FUSING IMAGES FOR VISION CLARITY

By

FAZLIANA MOHD ALI

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

CERTIFICATION OF APPROVAL

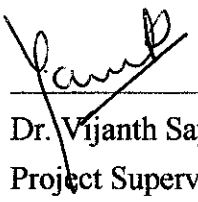
FUSING IMAGES FOR VISION CLARITY

by

Fazliana Mohd Ali

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:



Dr. Vijanth Sagayan Asirvadam
Senior Lecturer
Electrical and Electronic Engineering

Dr. Vijanth Sayagan Asirvadam
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.

Fazliana Mohd Ali

ABSTRACT

The importance of image fusion keeps increasing and its application can be found in many fields; from military and security up to medical diagnostics. This project of image fusing has been undertaken by the author to improve the clarity of night vision. The technique used is basically a pixel replacement technique between optical image and infra red (IR) image. By such technique, the fused image will have a better percept where it still have the color characteristic from optical image as well as clarity at dark portion, which is from IR image. The methodology involved in this project include identifying the problem statement, objective and scope of the project, research and review of relevant literature, familiarization with software (Matlab – Image Processing Tool and Graphical User Interface), creating the interface outline, perform fusion of still images and improvising the interface as well as the fused images. Further work for this project can be fusion for movie data, followed by real-time fusion. To work with stream of movie data, such data must be batched first into sequence of image frames, where the fusion will take part continuously on this sequence of frames. The image processing and interface used for this project will be done using MATLAB v7.1. Progress of the project, discussions and recommendations also had been detailed in this report in their respective chapters.

ACKNOWLEDGEMENTS

The author, Fazliana Mohd Ali, would like to offer her appreciation to Universiti Teknologi PETRONAS (UTP) for offering and giving her an opportunity to take the Final Year Project (FYP) course. Also, she would like to acknowledge the FYP committee of the Electrical & Electronic Engineering Department for the approval for the author to work on this project.

The author also would like to express her deepest gratitude to her Project Supervisor, Dr. Vijanth Sayagan Asirvadam, for his supervision, evaluation, guidance and support given to the author from the beginning of the project. Besides, all his motivative comments and advises were really appreciated.

Last but not least, the author is glad to acknowledge all her lecturers, fellow friends, and her family who gave her courage, strength and good feedback throughout this project, and to those who directly or indirectly involve and giving her support towards the accomplishment of this project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

EM – Electromagnetic

FYP – Final Year Project

IR – Infrared

NMF - Nonnegative Matrix Factorization

RGB – Red, Green, Blue component

CHAPTER 1
INTRODUCTION

1.1 Background of Study

Image fusion is basically a process of combining the same image from different types of input images to form a better visual percept of that image. It produces a single output image and the fused image will have more complete information which is more useful for human or machine perception. Image fusion can improve reliability by redundant information of the inputs. This technology offers a means of enhancing vision for numerous military and security applications, as well as in other fields including direct display and semi-autonomous systems:

- Intelligent robots
    - Require motion control, based on feedback from the environment from visual, tactile, force/torque, and other types of sensors
    - Stereo camera fusion
    - Intelligent viewing control
    - Automatic target recognition and tracking
- Medical image
    - Fusing X-ray computed tomography (CT) and magnetic resonance (MR) images
    - Computer assisted surgery
    - Spatial registration of 3-D surface
- Manufacturing
    - Electronic circuit and component inspection
    - Product surface measurement and inspection
    - non-destructive material inspection
- Manufacture process monitoring
    - Complex machine/device diagnostics

- Intelligent robots on assembly lines

- Military and law enforcement
  - Detection, tracking, identification of ocean (air, ground) target/event
  - Concealed weapon detection
  - Battle-field monitoring
  - Night pilot guidance
- Remote sensing
  - Using various parts of the electro-magnetic spectrum
  - Sensors: from black-and-white aerial photography to multi-spectral active microwave space-borne imaging radar
  - Fusion techniques are classified into photographic method and numerical method

By exploiting imagery from two or more spectral bands, operational effectiveness and capability can be increased significantly, leading to reduce operator workloads, as well as improving target detection and classification performance in cluttered environments. The benefits of image fusion technology are now becoming well recognized and there is a growing emphasis on incorporating such a capability within current and new systems.

Cameras can be used enhance human ability to monitor the surroundings. In many situations, it is insufficient to use just a single type of camera to provide an accurate perception of the real world. Thus data fusion from different input type has become an intense research area in recent years. The basic objective of data fusion is to derive more information through combining these inputs. The source images (or videos) will usually be collected from different types of cameras, for example, visual cameras, lowlight night vision cameras, infrared cameras (IR), millimeter wave (MMW) cameras, and X-ray imagers. The images generated from these cameras might have different characteristics, besides providing different and complementary information.

1.2 Problem Statement/Problem Identification

Vision clarity is crucial especially while driving. Most people are aware that driving at night is more demanding and stressful than driving in the daytime. Generally, at night, human eyes would loss up to 70% of their vision capability. This is one of the reasons that contribute to fact that the number of night and road accidents is much greater than those that occur during the day. Besides, a very limited vision capability during this period eases the criminals to escape after doing their job. With the help of a device that can enhance the vision capability at night, it would be troublesome to the criminals because they cannot escape easily since the police using a car equipped with the device can see their movement even in the dark and able to chase and arrest them.

1.3 Objectives and Scope of Study

The main objective of this project is to fuse two source images, which is obtained from different types of cameras – for this project, from visual and IR cameras in order to get better vision when the source of light is lacking. Intelligent pixel replacement techniques will be using for this project to get an image with better clarity. The scope would be fusing for still images using visual image from webcam/camera and infrared (IR) image from IR camera.

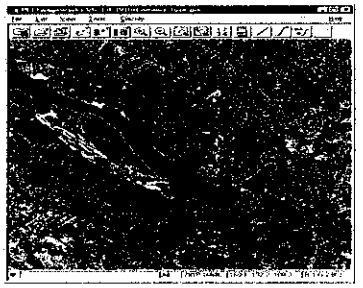
CHAPTER 2
LITERATURE REVIEW/THEORY

Research and literature review had been done with the purpose of reporting as well as for the author to get additional knowledge regarding this project. The findings obtained from the research include the importance of image fusion in today's application, as well as the techniques use in this process.

Image fusion continues to increase in importance as the complexity of data presented to human operators' increases. Fields as diverse as Military Reconnaissance, Geographical Imaging Systems, Geophysical Imaging, Medical Diagnostics, Marine Operations, Spectroscopy, and Machine Vision are all benefiting from current generation fusion systems.

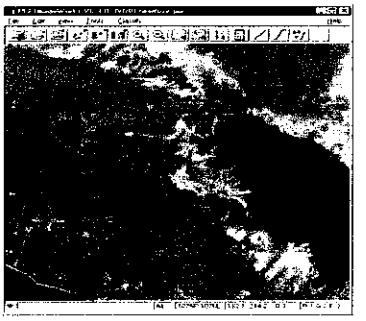
Data fusion involves the combining of one data set with a complimentary data set. This may take the form of sharpening multispectral (color) imagery with higher resolution panchromatic (black and white) imagery. This results in an image that has color information such as landuse type and detailed structural information such as drainage networks or forest cover extent. *Landsat TM 25m* is an example of multispectral imagery while *IRS-1C 5m* is an example of a higher resolution panchromatic imagery.

Another method of data fusion is fusing image data with complimentary data such as geophysical information (i.e. magnetics). This allows an effective interpretation of such things as geological structure from the imagery in comparison with magnetic anomalies from the geophysical data.



Orthorectified RADARSAT Image fused with DEM: Ottawa, Canada

Data Fusion of TM and ERSI data to a Color Image

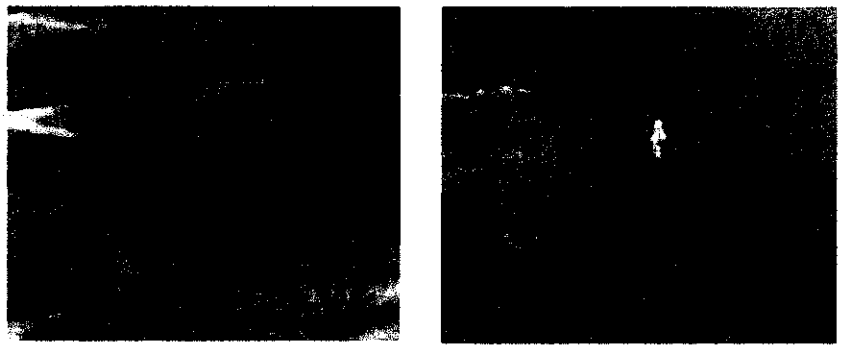


Fusion of SPOT and DEM data

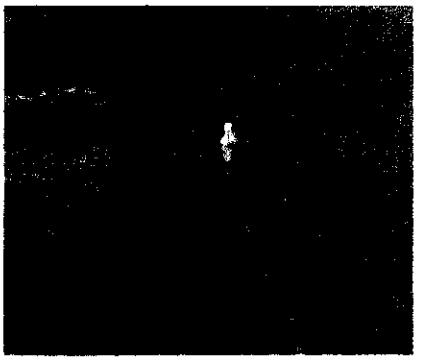
Figure 1: Fused images from different image types

There also a combined approach for fusing night-time infrared with visible imagery that had been done using several methods. In order to obtain color images resulted from visual and infrared imagery with enhanced interpretation capabilities, two-step approaches have been proposed. Fusion is carried out as a first step in order to reduce data dimensionality. One of the proposed fusion approaches is based on nonnegative matrix factorization (NMF), and provides an additive part-based representation of the source imagery. An alternative fusion approach incorporates attributes of color perception and thus provides increased discrimination capabilities. In the second step, the first order statistics of a natural color image are transferred to the fusion image in order to provide them with a natural day-time appearance.

In order to further improve the color appearance of the images and increase the visual discrimination capabilities the color transfer technique is employed. The source image for the color transfer technique is the image resulted from the fusion process and as target image the natural color image of the scene is used. In this way the color balance of the natural scene is transferred to the fused image and the salient features revealed by the fusion process are further highlighted. The color appearance of the images resulted by the NMF function process is improved significantly by the color transfer technique as can be seen in Figure 2 (b).



(a) A color image covering the visible part of the electromagnetic spectrum and with its corresponding infrared image, respectively



(b) Final image after the fusion and the use of a color transfer technique

Figure 2: (a) Inputs and (b) output of the image fusion

The final obtained color images possess a natural day-time color appearance due to the application of a color transfer technique. In this way inappropriate color mappings are avoided and the overall discrimination capabilities are enhanced.

