Creating Video Processing Suite Using Java Media Framework

By

Logenthiran A/L Manogaran

Dissertation submitted in partial fulfilment of
the requirements for the
Bachelor of Engineering (Hons)
(Electrical and Electronics Engineering)

DEC 2009

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Creating Video Processing Suite Using Java Media Framework

by

Logenthiran A/L Manogaran

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

Approved by,

____
(Dr. Vijanth Sagayan Asirvadam)
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

DEC 2009

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.

____

LOGENTHIRAN A/L MANOGARAN

ABSTRACT

This project is to develop a video processing suite using Java Media Framework. Previously an advanced video processing suite has been successfully been developed in MATLAB. However, due to the expensive licensing fees of MATLAB, not all computers are able to be equipped with it and this has limited a wider usage of the suite. By developing a simple video processing suite in Java, it is believed that it can be implemented in any computer because most of the computer is being installed with the Java Runtime Environment. The suite will be able to detect video from the webcam, able to use the input from webcam as the input for the image and video processing, able to manipulate the image and video input and to detect the presence of foreign or moving object in the video. This suite will acts as a base for future development of an advanced video processing suite matching or exceeding the one developed previously in MATLAB. The tools needed for the completion of this project are a laptop, webcam, Java Development Kit, Java Runtime Environment, Java Media Framework and Java Integrated Development Editor.

ACKNOWLEDGEMENT

I would like to thank God for the opportunity granted to me to be able to work on this project with the support of many great people, family and friends. There are so many people that I would like thanks for aiding and supporting me throughout the time period that I am doing this project. Firstly, I would like thank my supervisor, Dr. Vijanth Sagayan Asirvadam for his support, assistance, knowledge and solutions in completing this project. I could not have a much better and supporting supervisor than him and because of that, I would like say a lot of 'Thank You' to him. Besides that, I would like to thank my family, especially my parents, Mr. Manogaran and Mrs. Teng Siew Hong, for their support and advice in managing the time and stress due to the project. Another important person that I would like to thank is Mrs. Siti Hawa for her willingness to go through my report to ensure proper formatting. I would also like to thank all my course mates for their input and thought on this project. Finally, I would like to thank all the individuals that I have came to know through this project; you have made the journey to complete this project, a wonderful and unique journey. To all the individuals that have been involved directly and indirectly in my project, I would like to say: THANK YOU!

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1. Background Of Study

The definition of a software suite is a collection of software application of related tasks, which shares the same user interface.

For this project, the video processing suite is an application, which main function is to processing input signal in the video format. The functionality that can be incorporated into the suite can range from video playback and save, video editing, manipulation of video and other functionality. The video processing suite can incorporate a large range of function in it. So the required functionality of the suite in this project will be specified in the Scope of the Project section.

The development of the video processing suite will be done on Java Programming Language. This programming language is being chosen because of its large libraries support for the language, it is free to be used without any fees and the developed software can be implemented on any operating system, as long Java Runtime Environment is installed in the computer.

Another important part is the Java Media Framework, which is the extension of Java libraries that add support for audio and video processing. Besides that, there are a lot of communities and forums, which discussed regarding Java Programming Language. With this support, it will contribute to a more rapid development of this suite.

1.2. Problem Statement

Previously, a video processing suite was successfully built through the use of MATLAB. Even though the suite was successful in performing its tasks, the suite was not able to be used in any computer. This is due to the construction of the suite through the use of MATLAB. Since the license fees of MATLAB cost thousands, not all the computer are able to be equip with it and this reduce the implementation of the suite in any computer.

Due to that reason, this project is being initiated to build a video processing suite through the usage of open-source programming language, which is free for anyone or any computer to use it, which in this project will be the Java Programming Language.

1.3. Objective

To built a simple video processing suite through the use of Java Programming Language.

1.4. Scope Of Study

The scope of this project is to build a video processing suite, which function as a surveillance system. The suite will be able to:

1. Detect and receive input from the camera or webcam
2. Apply certain effects on the image input from the camera or webcam
3. Apply certain effects on the video input from the camera or webcam
4. Detect and track the objects that have entered to the environment that is being captured by the camera

CHAPTER 2
LITERATURE REVIEW

2.1 Theory

Initially, the objective of this project is to develop a video processing suite that have the ability to manipulate the video from an input that is a camera or webcam and the ability to detect any motion or foreign object in the input. The jump start the project, this project will be initialized through the development of an image processing suite that have the same function as the video processing suite. The only difference will be the type of input that the suite can operate.

2.2 Application Programming Interface (API)

This section will cover the components of Java Programming Language that will be used for this project to achieve the development of a complete image and video processing suite. The components that will be discussed are Java 2D API, Java Advanced Imaging and Java Media Framework. All these three components will work together to achieve the objective of this project that is the development of an image and video processing suite.

The three components are API that have been designed to achieve certain or specific functionality. API or Application Programming Interface is a set of routines, data structures, object classes and/or protocols provided by libraries and/or operating system services in order to support the building of applications. [1]

2.2.1 Java 2D API

The first required component is the Java 2D API, which has the primary functionality of drawing two dimensional graphics. [2] The main use is to provide the user with the tools to develop software, with similar features as paint program. How ever, by using the java.awt.image.BufferedImage package, images from the computer can be buffered into the memory of the computer and the buffered image can be manipulated to produce certain effects, be displayed to a window and the manipulated can be saved.

2.2.2 Java Advanced Imaging

The second component is Java Advanced Imaging or JAI. JAI is an API that functions to provide developer the ability to do high performance image processing. [3] This API has a lot of image processing functionality related to image processing built into it. The buffered image that is being implemented through the Java 2D API can be send to the JAI to achieve the required processing and can be saved to the hard disk or to be displayed to a window. A lot more research have to be done to know the full list of functions that this API can contributed towards the image and video processing suite.

2.2.3 Java Media Framework

The last API is the Java Media Framework, which enabled the developer to incorporate the ability to stream audio and video from the computer or audio and video input devices such as microphone, webcam and etc. [4] This API will enable the receiving of audio and video input and this input theoretically can be send to the buffered image to enable it to be processed through the use of Java 2D API and JAI.

Even though the title of this project "Video Processing Using Java Media Framework", it does not mean that this project will only focus on the usage of that API. Instead Java Media Framework only enables the reception of audio and video input and other API's are require to do some level of processing on to the input that are able to be receive through the Java Media Framework.

CHAPTER 3
METHODOLOGY

3.1. Procedure Identification

The objective of this project is to develop a simple video processing suite, which can be the bases of future upgrade or improvement in term of functionality and complexity. However, the initial step in developing this video suite to build a simple image suite, which will incorporate the effects or functionality of the video suite. The reason behind this approach is video is a series of images being displayed in a sequences at a certain speed. So by developing an image processing suite, it will provide a fundamental and basic knowledge regarding the programming processes and language.

The project flow is being divided to four main sections, which are: the video upload and image capture from the webcam, the image manipulation effects on images, the video manipulation effects on video inputs from the webcam and the graphic user interface of the suite. The first three section need to be completed or the general idea of the first three section need to be obtained so that a more accurate GUI can be designed for the suite and this will eliminate any major modification in the future.

For the video upload and image capture part, the input from the webcam are required to be displayed into a window on the computer and has the ability to capture the input from the webcam. For the image manipulation effects, this will consist of all the required effects that will be applied to the input image from the webcam.

For video manipulation effects, this will consist of all the required effects that will be applied to the input video from the webcam. For the motion detection function, the suite will be able to detect the entrance of new object or person into a set background. For the GUI, it will be the user interface that the user will be using when the user is operating the suite.

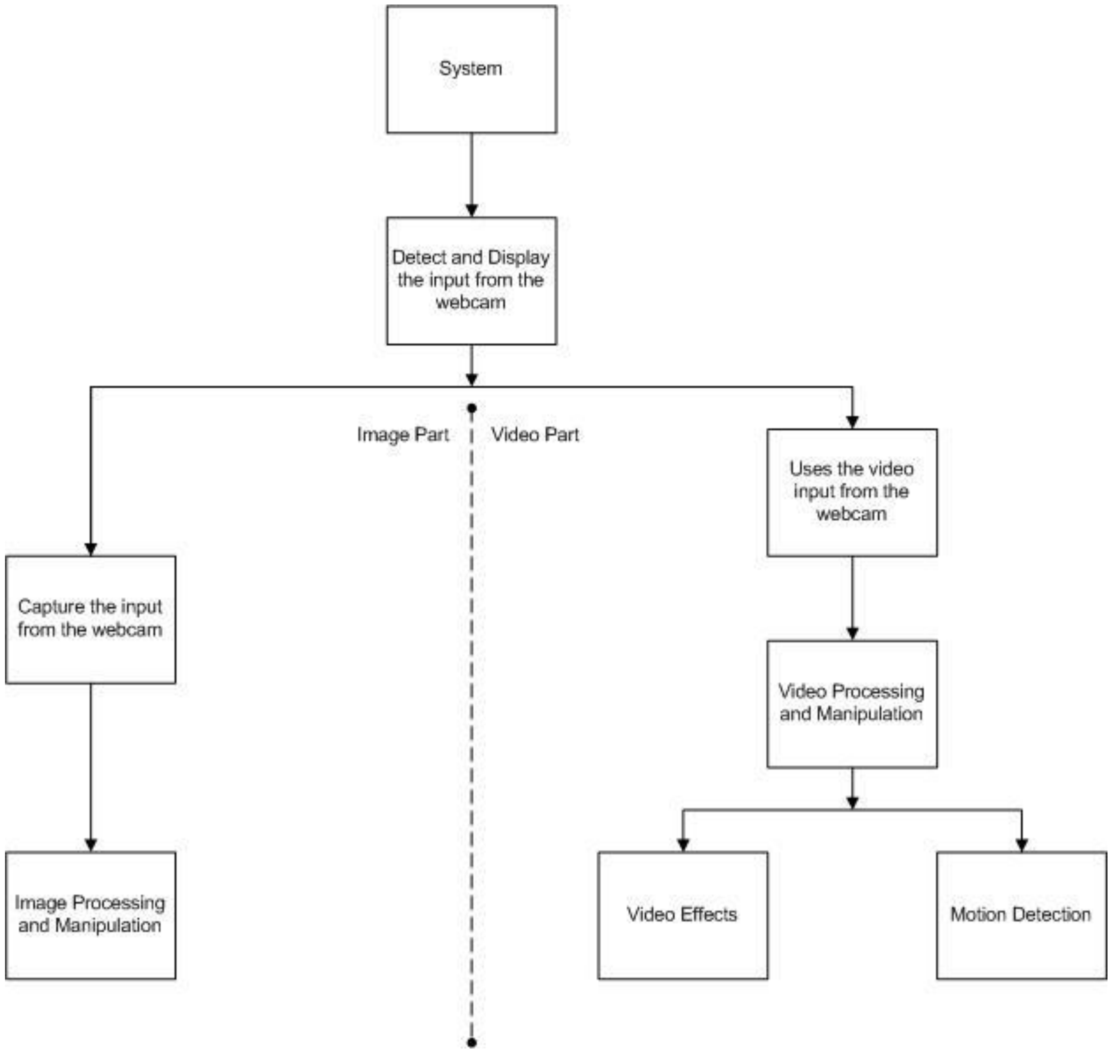

System
Detect and Display the input from the webcam
Image Part
Video Part
Uses the video input from the webcam
Capture the input from the webcam
Video Processing and Manipulation
Image Processing and Manipulation
Video Effects
Motion Detection

