
**rgb2ycbcr** - Convert RGB values to YCBCR color space.  
**ycbcr2rgb** - Convert YCBCR values to RGB color space.

**Array operations.**  
**circshift** - Shift array circularly.  
**padarray** - Pad array.

**Image types and type conversions.**  
**dither** - Convert image using dithering.  
**gray2ind** - Convert intensity image to indexed image.  
**grayslice** - Create indexed image from intensity image by thresholding.  
**graythresh** - Compute global image threshold using Otsu's method.  
**im2bw** - Convert image to binary image by thresholding.  
**im2double** - Convert image array to double precision.  
**im2uint8** - Convert image array to 8-bit unsigned integers.  
**im2uint16** - Convert image array to 16-bit unsigned integers.  
**ind2gray** - Convert indexed image to intensity image.  
**im2mis** - Convert image to Java MemoryImageSource.  
**ind2rgb** - Convert indexed image to RGB image (MATLAB Toolbox).  
**isbw** - Return true for binary image.  
**isgray** - Return true for intensity image.  
**isind** - Return true for indexed image.  
**isrgb** - Return true for RGB image.  
**label2rgb** - Convert label matrix to RGB image.  
**mat2gray** - Convert matrix to intensity image.  
**rgb2gray** - Convert RGB image or colormap to grayscale.  
**rgb2ind** - Convert RGB image to indexed image.

**Toolbox preferences.**  
**iptgetpref** - Get value of Image Processing Toolbox preference.  
**iptsetpref** - Set value of Image Processing Toolbox preference.

**Demos.**  
**dctdemo** - 2-D DCT image compression demo.



edgedemo - Edge detection demo.  
 firdemo - 2-D FIR filtering and filter design demo.  
 imadjdemo - Intensity adjustment and histogram equalization demo.  
 landsatdemo - Landsat color composite demo.  
 nrfiltdemo - Noise reduction filtering demo.  
 qtdemo - Quadtree decomposition demo.  
 roidemo - Region-of-interest processing demo.

Slide shows.

ipss001 - Region labeling of steel grains.  
 ipss002 - Feature-based logic.  
 ipss003 - Correction of nonuniform illumination.

Extended-examples.

ipexindex - Index of extended examples.  
 ipexsegmicro - Segmentation to detect microstructures.  
 ipexsegcell - Segmentation to detect cells.  
 ipexsegwatershed - Watershed segmentation.  
 ipexgranulometry - Granulometry of stars.  
 ipexdeconvwnr - Wiener deblurring.  
 ipexdeconvreg - Regularized deblurring.  
 ipexdeconvlucy - Lucy-Richardson deblurring.  
 ipexdeconvblind - Blind deblurring.  
 ipextform - Image transform gallery.  
 ipexshear - Image padding and shearing.  
 ipexmri - 3-D MRI slices.  
 ipexconformal - Conformal mapping.  
 ipexnormxcorr2 - Normalized cross-correlation.  
 ipexrotate - Rotation and scale recovery.  
 ipexregaerial - Aerial photo registration.