Arithmetic operations on that are performed on images also can be used to enhance image vision. These operations involve only one spatial pixel at a time and might need either several input images to perform the operations, or using only one input image and the operations can be performed using a constant. If several inputs are used, their (height and width), however, need not be the same. When operations are performed, origin locations for each image or single image for monadic operations are aligned. Examples of arithmetic operations include addition, subtraction, multiplication and division, which different operation give different effect and used for different applications.

Addition operation helps in the development of 'Image restoration'. It causes image averaging that results in reduction of noise. With a binary operation carried pixel by pixel, this operation can adjust brightness of images and is the basis for morphological operations on images. The output pixels values can be determined by:

$$Q(i,j) = P_1(i,j) + P_2(i,j)$$

The following figure shows an 8-bit image, which has a grey level values from 0 to 255, after addition operation with a constant 30 that results in a brighter image:

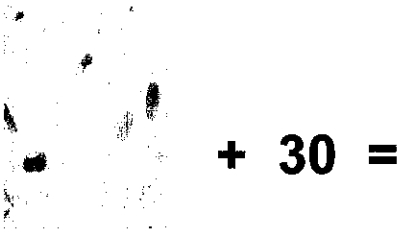


Figure3: Normal addition operation

It is important to realize that if the input images are already quite bright, then straight addition may produce a pixel value overflow, as shown in the following figure, where when a constant of 120 is added to the image, some pixel values are more than 255, which exceed the maximum grey level value.

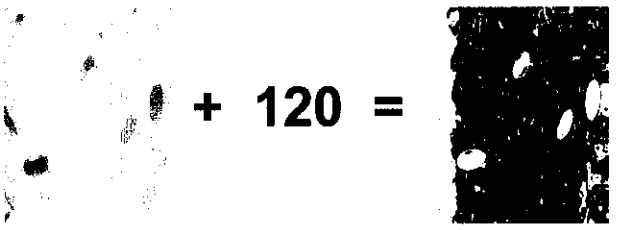


Figure4: Addition operation with overflow output

Subtraction is also can be a binary operation on two images, which is used to see relative motion effects. This operation is useful to remove or eliminate background information. It is a basic tool in medical imaging where it is used to remove noises and unnecessary details. The subtraction of two images can be performed straightforwardly in a single pass. The output pixels values are given by:

$$Q(i,j) = P_1(i,j) - P_2(i,j)$$

The example of the application of this operation can be viewed in the next figure:

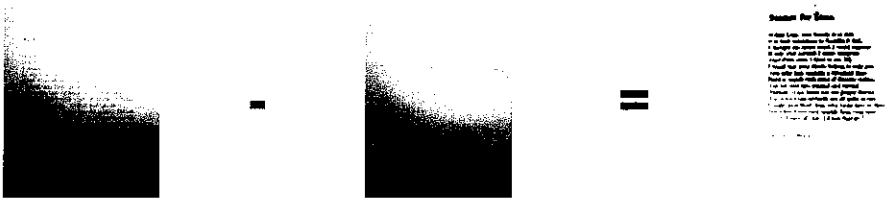


Figure 5: Background elimination using subtraction operation

There are lots of existing methods involving only one image source used to improve vision clarity in image processing field. Histogram equalization is a popular technique for improving the appearance of a poor image. Its function is similar to that of histogram stretching, which is another technique to enhance image, but histogram equalization gives visually more pleasing results across a wider range of images. It is a technique where the histogram of the resultant image is as flat as possible. The theoretical basis for histogram equalization involves probability theory, where we treat the histogram as the probability distribution of the grey levels. The next figure shows the comparison of the image as well as their corresponding histograms between the original image and after equalization:

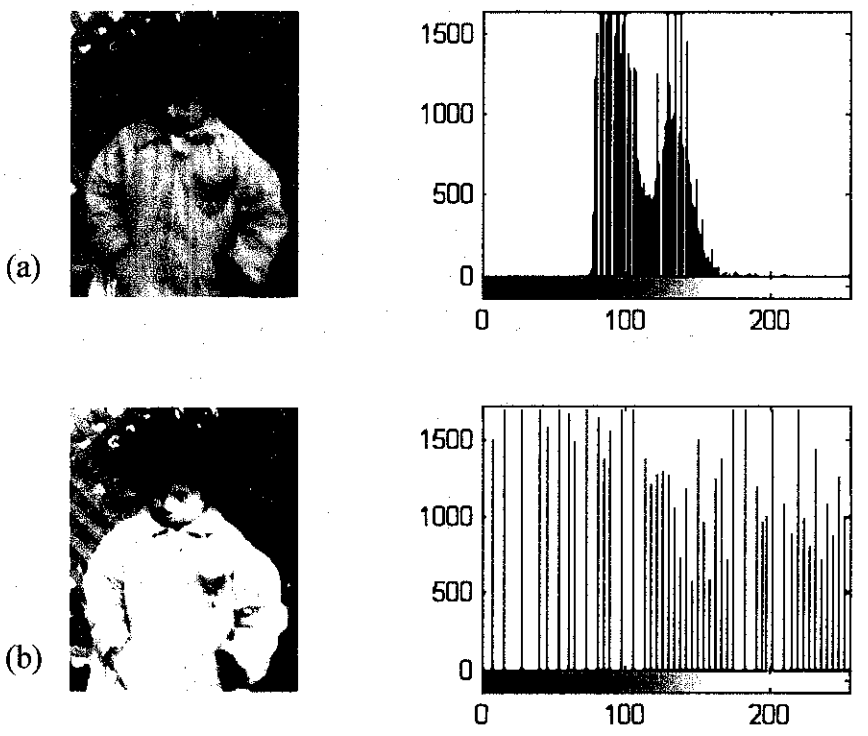
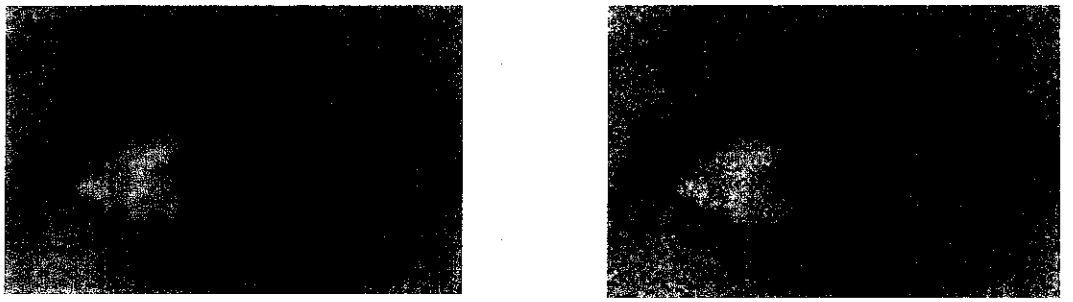


Figure 6: Images and histograms comparison of (a) original image (b) after equalization

Various types of filters also had been created for the purpose of improving the quality of images by reducing the noise, while some other filters might be used to detect edge of objects as well as to have certain effect on the image.

For example, a low pass filters pass low frequency and attenuate or eliminate high frequency information in the image. They are widely used for image compression or for hiding high frequency noises. When we remove high frequency components the image may be blurred. This filter is a linear type filter that can be used for smoothing or lowering the graininess (noise) of an image for image enhancement. It preserves the local mean of the image, but the spatial resolution of the image is decreased by the amount set forth in the low pass kernel. Figure 7 shows the difference between the original image containing high frequency noise and the resultant image after low pass filtering, which give a blur effect:



(a) (b)

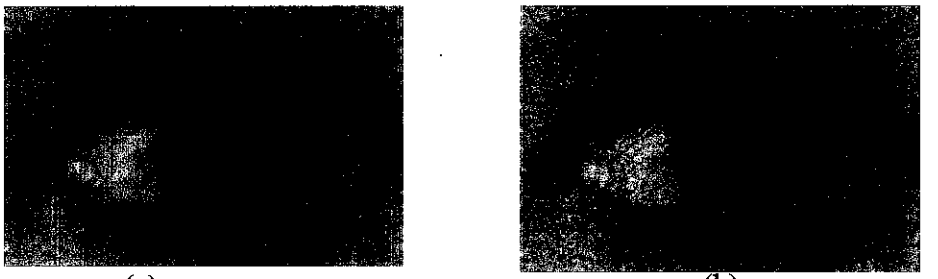
Figure 7: (a) Original image containing high frequency noise, (b) after low pass filtering

Meanwhile, a high pass filter yields edge enhancement or edge detection in the spatial domain, because edges contain many high frequencies. In high pass filtering convolution is performed and the high frequencies or edges in the filtered image are highlighted. With the low frequencies being diminished, the filtered image is getting sharpened. High pass filtering is a process used to enhance edges between different regions in an image, or well "sharpen" an image. This is achieved using a kernel with a high central value, typically surrounded by negatively weighted values, as shown in

Figure 8. The high pass filter replaces the center pixel value with a value that significantly increases its contrast from its neighbors.

-1	-1	-1		0	-1	0		1	-2	1
-1	9	-1		-1	5	-1		-2	5	-2
-1	-1	-1		0	-1	0		1	-2	1

Figure 8: Example of high pass filter kernels



(a) (b)

Figure 9: (a) Original image, (b) sharpened image after low pass filtering

Apart from the existing methods and techniques, as for this project, the reason why the author chose to use optical-IR fusion is because the application is for night vision, which IR image is more sensitive and able to see in the environment even with presence of small amount of light. Combining optical image and IR image will result in the output to have a color properties form optical image, as well as sensitivity towards dark area which is the property of IR image.

CHAPTER 3
METHODOLOGY

Project methodology had been divided into procedure identification, project flow chart and tool required. Procedures involved for the project design development included identifying the problem statement, objective and scope of the project, research and review of relevant literature, familiarization with software (Matlab – Image Processing Tool and Graphical User Interface), creating the interface outline, perform fusion of still images and identifying on how to improve the interface as well as the fused images.

3.1 Procedure identification

3.1.1 Identifying the problem statement, objective and scope of the project

The problem statement that had been identified is the need of a device that can give clarity for night vision or whenever there is lack of light. The objective is to fuse two source images and the scope of the project is fusing for visual image from webcam/camera and infrared (IR) image from IR camera.

3.1.2 Research and review of relevant literature

Research and review of relevant literature had been conducted for the purpose of having some ideas about subjects related to this project, such as the application of image fusion technology and approaches used for the fusion process.