Figure 1: The system diagram of the suite

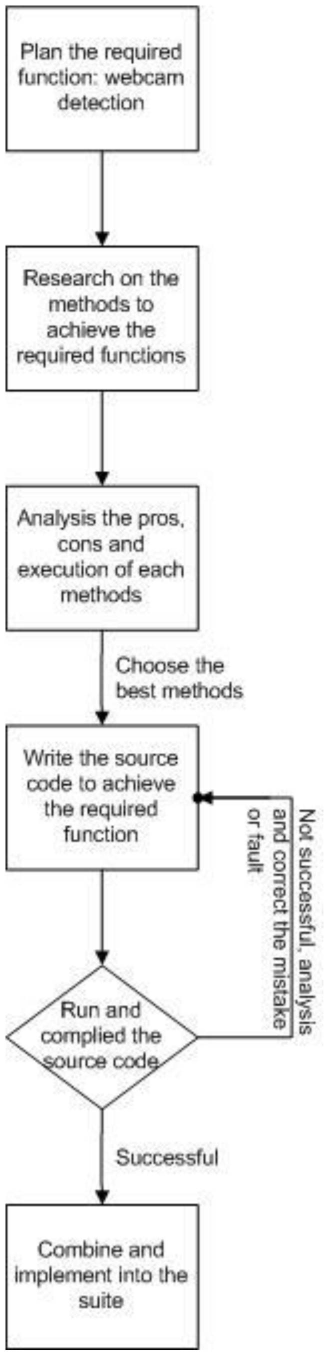
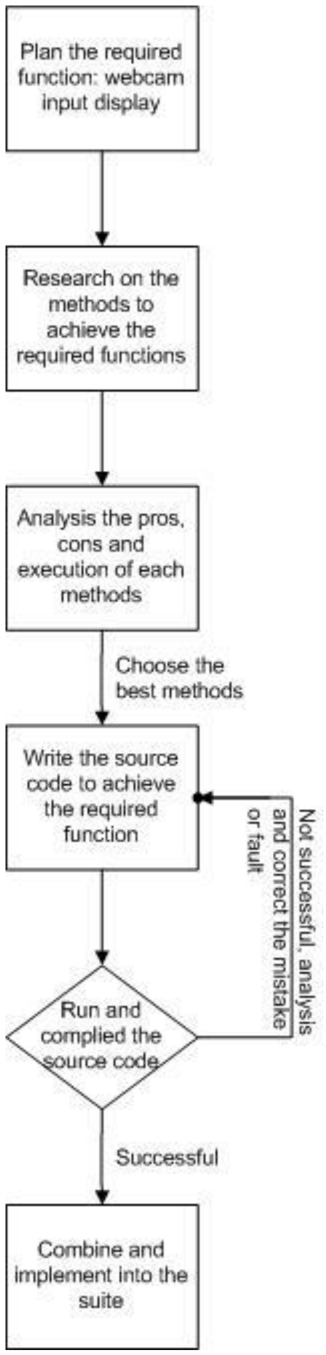

Plan the required function: webcam detection
Research on the methods to achieve the required functions
Analysis the pros, cons and execution of each methods
Choose the best methods
Write the source code to achieve the required function
Run and complied the source code
Not successful, analysis and correct the mistake or fault
Successful
Combine and implement into the suite

Figure 2: The sequence of tasks related to the development of webcam detection



Plan the required function: webcam input display
Research on the methods to achieve the required functions
Analysis the pros, cons and execution of each methods
Choose the best methods
Write the source code to achieve the required function
Not successful, analysis and correct the mistake or fault
Run and complied the source code
Successful
Combine and implement into the suite

Figure 3: The sequence of tasks related to the development of the webcam input display

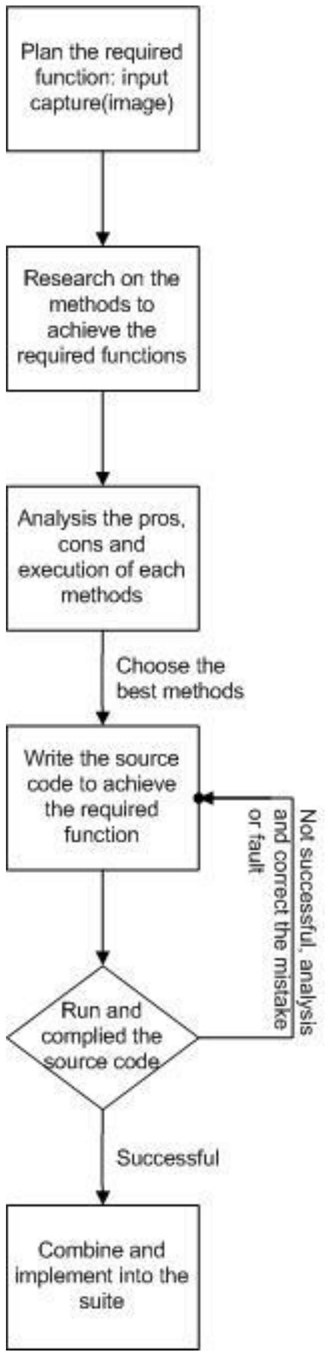

Plan the required function: input capture(image)
Research on the methods to achieve the required functions
Analysis the pros, cons and execution of each methods
Choose the best methods
Write the source code to achieve the required function
Not successful, analysis and correct the mistake or fault
Run and complied the source code
Successful
Combine and implement into the suite

Figure 4: The sequence of tasks related to the development of the webcam input capture

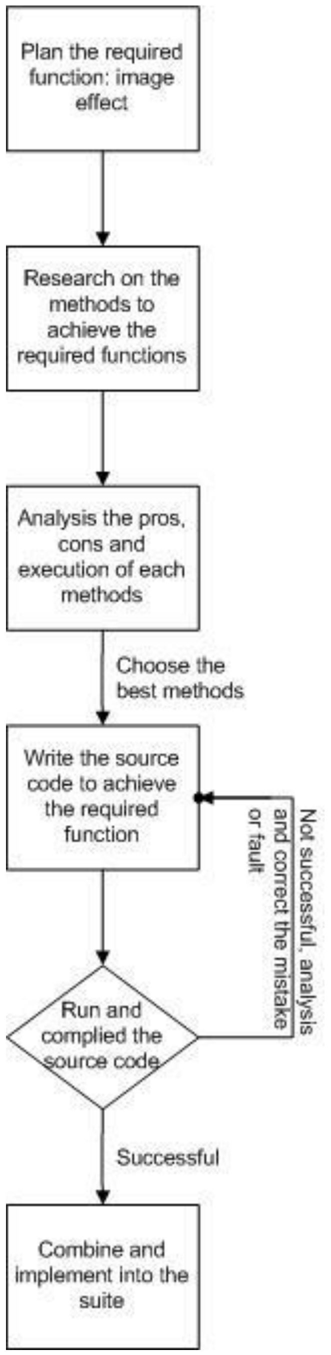

Plan the required function: image effect
Research on the methods to achieve the required functions
Analysis the pros, cons and execution of each methods
Choose the best methods
Write the source code to achieve the required function
Run and complied the source code
Not successful, analysis and correct the mistake or fault
Successful
Combine and implement into the suite

Figure 5: The sequence of tasks related to the development of each image effects

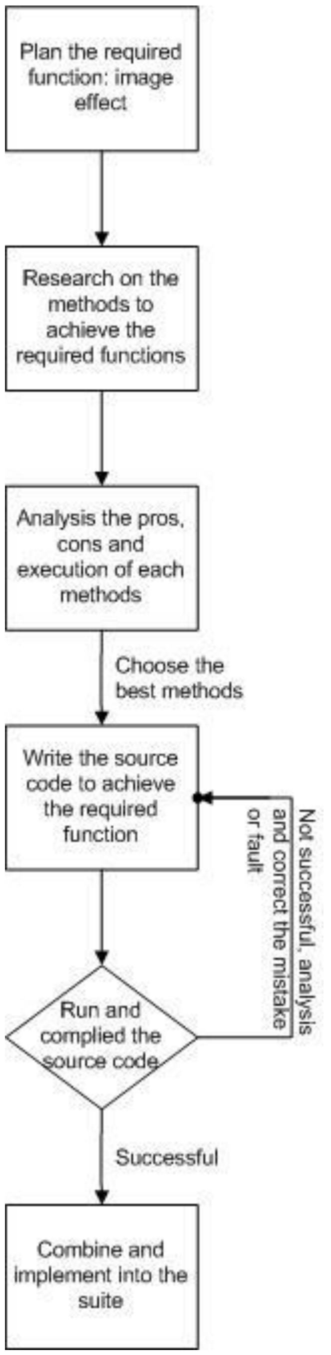

Plan the required function: image effect
Research on the methods to achieve the required functions
Analysis the pros, cons and execution of each methods
Choose the best methods
Write the source code to achieve the required function
Not successful, analysis and correct the mistake or fault
Run and complied the source code
Successful
Combine and implement into the suite