3.1.3 Familiarization with software (Matlab – Image Processing Tool and Graphical User Interface)

The author had spend some time gone through a few tutorials in order to know better regarding the software that would involved in this project, which is Matlab. The image processing will be done using the Image Processing Tool available in the software while the interface will be done using the Graphical User Interface, or simply known as GUI, which is also available in the software.

3.1.4 Creating the interface outline

The outline of the interface was created using the GUI. This outline gives a preview on how the skeleton of the interface would look like. This outline shows how the inputs and output images would be placed, as well as the information texts and control buttons.

3.1.5 Perform fusion of still images

The coding that can perform image fusion had been created. This coding basically analyzes each pixel of the first input image, which is from the optical image, and finds the value of each pixel. If the pixel value is defined as dark, it will replace the pixel with the corresponding pixel from the second input, which is the IR image.

3.1.6 Identifying on how to improve the interface as well as the fusion process

The fusion process can be improved by enabling the users to select the maximum pixel value for the fusion to occur as well as the region of their interest. More elements can be added into the interface to improve it and make it more user-friendly.

3.1.7 Improve the interface and the fusion process

The interface had been improved with more features such as enabling the users to select inputs, enter their preferred maximum pixel value for the fusion to occur, selecting the region of their interest, error dialog box for invalid values entered and provide information about the input size.

3.2 Flow Chart

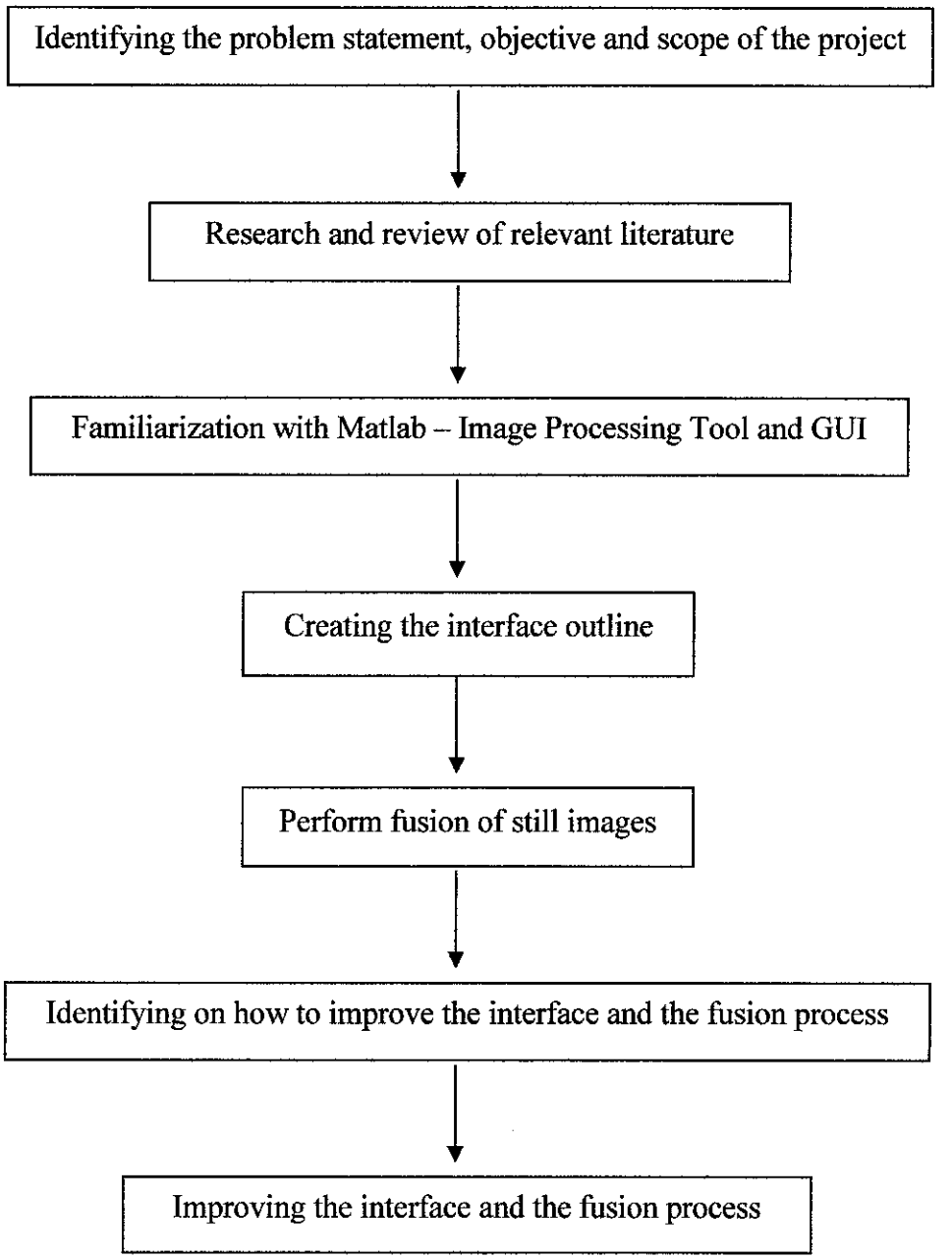


Figure 10: Flow chart of the project procedure

3.3 Tool Required

Software required in this project would be MATLAB version 6.5 or higher GUI components.

CHAPTER 4
RESULTS AND DISCUSSION

4.1 Project Progress

During previous semester, the author had worked on MATLAB v7.1 software from basic of image processing and the usage of GUI for the user interface purpose. The work included displaying and exploring images from files, differentiating image types, analyzing of images and preparing a simple layout for user interface.

As for the first four weeks of this semester, the author had started with the image processing of still images, batching of video files and improving the interface outline. The progress includes performing image fusion using two source images by pixel replacing technique, modifying the m-file code to specify the region of interest and manipulate the range of RGB component and creating outline that enable the user to open image sources from the corresponding lists provided.

For the second four weeks of this semester, the progress is more on improving the interface outline using MATLAB® software, which include adding more elements into the interface and program their functions, displaying a boundary line for the specified region of interest and adding error dialog boxes feature. Besides, enhancement of still images to make them as the second source for fusion also had been discussed.

More description of the progress is discussed further in this chapter.

4.1.1 Displaying images from files

The *imread* and *imshow* functions had been used to display an image stored in a graphics file that located in the selected directory. For example, this code reads an image into the MATLAB workspace and then displays it in a MATLAB figure window. When the code was written on the MATLAB command window (Figure 11), the figure will be displayed as shown in Figure 12:

```
>> a=imread ('Optimage01.jpg');
>> imshow (a);
```

Figure 11: Functions used for displaying images from files

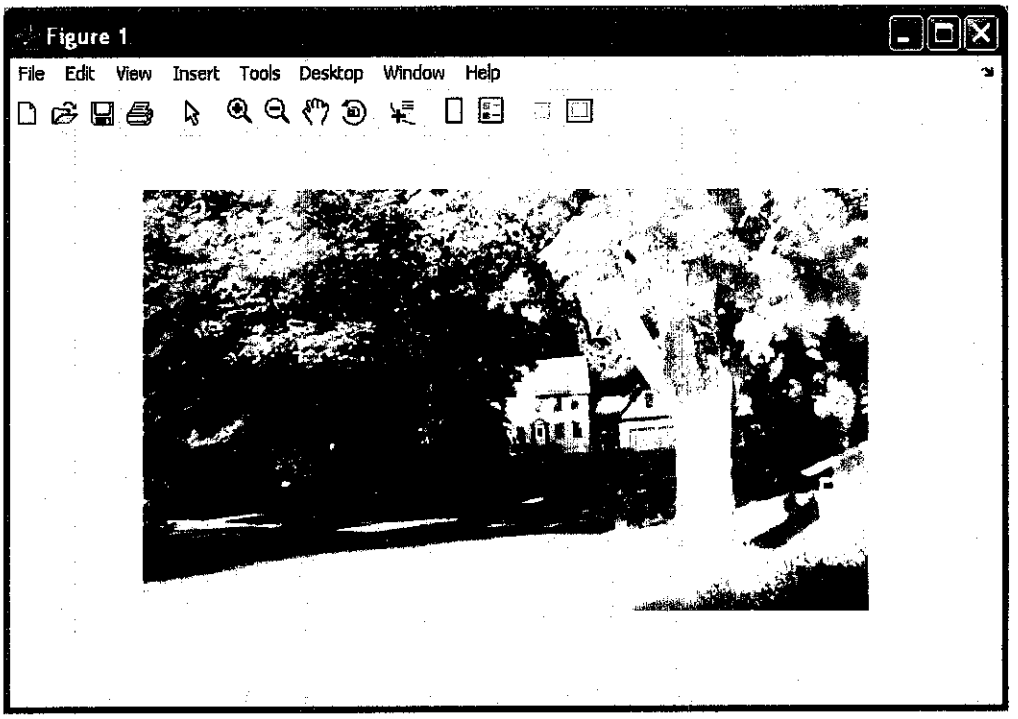


Figure 12: Displayed image from a graphic file named 'Optimage01.jpg'

4.1.2 Dialog boxes to open and save files

The *uigetfile* dialog box was used for opening and saving selected files, since the project might require input from different sources or files. The author had tried several commands for this purpose for the sake of the interface of her project as well. The form of the dialog box is shown below and Figure 13 shows an example of this code in the MATLAB command window:

[filename, pathname] = uigetfile(filter_spec,title);

Using this command, a dialog box will be displayed for the user to fill in and returns the filename and path strings and the index of the selected filter, as shown in Figure 14. A successful return occurs only if the file exists. If the user selects a file that does not exist, an error message is displayed, and control returns to the dialog box. The user may then enter another filename, or press the Cancel button.

```
>> [filename, pathname] = uigetfile( ...
       {'*.m;*.fig;*.mat;*.mdl', 'All MATLAB Files (*.m, *.fig, *.mat, *.mdl)'; ...
        '*.*',                   'All Files (*.*)'}, ...
        'Pick a file');
```

Figure 13: Using *uigetfile* function to call for the dialog box

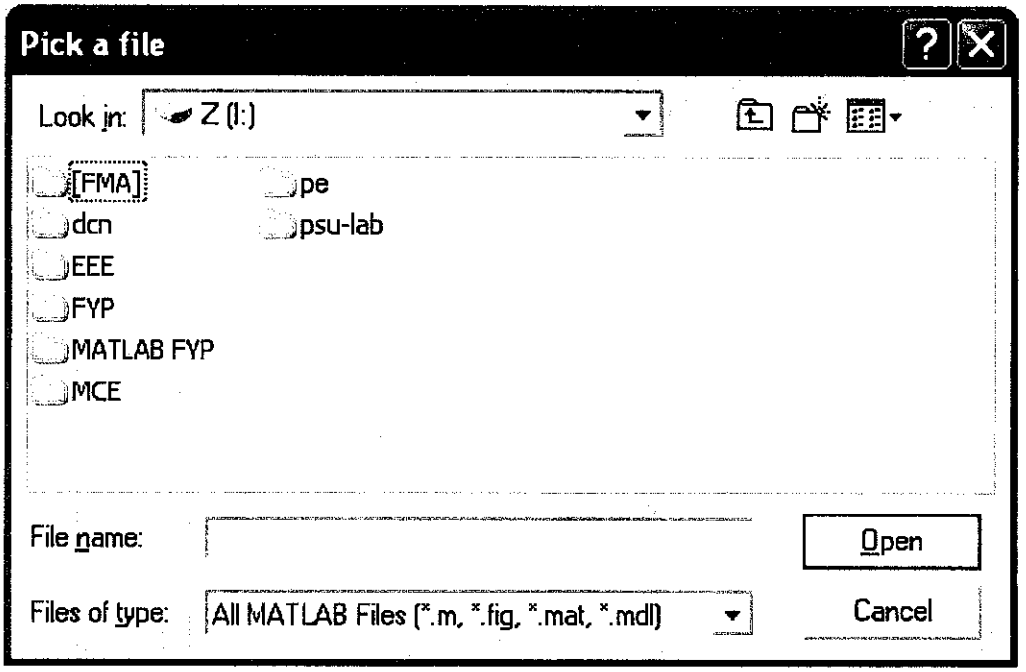


Figure 14: Dialog box used to import file to be processed

4.1.3 GUI in MATLAB v7.1

A Graphical User Interface, or GUI, provides the user with a familiar environment in which to work. This environment might contains pushbuttons, toggle buttons, lists, menus, text boxes, and so forth, all of which are already familiar to the user, so that the user can concentrate on using the application rather than on the mechanics involved in doing things. A GUI-based program must be prepared for mouse clicks (or possibly keyboard input) for any GUI element at any time. Such inputs are known as events, and a program that responds to events is said to be *event driven.* The three

principal elements required to create a MATLAB Graphical User Interface is components, figures and callbacks.

- **Components**. Each item on a MATLAB GUI is a graphical component. The types of components include graphical controls (pushbuttons, edit boxes, lists, sliders, etc.), static elements (frames and text strings), menus, and axes. Graphical controls and static elements are created by the function uicontrol, and menus are created by the functions uimenu and uicontextmenu. Axes, which are used to display graphical data, are created by the function axes.

- **Figures**. The components of a GUI must be arranged within a figure, which is a window on the computer screen. In the past, figures have been created automatically whenever data have been plotted. However, empty figures can be created with the function figure and can be used to hold any combination of components.

- **Callbacks**. Finally, there must be some way to perform an action if a user clicks a mouse on a button or types information on a keyboard. A mouse click or a key press is an event, and the MATLAB program must respond to each event if the program is to perform its function. For example, if a user clicks on a button, that event must cause the MATLAB code that implements the function of the button to be executed. The code executed in response to an event is known as a call back. There must be a callback to implement the function of each graphical component on the GUI.

The author had tried out several exercises to get familiar with the GUI feature in MATLAB and how to make use of that feature into practice. The exercises included on how to get started with the GUI, as well as some examples from the manual including how to create and display a GUI using the components mentioned above.

The author also had created an outline of the interface for her project that will be used later, as shown in the Figure 15. However, of course this GUI needs to be completed as the author go through her project until it is capable to perform the desired function.

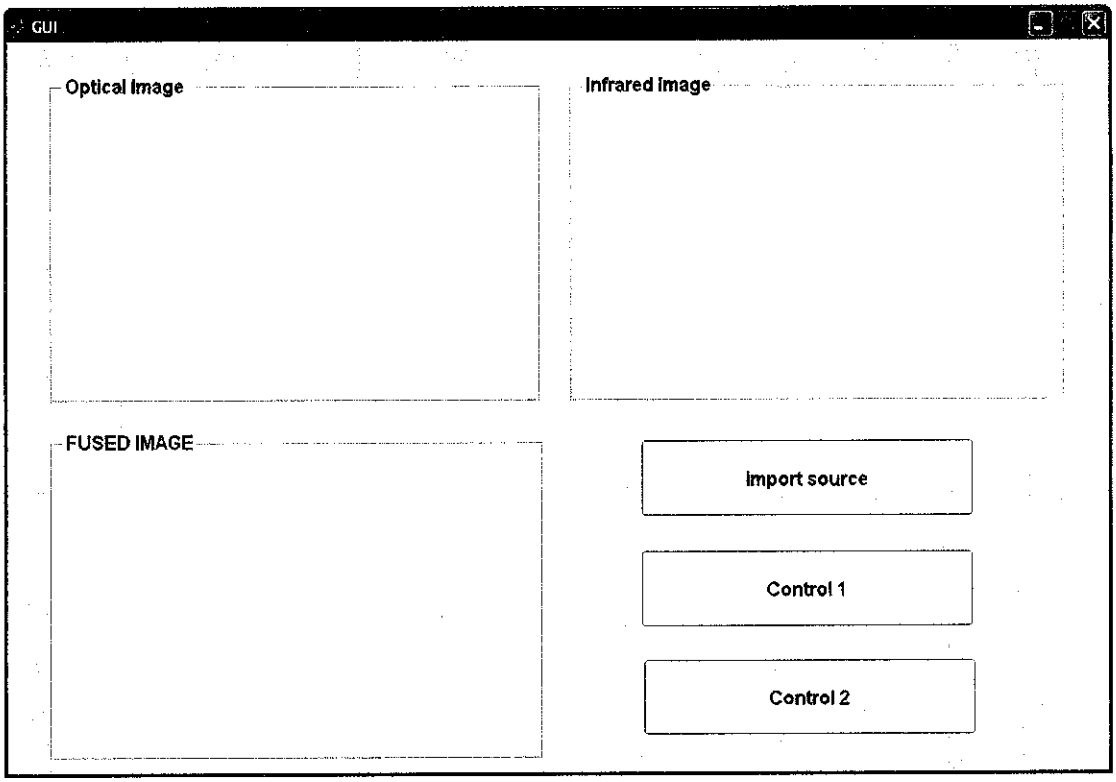


Figure 15: The interface outline that will be used for the project

From the interface outline shown in Figure 15, the user can choose any image that is to be processed by clicking on the 'Import source' button. This button might consist of a callback that will display a dialog box, as in Figure 14, so that the user can make a selection for the source of interest. This source will be displayed in the 'Optical Image' frame and its corresponding IR image might be automatically displayed in the 'Infrared Image' frame. 'Control 1' button might function to call for the image processing coding which will start the fusing process and the fused image will be displayed in the 'FUSED IMAGE' frame. 'Control 2' button might be useful for additional control feature.

4.1.4 Analyzing image pixels

The author had analyzed the pixels of different images, involving both optical images as well as IR images. The purpose of analyzing pixels was to find dark pixels from an optical image that fall within the specified range of RGB component, so that they can be replaced with their corresponding pixels from the IR image. So far, the analysis focused more on still images of both optical and IR having the same pixel coordinates. That is, a pixel of the optical image located at (i, j) has its corresponding

pixel also at (i, j) location of the IR image. An example that show a pair of optical and IR images of this type is in the following figure:

(a) Optical image

(b) IR image

Figure 16: A pair of optical and IR images with its corresponding pixel located at the same coordinates of the images.

During the analysis, the author had used *subplot* function to display the comparison between appropriate range of RGB value that makes the pixels of optical image be considered as dark pixels had been identified and selected. Only after that the selected pixels can be replaced with their corresponding pixels from the IR image. For better understanding, the author had work on a small portion of the figures that are to be analyzed. For example, when Figure 16 (a) and (b) were analyzed for a portion of pixels located at row number between 100 to 200 and column number between 100 to

175, the command used and figures displayed would be as shown in the following two figures, respectively.

```
>> a=imread ('OptimageO1.jpg');
>> b=imread ('InfraRedO1.jpg');
>> i=100:200;
>> j=100:175;
>> c=a(i,j,:);
>> d=b(i,j,:);
>> subplot (1,2,1);
>> imshow (c);
>> subplot (1,2,2);
>> imshow (d);
```

Figure 17: Command used for the analysis

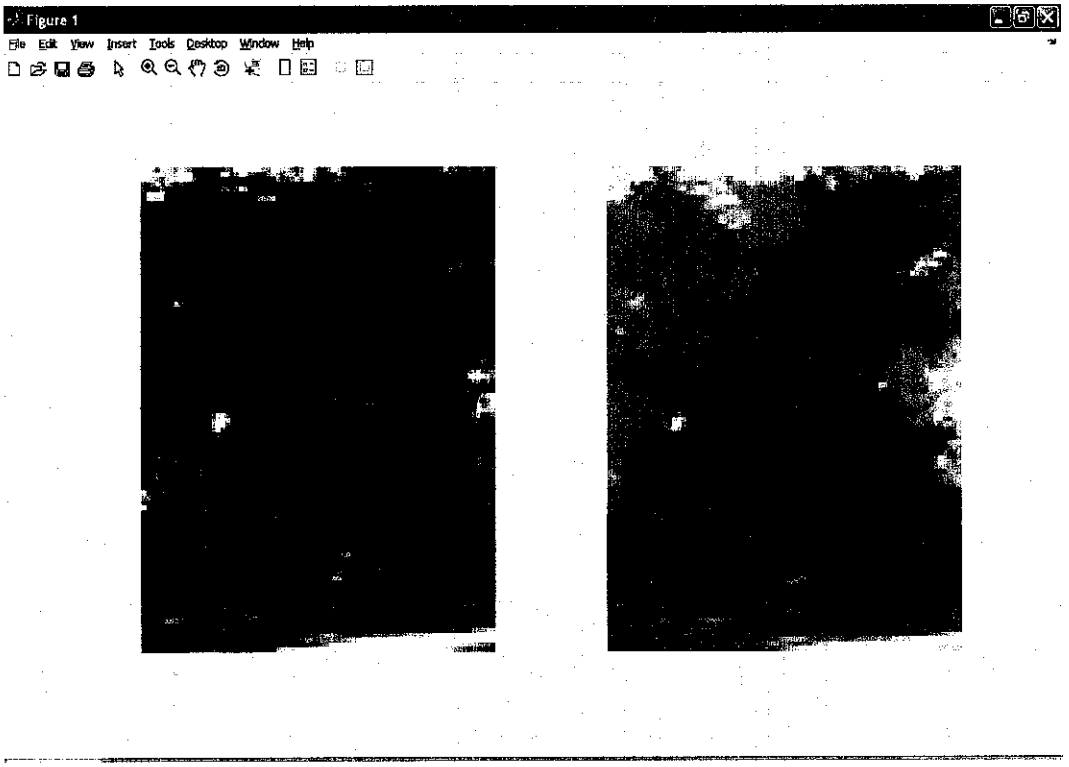


Figure 18: Figures displayed using *subplot* function

In order to analyze dark pixels, a smaller portion of the images had been used so that the analysis and comparison to be made would be easier. An example of 5 x 5 pixels from the optical image in Figure 18 had been analyzed to see which pixels have all of their RGB components value of less than 70, in order to identify black and dark pixels. The display of the 5 x 5 pixels is shown in Figure 19, while Figure 20 shows the command used as well as the results of the pixels that have RGB value of less than 70.