Figure 6: The sequence of tasks related to the development of each video effect

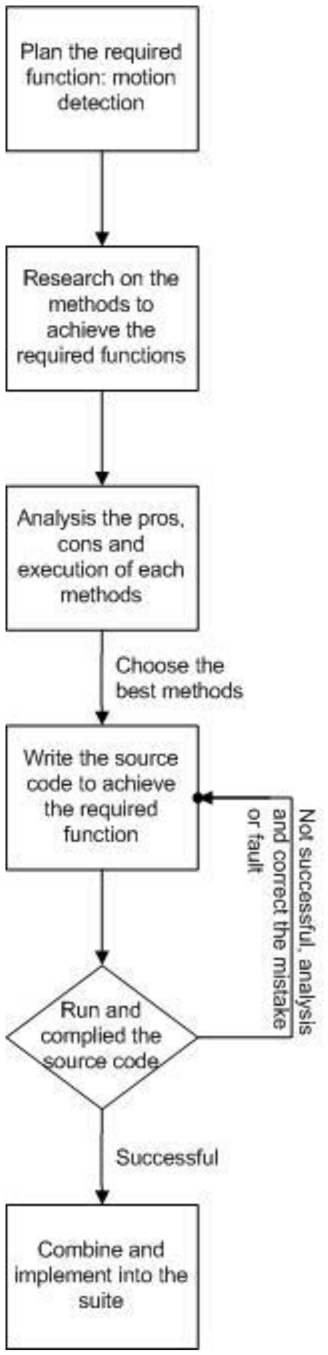

Plan the required function: motion detection
Research on the methods to achieve the required functions
Analysis the pros, cons and execution of each methods
Choose the best methods
Write the source code to achieve the required function
Not successful, analysis and correct the mistake or fault
Run and complied the source code
Successful
Combine and implement into the suite

Figure 7: The sequence of tasks related to the development of the motion detection function

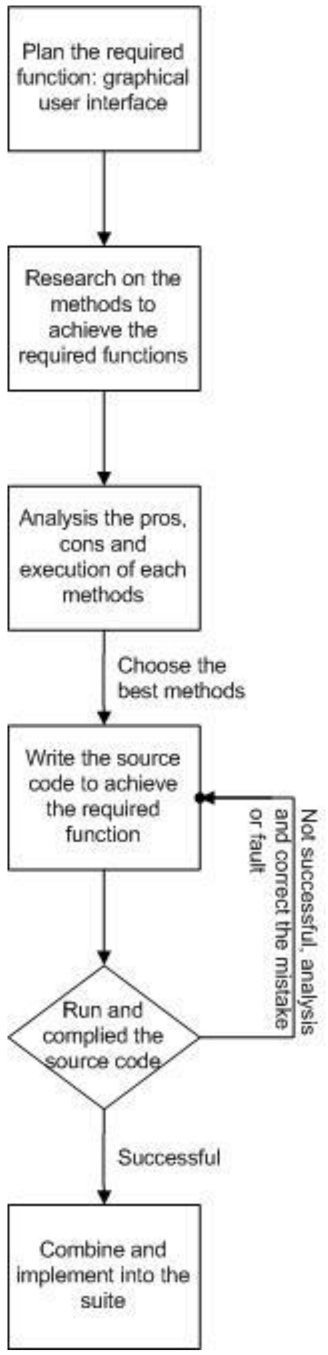

Plan the required function: graphical user interface
Research on the methods to achieve the required functions
Analysis the pros, cons and execution of each methods
Choose the best methods
Write the source code to achieve the required function
Not successful, analysis and correct the mistake or fault
Run and complied the source code
Successful
Combine and implement into the suite

Figure 8: The sequence of tasks related to the function of developing the graphic user interface of the suite

3.2. System Development Methodology

There a few methodologies that are widely being used as a reference for developing a certain system. For this project, a prototyping methodology has been chosen as a guide in designing and developing the video processing suite. The figure below shows the methodology that is being implemented for this project.

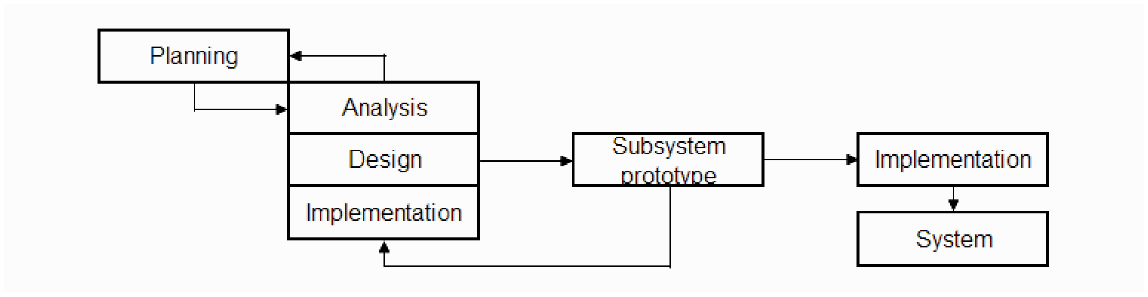


Figure 9: Prototyping methodology for video processing suite

Based on Figure 9 above, the first stage will be planning of the required subsystem that is required by the system. The subsystem is actually representing the class and the system is representing the image or video processing suite or application. For each subsystem, the three stages will be done concurrently.

If the class is having problem in implementation, the class will need to go through the three stages again until the class is successfully implemented. If the class is unable to be implemented, the class is required to be re-planned to find the alternative method of achieving the same required function. If the class is successfully implemented, the class will be implemented into the application and the whole methodology will be repeated again for the rest of the classes.

3.3. Tools And Equipments

Tools represent the equipment, hardware and software needed for the completion of this project. The tools are:

1. Laptop
2. Webcam
3. Java Development Kit and Java Runtime Environment
4. Java Integrated Development Editor
5. Java Media Framework

CHAPTER 4
RESULTS AND DISCUSSION

4.1. Results

This section will be discussing on the graphic user interface (GUI) of the suite and the functions and effects that have been incorporated into the suite. A detailed discussion or explanation on the results will be done in the Discussion section.

4.1.1. Graphic user interface overview

Graphic user interface (GUI) is a human-computer interface that uses windows, icons, buttons and menus, instead of command line interfaces, which relies on user to key in certain command to activate certain functions. [5]

The GUI of this suite consists of 4 main sections that are:

1. The webcam input
   The black box in the 'Webcam Input' section is the area, which the input from the webcam will be display. The input from the webcam will only be displayed after the 'Start Camera' button is being clicked.
2. The captured image
   The black box in the 'Captured Image' section is the area, which will display the image that has been captured from the webcam at the instant that the 'Capture' button is being clicked.

3. The control panel

   In the ‘Control Panel’ section, there are a few tabs and buttons and each button has its functionality and the functionality of each button will be discussed in the Discussion section. This section will contain all the buttons that the user can access to achieve the functions and effects that the suite has.

4. The processed image

   The black box in the ‘Processed Image’ Section is the area, which will display the image that has been manipulated according to the effects that the user has selected. The image in the black area will only be displayed after the button for the image effects, in the control panel is being clicked.

Figure 10 shows the GUI of the suite, which is the same as the one included in the interim report for FYP1. Since then changes have been done towards the suite and the current version of the suite is being shown in Figure 11.

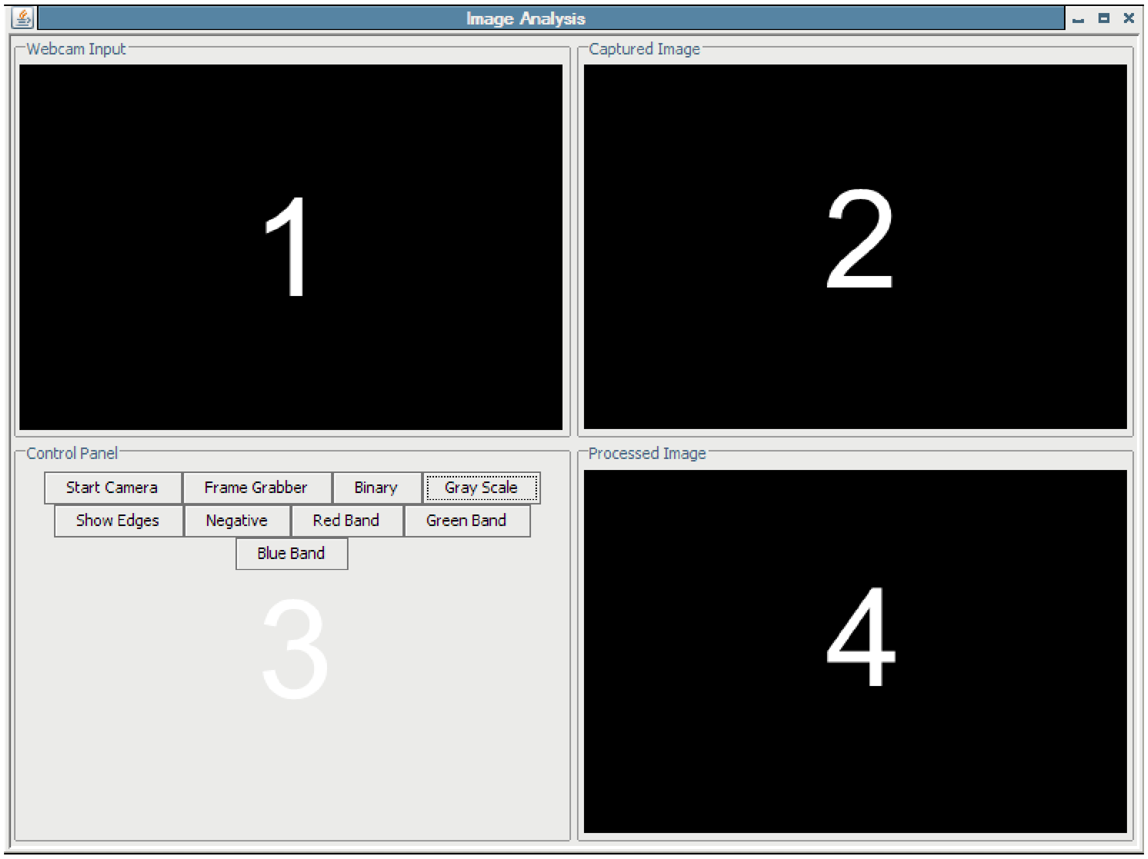


Figure 10: Overview of the previous GUI

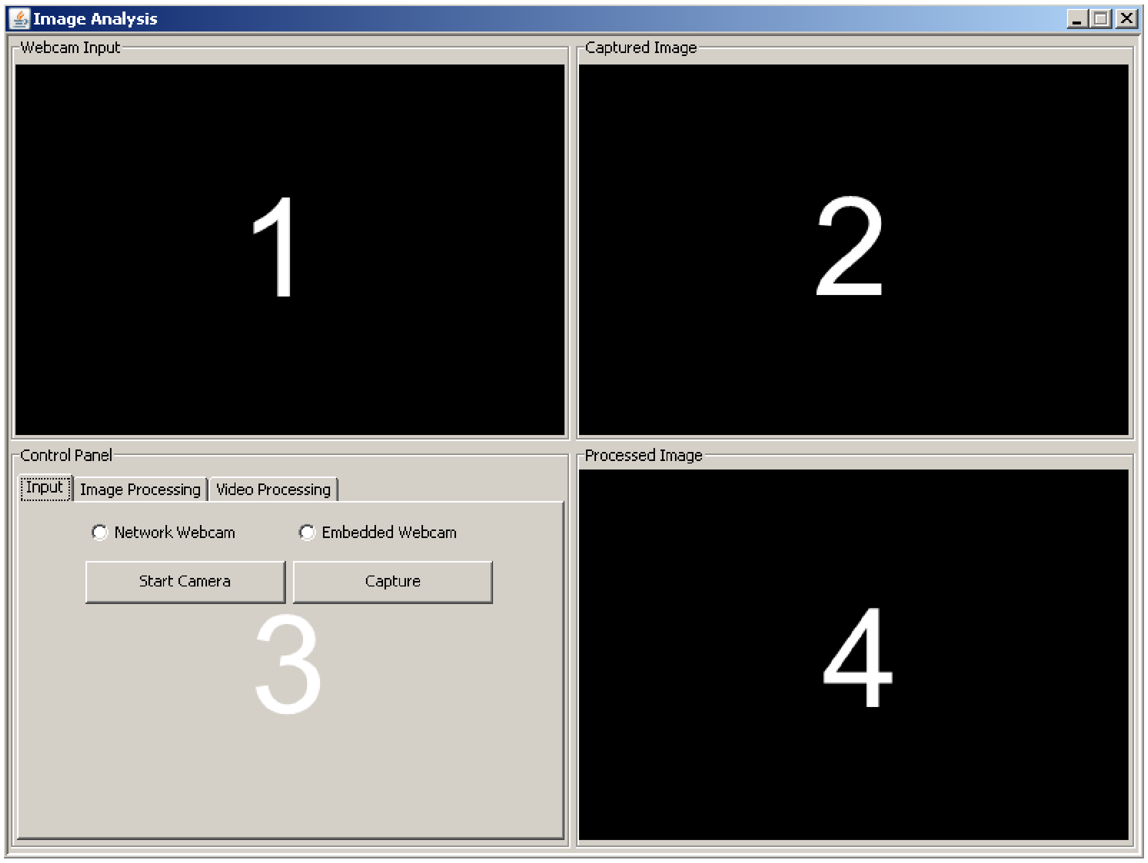


Figure 11: Overview of the current GUI

From the two figures above, it can be seen that changes have been in the Control Panel Section. The current Control Panel consists of the three tabs with each tab has its own grouping of functions that the suite can perform. The three different tabs are:

1. Input

   In the Input tab, there will be options for the input of the suite. Currently the input for the suite will be webcam. However, two different webcam is able to be implemented as input. Firstly, network webcam, which is the webcam located at another computer and the input from that webcam is being stream at a specific address. The input streaming will be detected by the suite and displayed at the Webcam Input Section. Secondly, embedded camera, which is the camera located at the computer that the suite is being operated.

2. Image Processing

   In the Image Processing tab, there will be options for the image manipulation of captured image.

3. Video Processing

   In the Video Processing tab, there will be options for the video manipulation of the input video from the webcam.

The figures below show the contents of each tab in the Control Panel section.

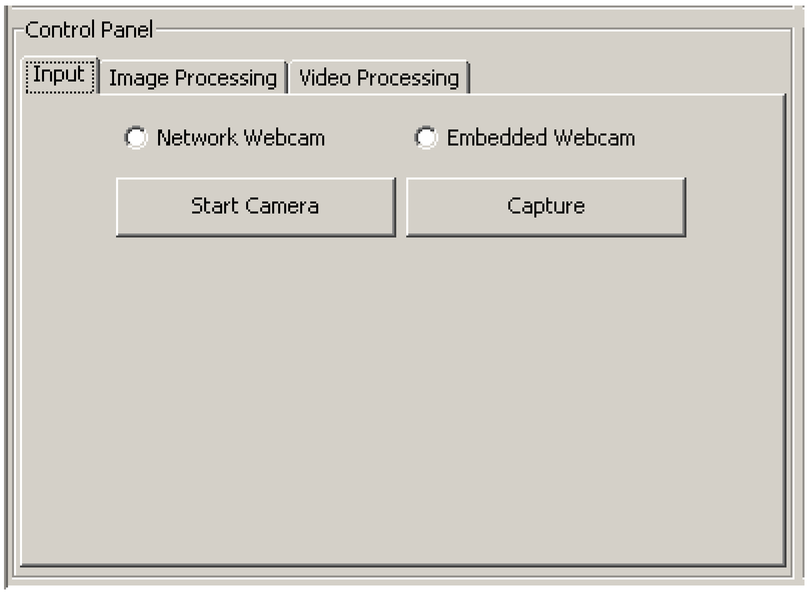


Figure 12: Content of the Input tab

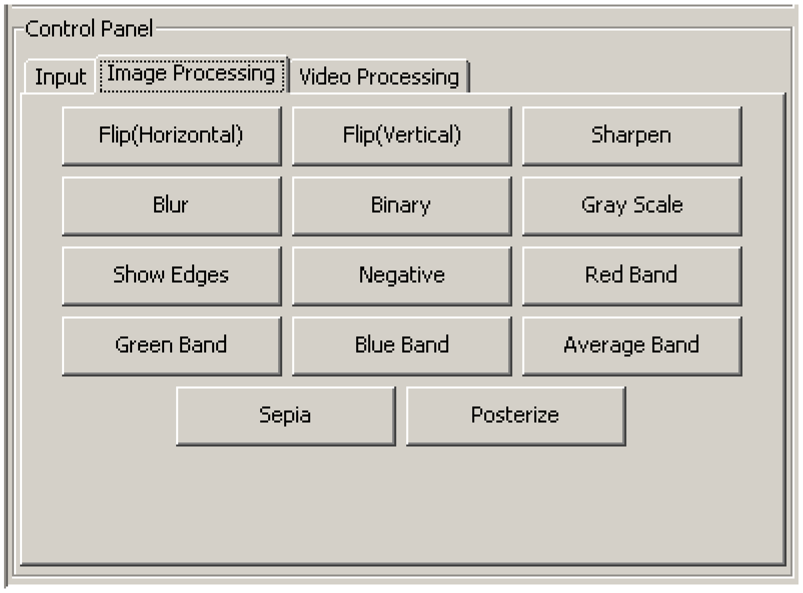


Figure 13: Content of Image Processing tab

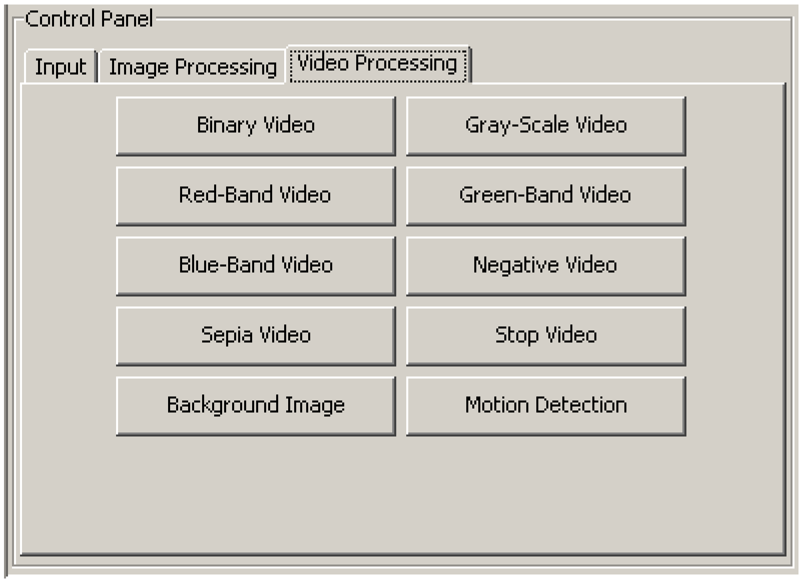


Figure 14: Content of Video Processing tab

4.1.2. Functions of the suite

As being shown in the previous section, in the Control Panel section, there are three tabs with each tab having their own group of functions. In this section, the functions in the Input and Image Processing tab will be discussed and since function for Video Processing have not been developed yet, its functions will be discuss in the next report.

In the Input tab, there are a total of three different functions that the user can access to and they are:

1. Type of webcam

   There are a total of two different webcam that the suite can access to as source of input. Firstly, network webcam, which is the webcam located at another computer and the input from that webcam is being stream at a specific address. The input streaming will be detected by the suite and displayed at the Webcam Input Section. Secondly, embedded camera, which is the camera located at the computer that the suite is being operated.
2. Start Camera

   By clicking at this button, it will enable the suite to display the input from the selected webcam at the Webcam Input section.
3. Capture

   By clicking at this button, it will enable the suite to capture the input from the webcam and displayed it at the Captured Image section.

4.1.3. Image effects of the suite

In the Image Processing tab, there are a total of 14 image manipulation effects that have been successfully implemented into the suite. From the 14 effects, a total of 7 effects have been implemented in the previous version of the suite and the other 7 effects has been developed and implemented into the suite since then. The image effects that have been successfully implemented into the suite are:

1. Horizontal image flip
2. Vertical image flip
3. Sharpen image
4. Blur image
5. Binary image
6. Gray-scale image
7. Show edges
8. Negative image
9. Red band image
10. Green band image
11. Blue band image
12. Average band image
13. Sepia image
14. Posterize image

The figures below show the execution of the suite from the initial start-up of the suite and the functions and effects that the suite can achieve.

The figure below shows the initial start-up of the suite.