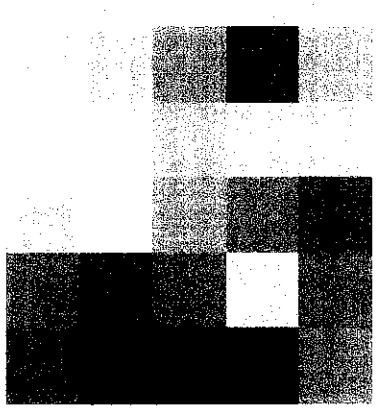


Figure 19: Display of 5 x 5 pixels from the optical image

```
>> k=100:104;
>> e=a(k,k,:);
>> e<70

ans(:,:,1) =

     0     0     0     1     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     1     1     0     0

ans(:,:,2) =

     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     1     1     0     0

ans(:,:,3) =

     0     0     1     1     0
     0     0     0     0     0
     0     0     0     1     1
     0     1     1     0     0
     1     1     1     1     0
```

Value 1 indicates the **red** component of the pixel is < 70

Value 1 indicates the **green** component of the pixel is < 70

Value 1 indicates the **blue** component of the pixel is < 70

Figure 20: Command used and the results of the pixels with RGB value less than 70

From the above figure, all answers that give the '1' values indicate that the corresponding component (the red, green or blue component) of that particular pixels

fall within the dark range, which is less than 70. For the pixels that have all of their RGB components fall within the dark range, these pixels should be replaced with their corresponding pixels from IR image.

4.1.5 Perform image fusion using two images (optical and IR)

The author had performed image fusion using two images by pixel replacing technique. Using MATLAB software, dark area of an optical image can be replaced by the corresponding pixel from IR image. Figure 21 shows an example of basic coding for the fusion process while Figure 22 shows the source images (above) and the fused image (bottom) display when *subplot* function is used:

```
 1 -  vr=imread ('vis4.jpg');
 2 -  Ir=imread('ir4.jpg');
 9 -  vis = vr;
10 -  for i = 1:250
11 -    for j = 1:445
12 -         clcd = vis(i,j,:);
13 -         if (max(clcd)<70)
14 -              vis(i,j,:) = Ir(i,j,:);
15 -         end
16 -    end
17 -  end
```

Figure 21: Example of basic coding for the fusion process

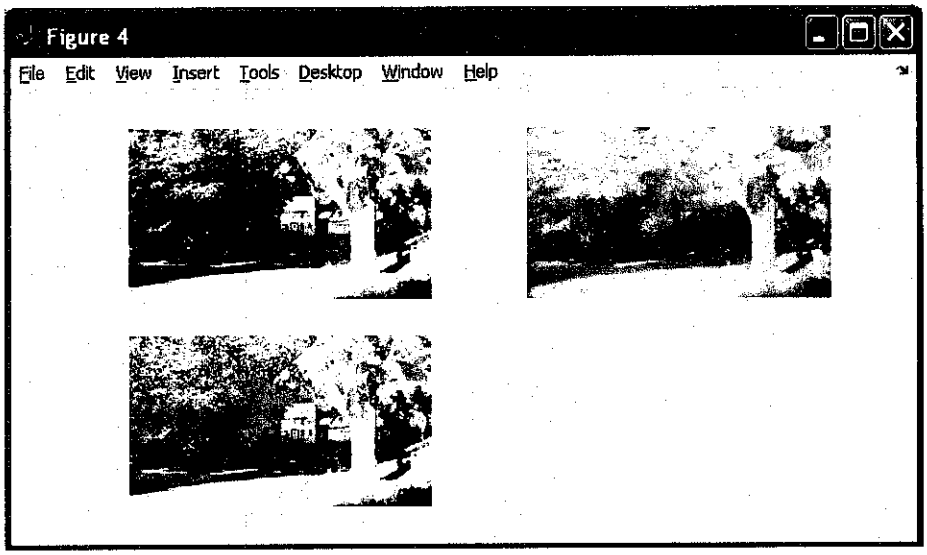


Figure 22: Source images (above) and the fused image (bottom) display

Besides that, the author can specify the region of interest that she wants the fusion to occur, as shown in Figure 23. This is done by manipulating the value of variables *i* and *j* in the coding shown in Figure 21, which will define the rows and columns that will be affected by the fusion process. She also had seen the effect to the output when the range value of RGB component was manipulated.

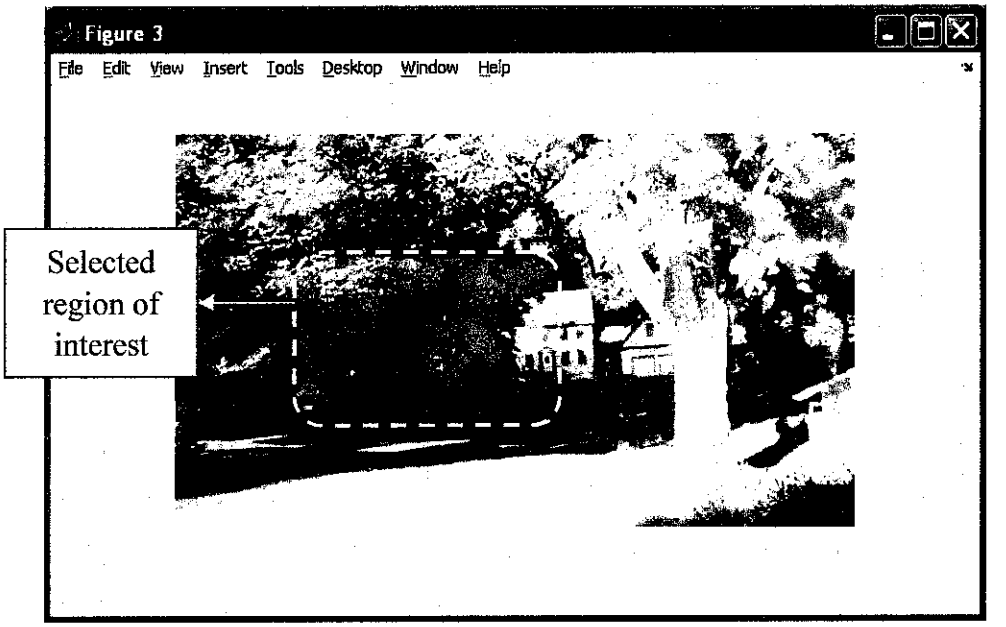


Figure 23: Fusion at selected region of interest

4.1.6 Creating an interface outline using GUI

The author had created an interface outline using the GUI component in Matlab. This outline enables the users to open image sources (either optical or IR images) from the corresponding lists provided. The next figure shows the interface outline that had been created.

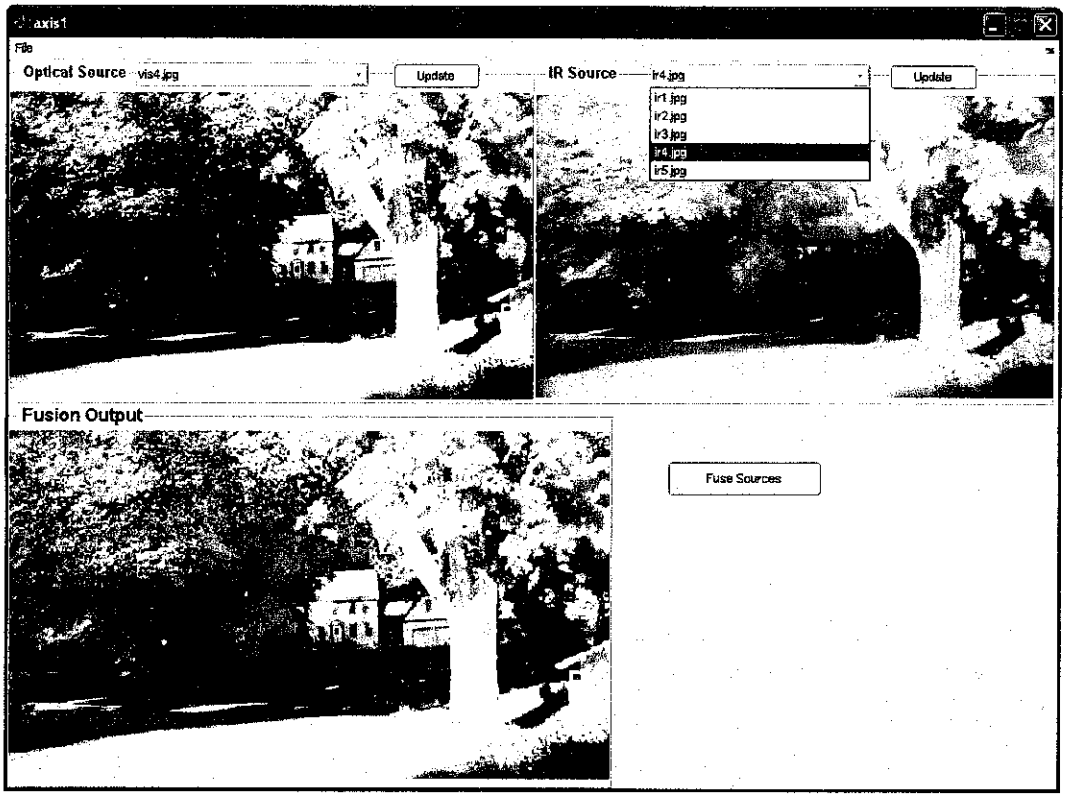


Figure 24: The interface outline

However, the image files that can be opened from this interface are limited to those shown in the popup menu only. In case the user wants to open image file that is not in the popup menu, the user has to modify the m-file code first so that the program can performed the required task. This interface can be improved so that it will be more user-friendly besides able to perform desired multitasks such as source selecting, video batching and continuous image fusion from video sources.

4.1.7 Batching of video files

A source of video files was successfully batched into a sequence of frames. This process can be performed using video decompiler software. The purpose of batching this video file is to enable the execution of continuous fusion for the video frames. With this method, a continuous fusion can be obtained.

For the time being, the video source used was a mixture of optical and IR in a single file; that is, some durations consist of optical vision and some durations

consist of IR vision. After decompiling the file, the optical and IR frames were identified and will be used for the next process, which is fusing those frames.

4.1.8 Added more elements into the interface outline for better features

The author had added more elements into the created interface outline using GUI to improve its features. The elements include radio buttons which will be used for fusion option; whether the user wants to fuse the entire image or specified region of interest.

There were also static boxes and edit boxes added for the user to enter the coordinate of the specified region as wished, in case he/she chooses to fuse a specified region of interest instead of the entire image. More of the boxes were added so that the users can specify the maximum darkness level for the fusion, which can be varied between 0 and 255. The following figure shows the added elements:

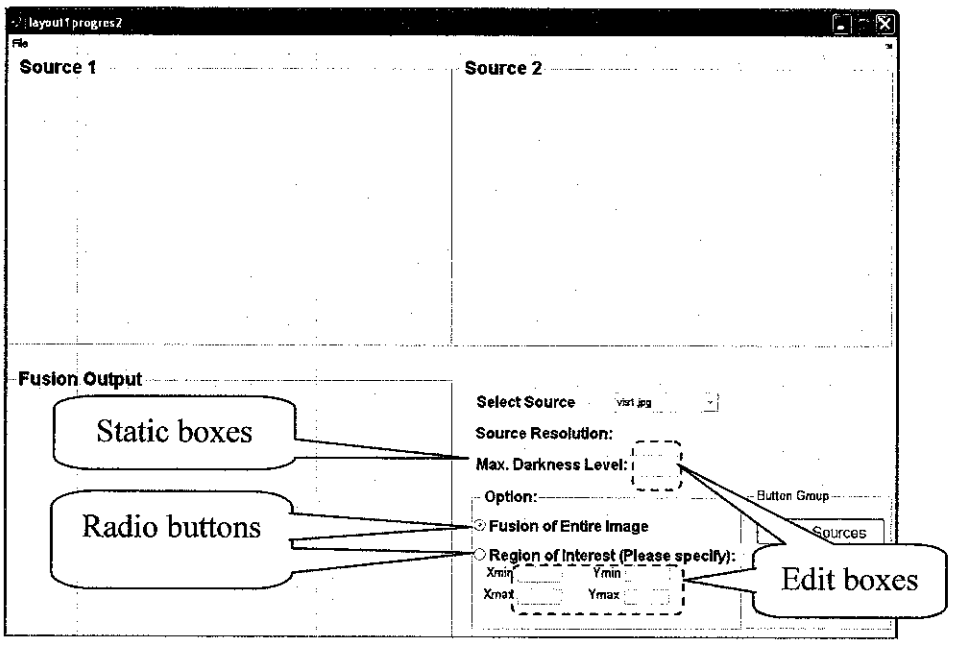


Figure 25: Added elements; radio buttons, static boxes and edit boxes

4.1.9 Programming the elements to be functioning

The functions of the added elements also had been programmed in the m-file of the interface so that it can work as desired. These include the function of fusing for pixels that fall within the specified darkness level. When a user enters the value, the fusion will take part only to the specified darkness level; the user can see the effect by changing the value anytime just from the interface, rather than the user has to modify the m-file coding every time he/she wants to change the darkness level.

The function of the radio buttons also had been programmed and thus, when the user selects "*Fusion of Entire Image*", the fusion will take part to the whole image. If "*Region of Interest*" is selected, then the user has to specify his/her region of interest, by entering valid values in the edit boxes for minimum and maximum coordinates of the region. For this purpose, `if` and `elseif` functions had been used in programming the commands that ensure the program will fuse either the whole image or at the selected region, depending on what option that have been chosen by the user. The part of the m-file coding that associates with these functions is shown in the next figure:

```
entire=get(handles.entire,'Value'); %radio button for fusing the whole image
region=get(handles.region,'Value'); %radio button for fusing selected region
        xmin=str2double(get(handles.editxmin, 'String'));
        xmax=str2double(get(handles.editxmax, 'String'));
        ymin=str2double(get(handles.editymin, 'String'));
        ymax=str2double(get(handles.editymax, 'String'));
%start process
popup_sel_index = get(handles.popupmenu1, 'Value');
fuseIR=eval(['i' num2str(popup_sel_index)]);
fuse1 = eval(['v' num2str(popup_sel_index)]);
ss1=size(fuse1);

%fuse sources
if entire==1
    for i = 1:ss1(1,1)                          %|
        for j = 1:ss1(1,2)                      %|fuse whole image
           clcd = fuse1(i,j,:);                 %|
            if (max(clcd)<darklvl)              %|
            fuse1(i,j,:) = fuseIR(i,j,:);       %|
            %fuse(i,j,:) = dispi(i,j,:);
            end
        end
    end
elseif region==1
    for i = xmin:xmax                           %|
        for j = ymin:ymax                       %|fuse region of interest
           clcd = fuse1(i,j,:);                 %|
            if (max(clcd)<darklvl)              %|
            fuse1(i,j,:) = fuseIR(i,j,:);       %|
            end
        end
    end
end
        imshow(fuse1);
```

Figure 26: `if` and `elseif` functions used regarding the value of the radio buttons

4.1.10 Boundary line for the specified region of interest

Modification had been done to the coding so that after the user defines the region of interest (the minimum and maximum coordinates of the region), a boundary line of the region will be displayed on the output image. It enables the user to see which portion of the image was selected as his/her region of interest. An example of the boundary and its coding are shown in the next two figures:

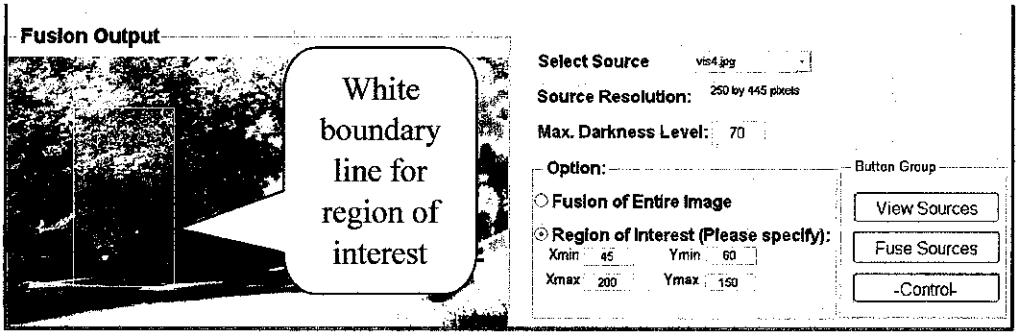


Figure 27: Boundary line for region of interest

```
if region==1
        fuse1(xmin:xmax,ymin,:) = 250;
        fuse1(xmin:xmax,ymax,:) = 250;
        fuse1(xmin,ymin:ymax,:) = 250;
        fuse1(xmax,ymin:ymax,:) = 250;
end
```

Figure 28: Coding used to make the boundary

4.1.11 Error dialog boxes features

The author had improved the created interface outline so that if a user enters invalid values of the maximum darkness level for the fusion (which should be between 0 and 255), an error dialog box will appear asking the user to enter a valid value. Similarly, the error dialog box will appear if the values of region of interest entered are invalid. This feature helps the user to determine the suitable values to be entered since it will tell the user the acceptable range values for the respective variable. The following figure is an example of an error dialog box that appears due to invalid values entered; exceed the valid value (0 to 255 for the darkness level).

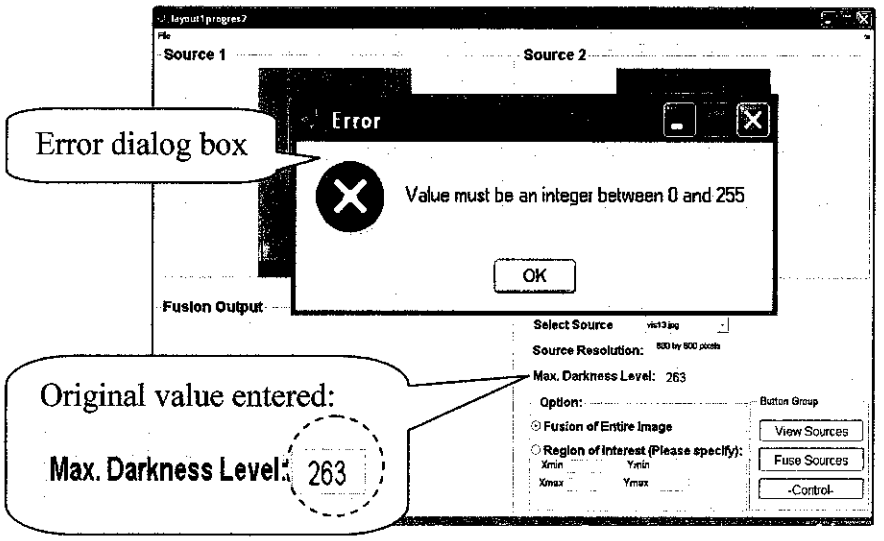


Figure 29: Error dialog box that appear when invalid values is entered

The user might have to click the 'OK' button in the error dialog box and reenter a valid value so that the fusion will be performed as desired. The `errordlg` function was used for this feature, as in the following figure, which shows the code used for the maximum darkness level values entered.

```
editdark= str2double(get(handles.editdark, 'String'));

if isnan(editdark) | editdark<=0 | editdark>=255;
   set(hObject, 'String', 0);
   errordlg('Value must be an integer between 0 and 255','Error');
end
```

Figure 30: `errordlg` function used in the coding

The same function was used for the minimum and maximum coordinate values and the dialog box that appears will tell the user the range of the valid values for the particular edit box.