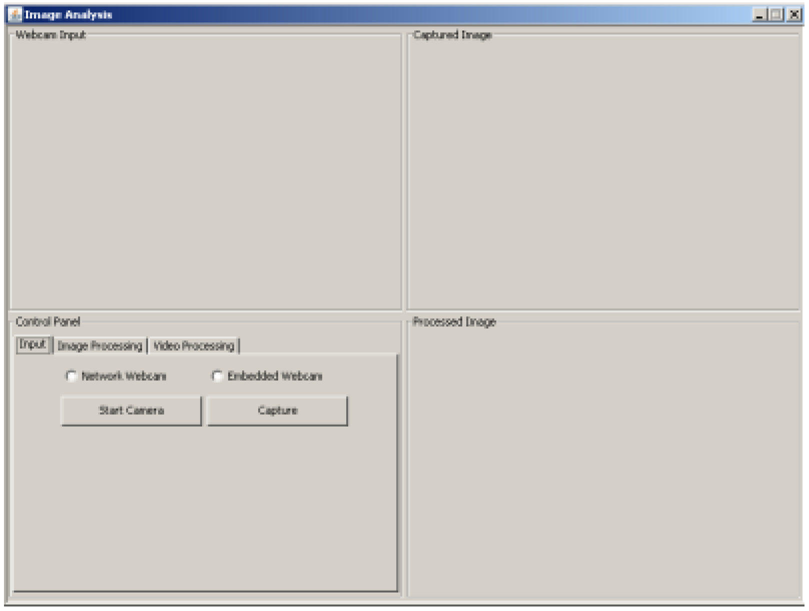


Figure 15: Initial start-up of the suite

After the start-up is complete, the user will need to choose the type of webcam that the suite will access as its input and click on the Start Camera button. After the button has been clicked, the visual from the webcam will be displayed in the Webcam Input section, as shown in the figure below.

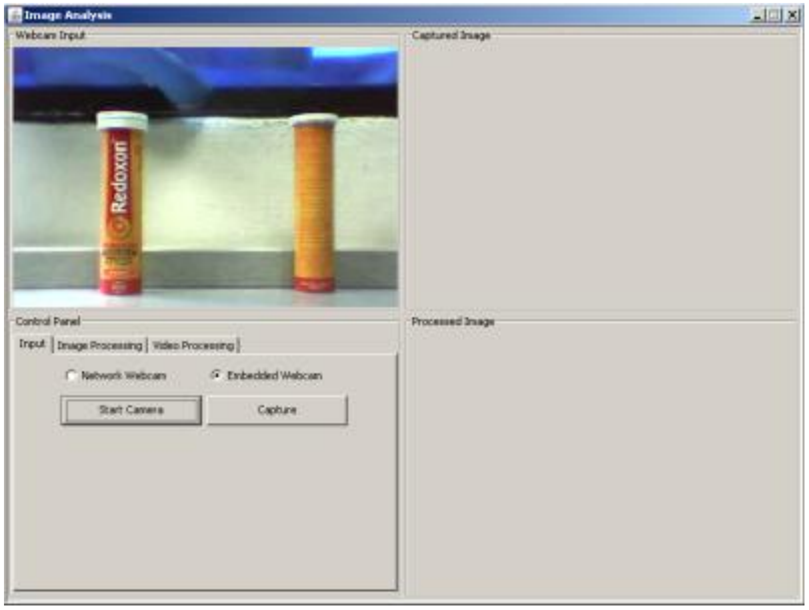


Figure 16: Webcam input initialized

After the input from the webcam is successfully being displayed at the Webcam Input section, the user will be able to choose the instant to capture the input from the webcam input by clicking the Capture button and the captured image will be displayed in the Captured Image section and this is being shown in the figure on the next page.

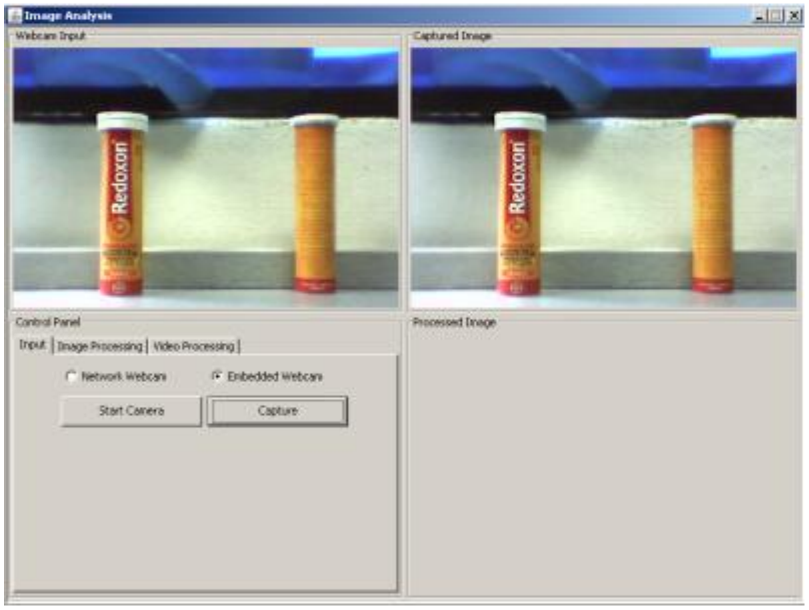


Figure 17: Captured image initialized

After image has successfully being captured from the webcam, the user will be able to choose the image effects to be applied to the captured image and the image that has been applied with the selected effect will be displayed in the Processed Image section. The figures after this will show the manipulated image for each effect.