4.1.12 Enhancing still images to make them as the second source for fusion

Image enhancements had been performed on several images using MATLAB® software. One of the techniques used was increasing the brightness of the image. This enables objects to be seen clearer in the dark. The following figure shows the effect of enhancing the image by increasing the brightness of the image:

(a) (b)

Figure 31: (a) Original image, and (b) enhanced image

Comparing both images, the background of the original image is totally dark but there are some chairs at the back that can be seen after the image had been enhanced. However, there are portions of the enhanced image that are too bright; the portion where the original image is already bright, for example the girls' faces. When both images are fused, the bright portion of the original image will remain and the dark portion will be replaced with the enhanced image. The brightness level for the enhancement, of course, can be determined by the user.

Another technique for image enhancement can be done by removing the red and blue color planes from the images. This technique made the images looks greenish. However, it do not help in improving the clarity of the images since the greenish images looks darker from its original image because the removed red and blue planes were set to zero (black).

The author had found that using this enhanced image as the second source would increase the execution or processing time. But still, the resultant fused image would not be as satisfactory as using IR image as the second source.

4.2 Discussion

Several difficulties that had aroused and challenged the author in this project had been identified. Amongst them is the challenge for the author to deal with a pair of optical and IR images with their corresponding pixel might be at different coordinates of the images. In other words, the image pair does not exactly the same. This type of image pair can be viewed in the next figure and the difference be compared.

(a) Optical image

(b) IR image

Figure 32: Image pair with their corresponding pixel at different coordinates of the images

From the above figure, both images look alike. However, it is obvious that the optical image was snapped at slightly higher view compared to its IR image. As the result, the corresponding pixel of both images is definitely distorted and located at different coordinate. Thus, direct pixel replacement might not be applicable in this case and additional technique needs to be performed first.

Besides that, the image pair might not be of the same size. For instant, the optical image might have a size of 450 x 253 pixels while its corresponding IR image has a size of 900 x 506 pixels. This might lead to difficulties in comparing the pixels of both images. Thus, image filter might be required to get the same size of the image pair so that the fusion using the pixel replacement technique can be performed accordingly.

This project might be involved with real time video input. Therefore, the challenge present as the author might have to be able to fuse image from this kind of input. Batching the movie data might be required for this purpose since movie is actually a sequence of images that give an illusion of continuous movement. The batching might be performed for both data from optical and IR movies. So, once these movies are batched producing sequences of images, a continuing fusing process can be performed towards these images.

In order to get the input from real time video data, a configuration of the hardware must be managed as well. In order to have both optical and IR video source, independent IR video recorder and conventional video camera, such as webcam, can be used. These two video recorders might be attached together and their view angles should be set so that the overlap view is maximized and suitable for fusing purpose. These cameras will be placed on a moving device to simulate the scenario of a moving vehicle.

CHAPTER 5
RECOMMENDATION AND CONCLUSION

5.1 Recommendations

5.1.1 Fusing video inputs

As for the future work, this project can be continued on fusing for video inputs. The interface should be able to support video type of inputs and batch them into sequences of frames, where these frames will be processed and continuous fusing can be performed. After that, the fused frames should be compiled back into video type so that the resulting output would be a video that had been fused and has a better clarity.

5.1.2 Hardware configuration

Further fusion work can be done for real time inputs. For this purpose, two cameras might be needed – webcam and IR camera – to provide real time video inputs. These two cameras can be attached on a remotely moving vehicle to analogize the real implementation of this system where it will be installed in a car. These cameras are then will be connected to the computer so that the display can be viewed from the monitor. The interface created will acquire the video inputs from both cameras and perform fusion process to these inputs.

5.1.3 Commercial purpose

This project can be further enhanced so that it would be suitable for commercial purpose. For that matter, the computer used can be replaced with a small processor unit that can communicate with the cameras and display the output on a suitable sized monitor so that it can fit in a car and consume minimum space.

5.2 Conclusion

A project of image fusing for vision clarity had been undertaken by the author by means of combining the same image from different types of input images to form a better visual percept of that image. The technique used is basically a pixel replacement technique between optical image and infra red (IR) image. By such technique, the fused image will have a better percept where it still have the color characteristic from optical image as well as clarity at dark portion, which is from IR image. The methodology involved in this project include identifying the problem statement, objective and scope of the project, research and review of relevant literature, familiarization with software (Matlab – Image Processing Tool and Graphical User Interface), creating the interface outline, perform fusion of still images and improvising the interface as well as the fused images. Further work for this project can be fusion for movie data, followed by real-time fusion. To work with stream of movie data, such data must be batched first into sequence of image frames, where the fusion will take part continuously on this sequence of frames.

REFERENCES

[1] R.C. Gonzalez and R.E. Woods, Digital Image Processing using Matlab, Latest Edition, Pearson, Prentice Hall, 2004

[2] Image Fusion 2006, < http://www.idga.org/cgi-bin/templates/document.html?document=63447&event=8626&topic=228>

[3] Ateya, Antoun I. and Stryjewski, Walter A., <http://www.sacnewsmonthly.com/invent/conveyor/toner_image_fusing.html>

[4] Image Processing, <http://www.bluechipintl.com/services_image_processing.html>

[5] V. Tsagaris, V. Anastassopoulos, 2004, *"Fusion of visible and infrared imagery for night color vision"*, Electronics and Computers Division, Physics Department, University of Patras, Greece,

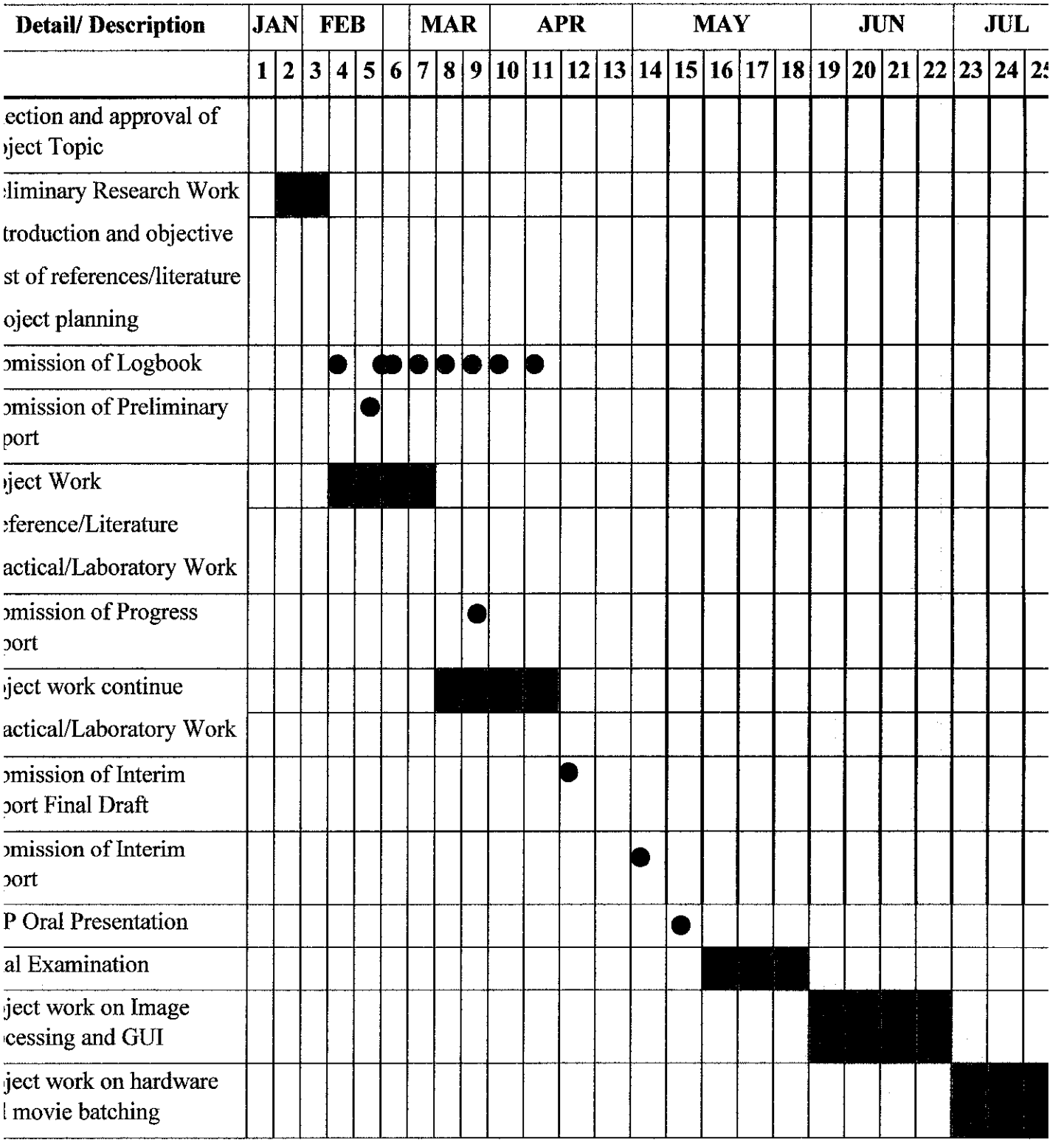
[6] Yin Chen, Rick S. Blum, "Experimental Tests of Image Fusion for Night Vision", Electrical Engineering and Computer Science Department, Lehigh University, Bethlehem, PA 18015

[7] "Investigations of Image Fusion", Electrical Engineering and Computer Science Department, Lehigh University, Bethlehem, PA 18015 <http://www.eecs.lehigh.edu/SPCRL/IF/image_fusion.htm>

[8] "Night Vision and Driving", <http://www.college-optometrists.org/index.aspx/pcms>

APPENDICES

APPENDIX 1: Gantt Chart for FYP 1



APPENDIX 2: Gantt Chart for FYP 2

No	Detail/ Week no.	AUG					SEPT													
		2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	20	21	22
1	Project Work Continue	■	■																	
	-Practical/Lab Work																			
2	Submission of Logbook		●	●	●	●	●	●	●	●										
3	Submission of Progress Report 1			●																
4	Project Work Continue			■	■	■	■													
	-Practical/Lab Work																			
5	Submission of Progress Report 2								●											
6	Project work continue							■	■	■	■									
	-Practical/Lab Work																			
7	Submission of Draft Report													●						
8	Submission of Final Report (softcover)															●				
9	Submission of Technical Report																●			
10	Oral Presentation																	■		
11	Submission of Final Report (hardcover)																			●