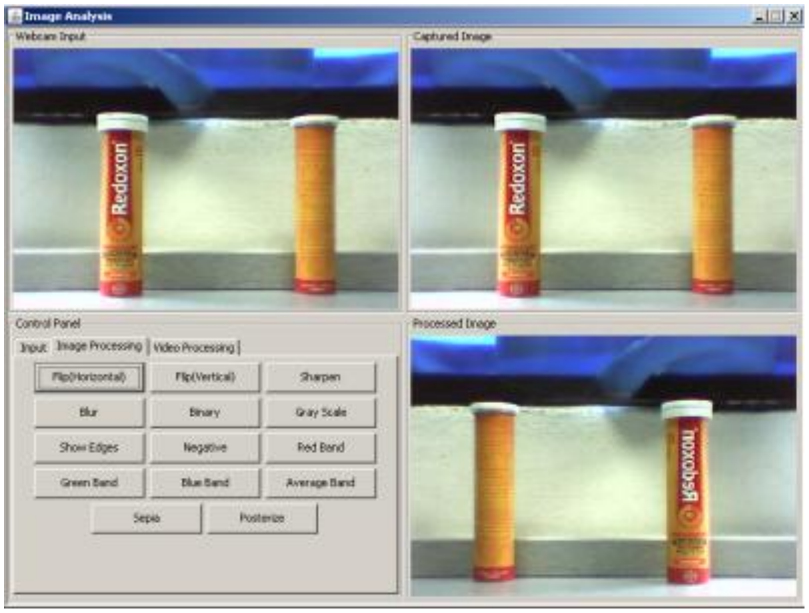


Figure 18: Horizontal image flip effect initialized

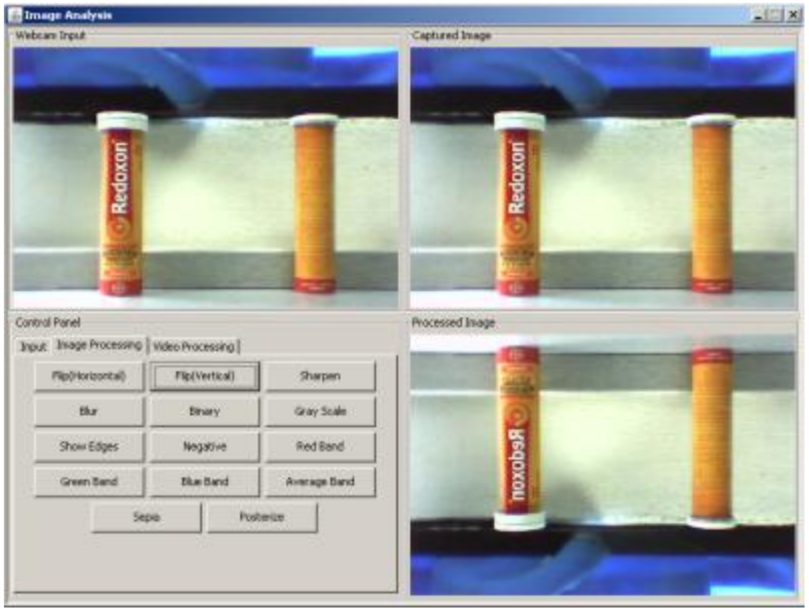


Figure 19: Vertical image flip effect initialized

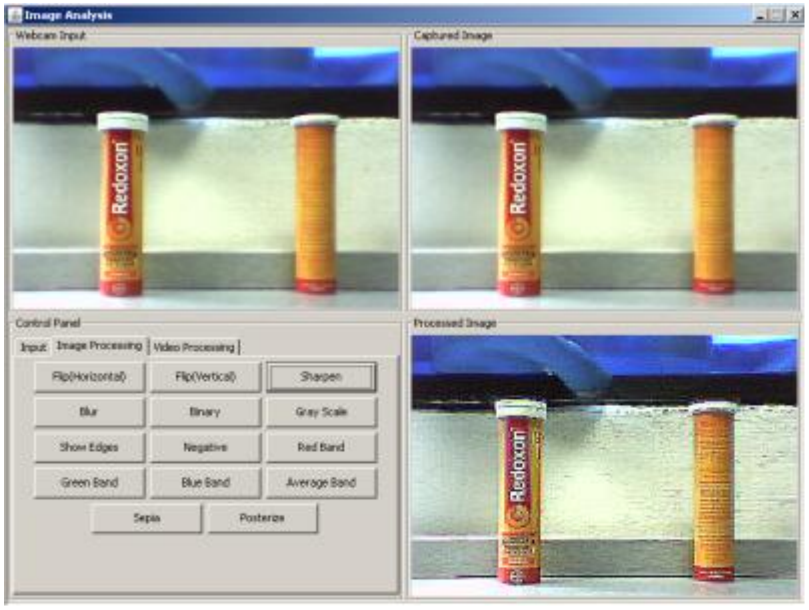


Figure 20: Sharpen image effect initialized

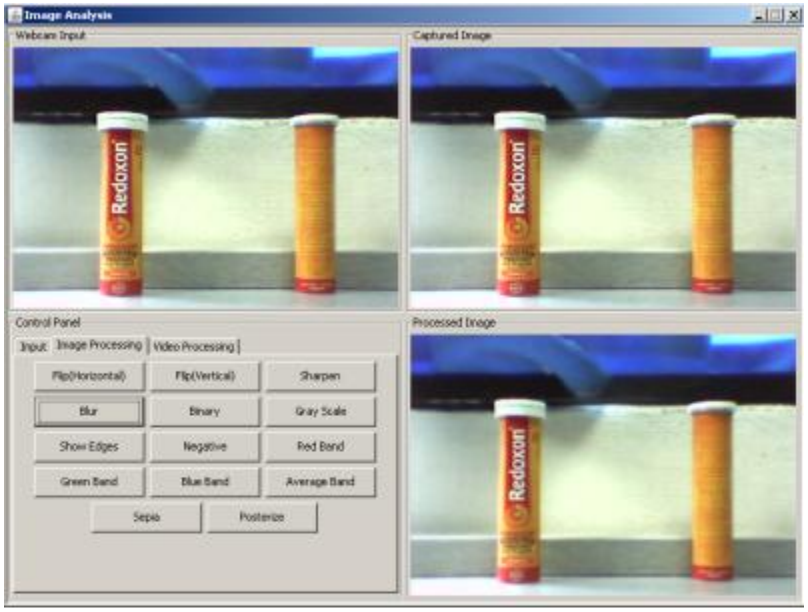


Figure 21: Blur image effect initialized

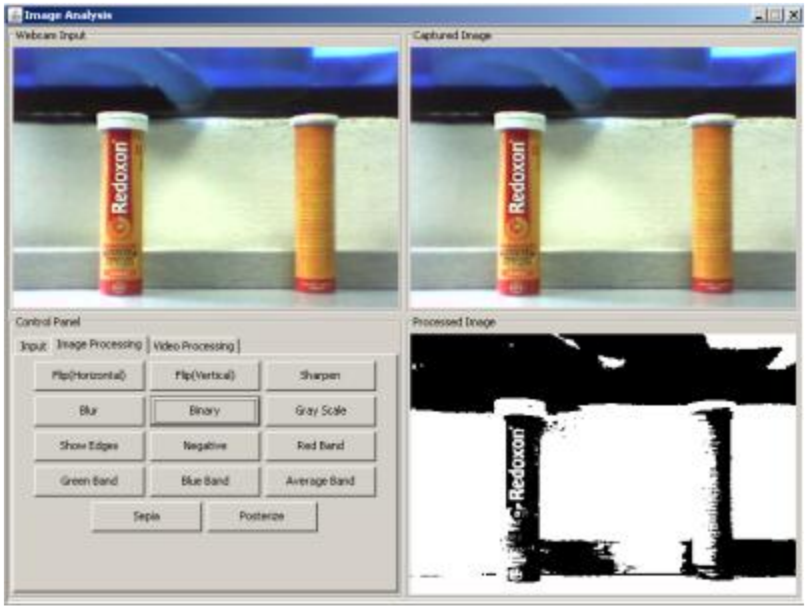


Figure 22: Binary image effect initialized

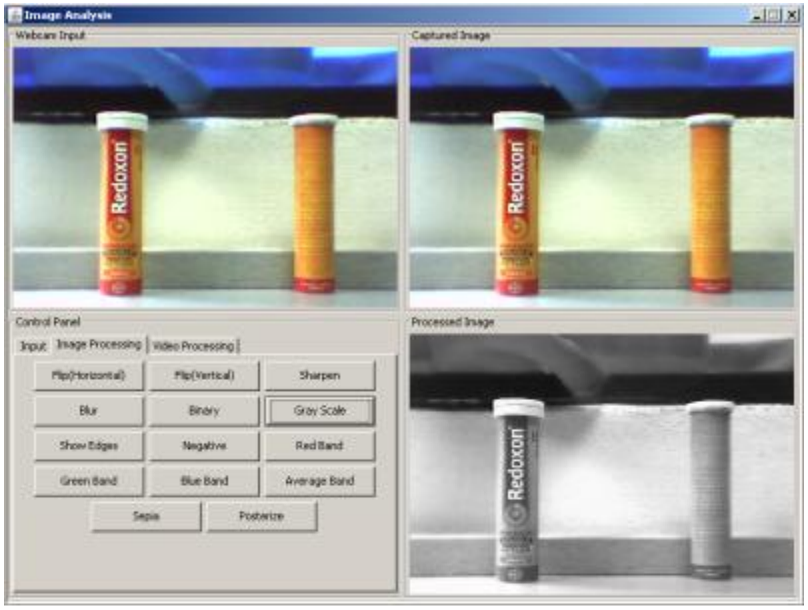


Figure 23: Gray-scale image effect initialized

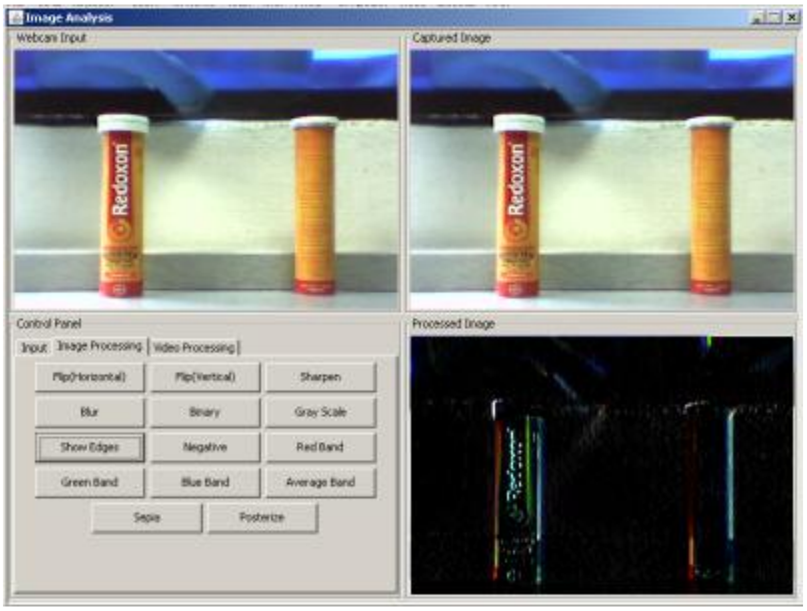


Figure 24: Show edges effect initialized

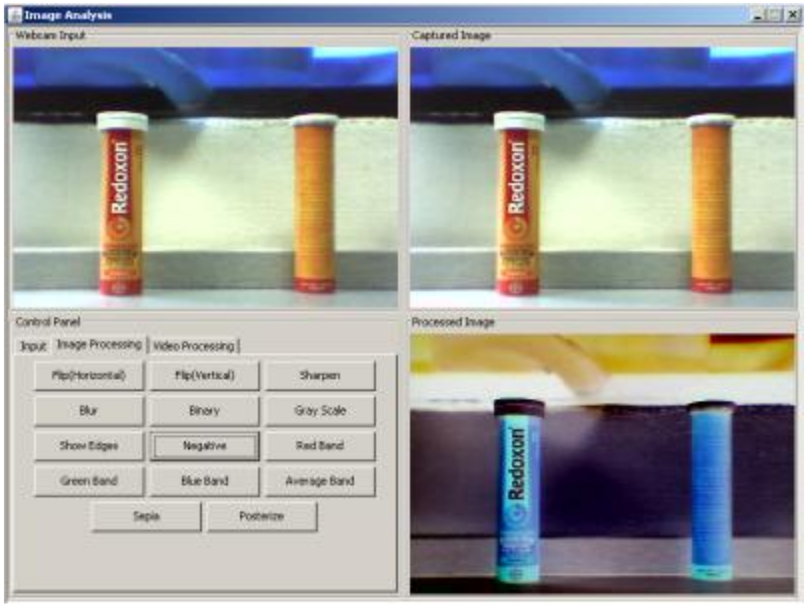


Figure 25: Negative image effect initialized

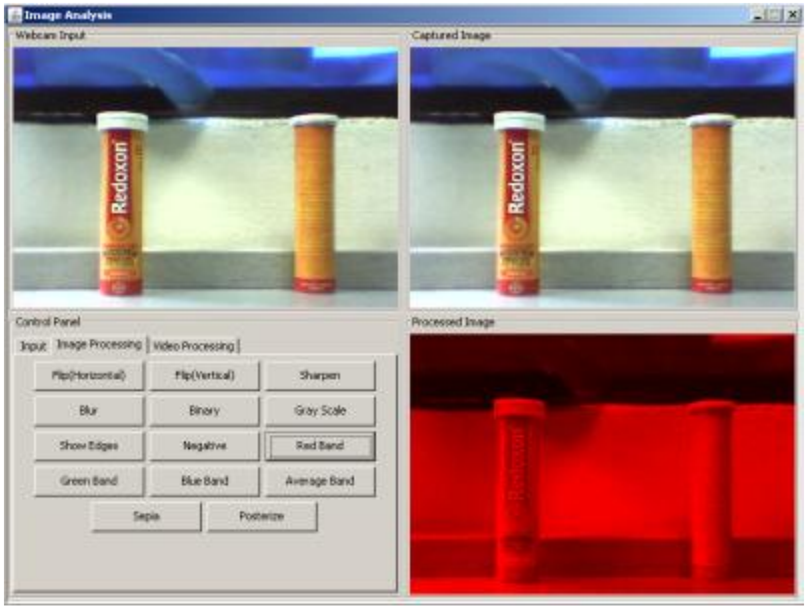


Figure 26: Red band image effect initialized

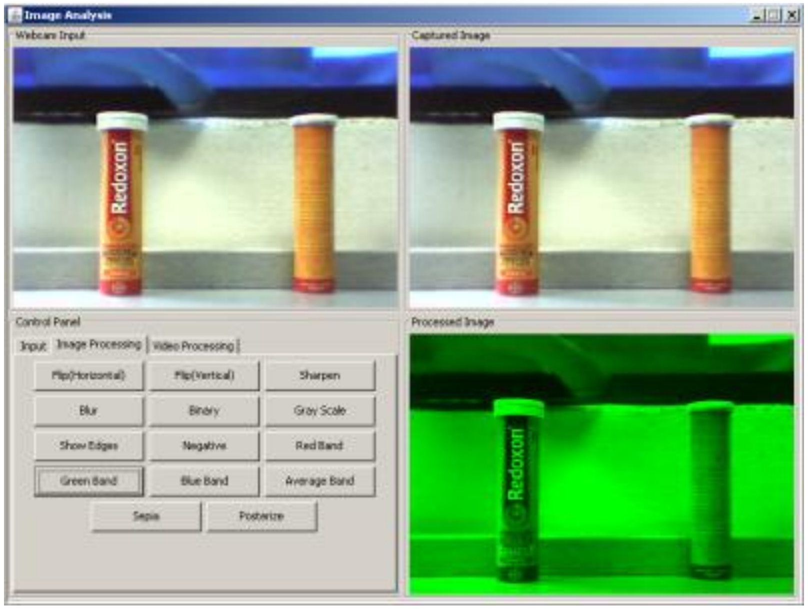


Figure 27: Green band image effect initialized

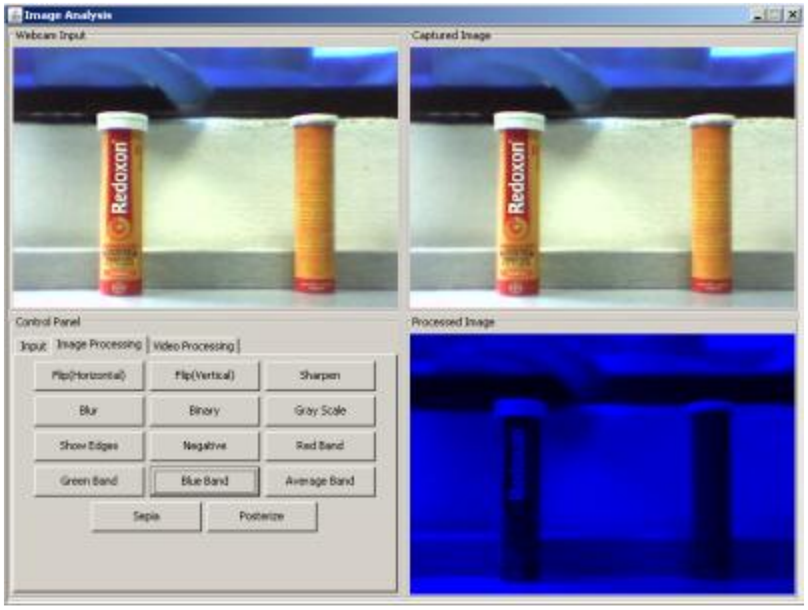


Figure 28: Blue band image effect initialized

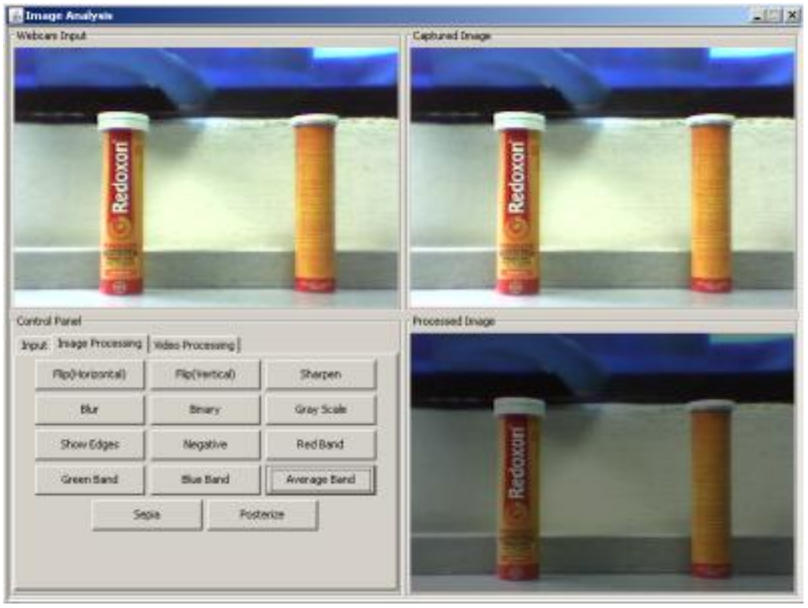


Figure 29: Average band image effect initialized

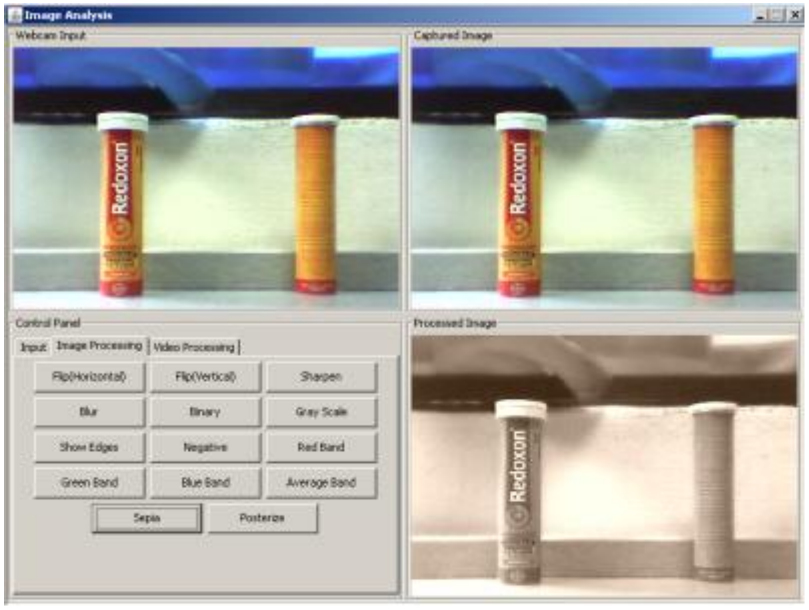


Figure 30: Sepia image effect initialized

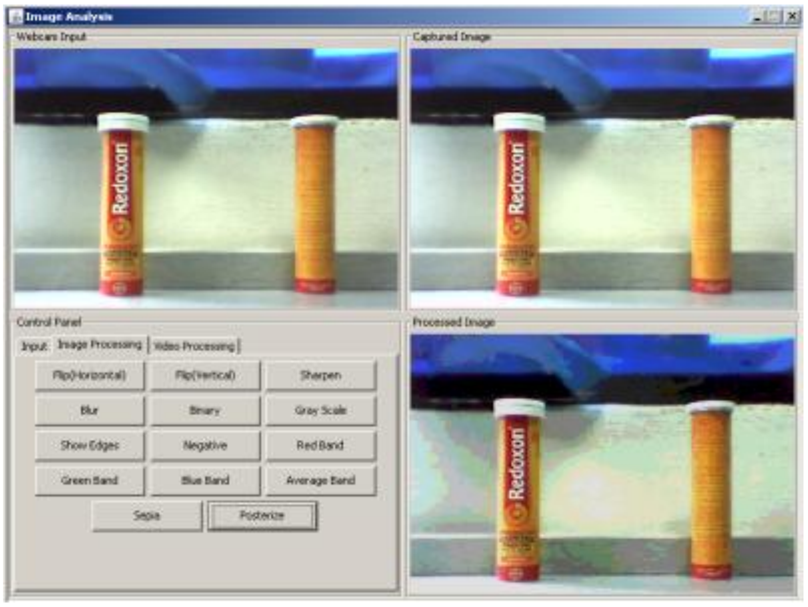


Figure 31: Posterize image effect initialized

4.1.4. Video effects of the suite

In the Video Processing tab, as seen in Figure 14, a total of 8 video effects have been included into the suite and an additional two functions button have been included to control the execution and process of the video effects. The video effects are:

1. Binary video
2. Gray-scale video
3. Red-band video
4. Green-band video
5. Blue-band video
6. Negative video
7. Sepia video
8. Motion detection

The additional function buttons are:

1. Stop video
2. Background image

As can be seen from the list of video effects above, it can be seen that it is the same effects that have been implemented in the image processing section, expect for the motion detection function. Due to that reason, the example for the effects numbered from 1 to 7, will not be shown or discuss. More attention will be given to the motion detection function.

For the two additional function buttons, firstly, the ‘Stop Video’ button has the functionality to stop the execution of the selected video effect. Before the user changes from one video effect to another, the user has to stop the execution of the previous effects and continue with the execution of the required effect. Secondly, the ‘Background Image’ button has the functionality to capture the image from the webcam, to be used as a source of reference for the motion detection function.

For the motion detection function, this function is achieved through the implementation of background subtraction method. Due to that reason, the suite requires an initial image, which is the background image of the environment that the function will be performed. With the reference of background image, the suite will be able to detect any foreign object that enters to the environment, which is being represented by the background image. The figures in the next pages show the execution of the motion detection function of the suite.