APPENDIX 3: Source Code of the Interface

```
function varargout = layout1d(varargin)
%LAYOUT1D M-file for layout1d.fig
%H = LAYOUT1D returns the handle to a new LAYOUT1D or the handle to
%the existing singleton*.
%LAYOUT1D('CALLBACK',hObject,eventData,handles,...) calls the local
%function named CALLBACK in LAYOUT1D.M with the given input arguments.
%LAYOUT1D('Property','Value',...) creates a new LAYOUT1D or raises the
%existing singleton*.  Starting from the left, property value pairs are
%applied to the GUI before layout1d_OpeningFunction gets called.

%Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @layout1d_OpeningFcn, ...
                   'gui_OutputFcn',  @layout1d_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

% End initialization code - DO NOT EDIT

% --- Executes just before layout1d is made visible.
function layout1d_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% varargin   command line arguments to layout1d (see VARARGIN)

% Choose default command line output for layout1d
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = layout1d_OutputFcn(hObject, eventdata, handles)

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

% --- Executes on button press in pb1.
function pb1_Callback(hObject, eventdata, handles)

%read optical sources
v1=imread ('vis1.jpg');
v2=imread ('vis2.jpg');
v3=imread ('vis3.jpg');
v4=imread ('vis4.jpg');
v5=imread ('vis5.jpg');
v6=imread ('vis6.jpg');
v7=imread ('vis7.jpg');
v8=imread ('vis8.jpg');
```

```
v9=imread ('vis9.jpg');
v10=imread ('vis10.jpg');
v11=imread ('vis11.jpg');
v12=imread ('vis12.jpg');
v13=imread ('vis13.jpg');
v14=imread ('vis Frame147.jpeg');

%read IR sources
i1=imread ('ir1.jpg');
i2=imread ('ir2.jpg');
i3=imread ('ir3.jpg');
i4=imread ('ir4.jpg');
i5=imread ('ir5.jpg');
i6=imread ('ir6.jpg');
i7=imread ('ir7.jpg');
i8=imread ('ir8.jpg');
i9=imread ('ir9.jpg');
i10=imread ('ir10.jpg');
i11=imread ('ir11.jpg');
i12=imread ('ir12.jpg');
i13=imread ('ir13.jpg');
i14=imread ('ir Frame146.jpeg');

%update display for optical
axes(handles.axes1);
cla;
popup_sel_index = get(handles.popupmenu1, 'Value');
for a=1:14;
    dispv=eval(['v' num2str(a)]);
 switch popup_sel_index
     case (a)
        imshow (dispv);
 end
end

%update display for IR
axes(handles.axes2);
cla;
popup_sel_index_IR = get(handles.popupmenu2, 'Value');
for a=1:14;
    dispi=eval(['i' num2str(popup_sel_index)]);
 switch popup_sel_index_IR
    case (a)
         imshow (dispi);
 end
end

%to find the value typed into the editbox
fuse = eval(['v' num2str(popup_sel_index)]);
ss=size(fuse);
resoln = sprintf('%d by %d pixels',ss(1,1) ,ss(1,2));

%place the value into the text field
set (handles.text2,'string', resoln);

% --------------------------------------------------------------------
function FileMenu_Callback(hObject, eventdata, handles)
% --------------------------------------------------------------------
function OpenMenuItem_Callback(hObject, eventdata, handles)

file = uigetfile('*.fig');
if ~isequal(file, 0)
```

```
    open(file);
end
% --------------------------------------------------------------------
function PrintMenuItem_Callback(hObject, eventdata, handles)

printdlg(handles.figure1)
% --------------------------------------------------------------------
function CloseMenuItem_Callback(hObject, eventdata, handles)

selection = questdlg(['Close ' get(handles.figure1,'Name') '?'],...
                     ['Close ' get(handles.figure1,'Name') '...'],...
                     'Yes','No','Yes');
if strcmp(selection,'No')
    return;
end

delete(handles.figure1)

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)

if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

set(hObject, 'String', {'vis1.jpg', 'vis2.jpg', 'vis3.jpg', 'vis4.jpg',
'vis5.jpg','vis6.jpg','vis7.jpg','vis8.jpg','vis9.jpg','vis10.jpg','vis11.jpg'
,'vis12.jpg','vis13.jpg','vis14.jpg'});

% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)

set(hObject, 'String', {'ir1.jpg', 'ir2.jpg', 'ir3.jpg', 'ir4.jpg',
'ir5.jpg','ir6.jpg','ir7.jpg','ir8.jpg','ir9.jpg','ir10.jpg','ir11.jpg','ir12.
jpg','ir13.jpg','ir14.jpg'});

% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

set(hObject, 'String', {'ir1.jpg', 'ir2.jpg', 'ir3.jpg', 'ir4.jpg',
'ir5.jpg','ir6.jpg','ir7.jpg','ir8.jpg','ir9.jpg','ir10.jpg','ir11.jpg','ir12.
jpg','ir13.jpg','ir14.jpg'});


% --- Executes on button press in pb2.
function pb2_Callback(hObject, eventdata, handles)

axes(handles.axes3);
cla;

darklvl=str2double(get(handles.editdark, 'String'));
entire=get(handles.entire,'Value');
```

```
region=get(handles.region,'Value');
compliment=get(handles.compliment,'Value');
        xmin=str2double(get(handles.editxmin, 'String'));
        xmax=str2double(get(handles.editxmax, 'String'));
        ymin=str2double(get(handles.editymin, 'String'));
        ymax=str2double(get(handles.editymax, 'String'));
%start process
popup_sel_index = get(handles.popupmenu1, 'Value');
fuseIR=eval(['i' num2str(popup_sel_index)]);
fuse1 = eval(['v' num2str(popup_sel_index)]);
ss1=size(fuse1);

        xminval=get(handles.editxmin, 'Value');
        xmaxval=get(handles.editxmax, 'Value');
        yminval=get(handles.editymin, 'Value');
        ymaxval=get(handles.editymax, 'Value');

 if region==1|compliment==1
       if xminval==0 | xmaxval==0 | yminval==0 |ymaxval==0
            errordlg('Please specify Region of Interest first','Error');
       else
        fuse1(xmin:xmax,ymin,:) = 250;
        fuse1(xmin:xmax,ymax,:) = 250;
        fuse1(xmin,ymin:ymax,:) = 250;
        fuse1(xmax,ymin:ymax,:) = 250;
       end
 end

%fuse sources
if entire==1
    for i = 1:ss1(1,1)                          %|
        for j = 1:ss1(1,2)                      %|fuse whole image
           clcd = fuse1(i,j,:);                 %|
            if (max(clcd)<darklvl)              %|
            fuse1(i,j,:) = fuseIR(i,j,:);       %|
            %fuse(i,j,:) = dispi(i,j,:);
            end
        end
    end
elseif region==1
    for i = xmin:xmax                           %|
        for j = ymin:ymax                       %|fuse region of interest
           clcd = fuse1(i,j,:);                 %|
            if (max(clcd)<darklvl)              %|
            fuse1(i,j,:) = fuseIR(i,j,:);       %|
            end
        end
    end
end
        imshow(fuse1);

function edit1_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in text1.
```

```
function text1_Callback(hObject, eventdata, handles)

% --- Executes on button press in pb3.
function pb3_Callback(hObject, eventdata, handles)

% --- Executes on button press in entire.
function entire_Callback(hObject, eventdata, handles)

% --- Executes on button press in region.
function region_Callback(hObject, eventdata, handles)

function editxmin_Callback(hObject, eventdata, handles)
% Hints: get(hObject,'String') returns contents of editxmin as text
% while str2double(get(hObject,'String')) returns as a double

popup_sel_index = get(handles.popupmenu1, 'Value');
fuse = eval(['v' num2str(popup_sel_index)]);
ss=size(fuse);
rownum = ss(1,1);
colnum = ss(1,2);
editxmin = str2double(get(hObject, 'String'));
errx1=sprintf('Xmin must be an integer between 1 and %d',(rownum-1));

if isnan(editxmin)|editxmin<=0|editxmin>=rownum;
    set(hObject, 'String', 0);
    errordlg(errx1,'Error');
end

% --- Executes during object creation, after setting all properties.
function editxmin_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editxmax_Callback(hObject, eventdata, handles)

popup_sel_index = get(handles.popupmenu1, 'Value');
fuse = eval(['v' num2str(popup_sel_index)]);
ss=size(fuse);
rownum = ss(1,1);
colnum = ss(1,2);
editxmax = str2double(get(hObject, 'String'));
xmin= str2double(get(handles.editxmin, 'String'));
errx2=sprintf('Xmax must be an integer between %d and %d ', (xmin+1),rownum);

if isnan(editxmax) | editxmax<=xmin | editxmax>=(rownum+1);
    set(hObject, 'String', 0);
    errordlg(errx2,'Error');
end


% --- Executes during object creation, after setting all properties.
function editxmax_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editymin_Callback(hObject, eventdata, handles)
```

```
popup_sel_index = get(handles.popupmenu1, 'Value');
fuse = eval(['v' num2str(popup_sel_index)]);
ss=size(fuse);
rownum = ss(1,1);
colnum = ss(1,2);
editymin = str2double(get(hObject, 'String'));
errx1=sprintf('Ymin must be an integer between 1 and %d',(colnum-1));

if isnan(editymin)|editymin<=0|editymin>=colnum;
    set(hObject, 'String', 0);
    errordlg(errx1,'Error');
end

% --- Executes during object creation, after setting all properties.
function editymin_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editymax_Callback(hObject, eventdata, handles)
popup_sel_index = get(handles.popupmenu1, 'Value');
fuse = eval(['v' num2str(popup_sel_index)]);
ss=size(fuse);
rownum = ss(1,1);
colnum = ss(1,2);
editymax = str2double(get(hObject, 'String'));
ymin= str2double(get(handles.editymin, 'String'));
erry2=sprintf('Ymax must be an integer between %d and %d ', (ymin+1),colnum);

if isnan(editymax) | editymax<=ymin | editymax>=(colnum+1);
    set(hObject, 'String', 0);
    errordlg(erry2,'Error');
end

% --- Executes during object creation, after setting all properties.
function editymax_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editdark_Callback(hObject, eventdata, handles)

editdark= str2double(get(handles.editdark, 'String'));

if isnan(editdark) | editdark<=0 | editdark>=255;
    set(hObject, 'String', 0);
    errordlg('Value must be an integer between 0 and 255','Error');
end

% --- Executes during object creation, after setting all properties.
function editdark_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```