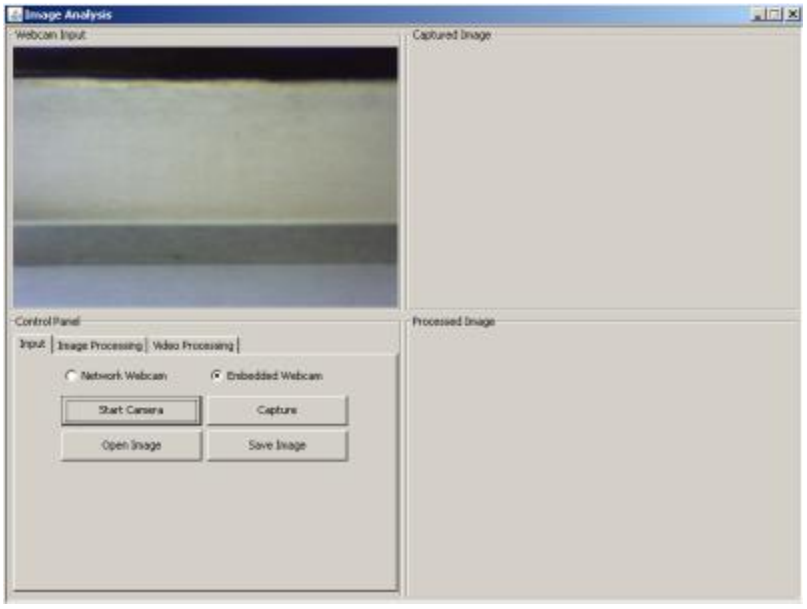


Figure 32: Webcam input initialized

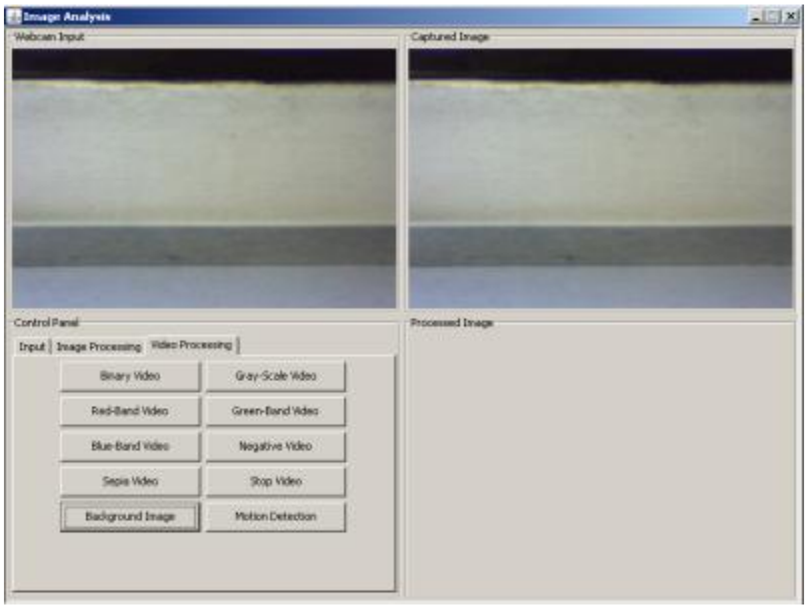


Figure 33: Background image captured

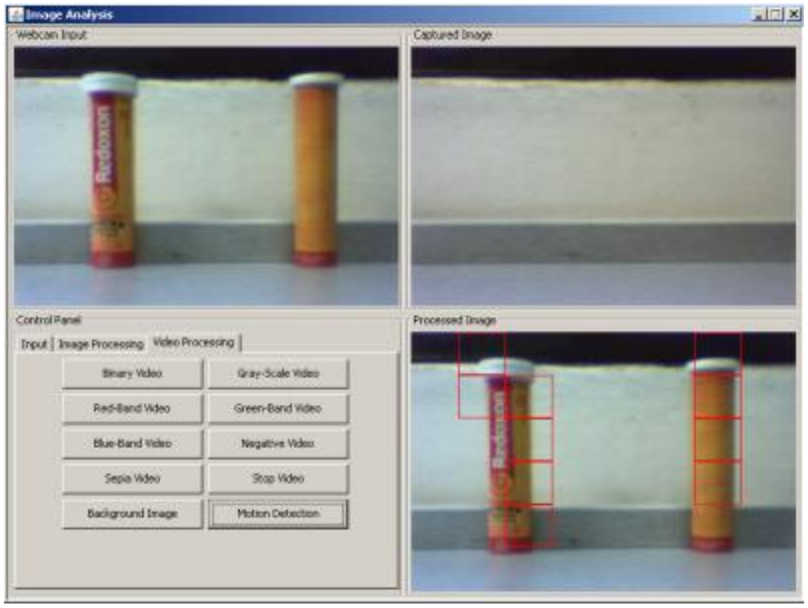


Figure 34: Motion detection initialized

4.2. Discussion

This section will discuss on the functions and effects that have been implemented into the suite.

Firstly, the addition of tabs into the Control Panel section, this is being done to organized the functions and effects that the suite have into a more pleasant and organized manner. Since each tab will have a group of related functions or effects in it, it will further simply or ease the user to find the required functions or effects that the user would like to implement.

There are two type of webcam that can be accessed as the input for the suite that are embedded webcam and network webcam. For the network webcam, the user will to transmit the webcam at the computer at a specific address, which has been hardcoded into the suite. The transmission can be done using 'jmstudio'. The transmission of video is being achieved through the use of real-time transfer protocol. [6] For the embedded webcam, the webcam can be used as the input for the suite as long as the webcam at the computer is detected and the driver for the webcam is being installed. This is being achieved through directing the suite to receive the input from the webcam through the internal address of the webcam, which is being connected to the computer. [7]

In order for the suite to be able to use the input from the webcam as the image input, the video from webcam will be captured and stored as bufferedimage. Since video is a sequence of images, a frame grabber function is being used to capture the frame of image at the required instant and use it as the input for the image manipulation and processing. [7]

For the effects, there are specific ways to achieve the effects. For the binary and gray-scale image, the captured image from the webcam will be altered its properties in term of its colour. For binary image, the image will be created in type byte binary [8] and for the gray-scale image; the image will be created in type byte gray. [9] For the vertical and horizontal image flip, the processed image that is being created, will be specified the orientation and coordinate, which the image will be drawn and this will lead to the horizontal and vertical flip image. [10]

For the show edges, blur and sharpen effects, the captured image will be passed through a filter. The filter will convolve each pixel of the picture with a specific 3x3 matrix for each effects and resulting on the processed image that fulfil the selected effects. [11] The convolution process that has been used in achieving the effects is a spatial operation, which multiply the surround of the input pixel with a specific 3x3 matrix. This method will allow the output pixel to be affected by the immediate neighbour in a way that can only be mathematically specified by the value of the matrixes that have been used. [12] The matrixes values are:

Matrix for show edges effect $= \begin{bmatrix} 1.0f & 1.0f & 1.0f \\ 0.0f & 0.0f & 0.0f \\ -1.0f & -1.0f & -1.0f \end{bmatrix}$

Matrix for blur effect $= \begin{bmatrix} 0.0625f & 0.125f & 0.0625f \\ 0.125f & 0.25f & 0.125f \\ 0.0625f & 0.125f & 0.0625f \end{bmatrix}$

Matrix for sharpen effect $= \begin{bmatrix} -1.0f & -1.0f & -1.0f \\ -1.0f & 9.0f & -1.0f \\ -1.0f & -1.0f & -1.0f \end{bmatrix}$

For the red band, green band, blue band and average band effect, the captured image will be passed through a filter, which will filter out or changes the intensity value of the colours. [13] For these effects, only the colour properties of the image is being altered, which in the Java Programming Language, the raster of the image is being altered. For red band, green band and blue band image, only the required colour will not be filtered out by the filter and for the average band image, the intensity value of each colours will be reduced into half. The colour properties of the image or raster, is a 3x3 matrix, which the first column is representing the colour red, the second column representing the colour green and the third column is representing the colour blue. The matrix below shows the values for each of the effects:

Matrix for red band effect $= \begin{bmatrix} 1.0f & 0.0f & 0.0f \\ 0.0f & 0.0f & 0.0f \\ 0.0f & 0.0f & 0.0f \end{bmatrix}$

Matrix for green band effect $= \begin{bmatrix} 0.0f & 0.0f & 0.0f \\ 0.0f & 1.0f & 0.0f \\ 0.0f & 0.0f & 0.0f \end{bmatrix}$

Matrix for blue band effect $= \begin{bmatrix} 0.0f & 0.0f & 0.0f \\ 0.0f & 0.0f & 0.0f \\ 0.0f & 0.0f & 1.0f \end{bmatrix}$

Matrix for average band effect $= \begin{bmatrix} 0.5f & 0.0f & 0.0f \\ 0.0f & 0.5f & 0.0f \\ 0.0f & 0.0f & 0.5f \end{bmatrix}$

For negative image, the captured image will be passed through a filter, which will invert the colour value of each pixel, from high value to low value, which will lead to a tonal inversion of the image. [14]

For the sepia image effect, the colour properties are being changed to achieve a brownish coloured image. [15] The formula below shows the alterations that are being made on the intensity value of the colour properties of the image, which is the raster.

Gray Value	= (Red Value + Blue Value + Green Value)/3
Red Value	= Green Value = Blue Value = Gray Value
Red Value $_{new}$	= Red Value $_{old}$ (Gray Value) + 40
Green Value $_{new}$	= Green Value $_{old}$ + 20

For any intensity value that below 0, the value will be changed to 0 and any intensity value, which exceeds 255, will be changed to 255. The addition that has been made to the intensity value of the red and green colour is to achieve the brownish colour of the image.

For posterize image effects, the image is being passed through a filter that will alter the intensity value of each pixel of the picture, to achieve the oil paint like image. The formula below is being used to alter the intensity value of each pixel for the posterize image effect:

Intensity Value $_{new}$	= Intensity Value $_{old}$ – Remainder
Remainder	= Remainder of (Intensity Value $_{old}$ /32)

The intensity value ranges from 0 to 255. [16]

For the video effects that have been implemented into the suite, except for the motion detection function, the video effects is being achieved through sequencing the image effects in a continuous manner, which will leads to the achievement of video effects. The sequencing is being achieved through the usage of timer functionality in Java Programming Language. [17]

For the motion detection function, it is required to have an initial representation of the environment that the function needs to be implemented, which is the background image. After the background image is captured, the image will be converted into gray scale image and divided into smaller segments. Then, the intensity value of each segment is being determined.

When the motion detection function is initiated, the current frame of the video will be processed by the suite, converting it into gray scale image, dividing it into same segments as the background image and determining the intensity value of the segments.

When the intensity value of the background image and current frame is known, the absolute value of the difference of both of the intensity value is calculated and the segment, in which the difference of the intensity value exceeded the predefined value, will be boxed to highlight the object that appears on it. [18] This function is being achieved through the usage of the background subtraction method.

CHAPTER 5
CONCLUSION AND RECOMMENDATION

5.1. Conclusion

A simple image and video processing suite is able to be developed through the use of Java Programming Language. The suite is able to receive and display input from the webcam, use the webcam as an image and video source for the webcam and also able to manipulate the image and video input from the webcam. There are a total of 14 image effects that have been incorporated into the suite, which focus on the manipulation of the colours and properties of the image. For the video effects, there are a total of 8 video effects that been added and out of the 8 effects, which includes motion detection function. To achieve the motion detection function, the background subtraction method is being used.

5.2. Recommendations

For future development and improvement of this project, it is suggested that an offline image upload is being added into the suite. This will add the versatility of the suite in working with image and video inputs, besides the one from the webcam. Another recommendation is that a save function is being added to the suite to save the image and video that have been manipulated or processed. Besides that, another further improvement that can be done is the motion detection function. The method, which is being used in the suite doesn’t take into the account of changing light condition and small objects that maybe come into the background image. Due to that reason, it is suggested that a better and smarter algorithm is being develop in the future to take into account of the dynamic factor of the environment.

REFERENCES

[1] Orenstein D. (2000). Quick Study: Application Programming Interface (API). Retrieved on 12th of March, 2009, from Computer World website
Website:
http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=43487

[2] Mihalenko P. (2007). Learn what the Java 2D API graphics package offers. Retrieved on 12th of March, 2009, from ZD Net Asia website
Website: http://www.zdnetasia.com/techguide/java/0,39044898,62035312,00.htm

[3] FAQ: Java Advanced Imaging API,
Retrieved on 11th of March, 2009, from Java Sun website
Website: http://java.sun.com/products/java-media/jai/forDevelopers/jaifaq.html#fileformat

[4] Java SE Desktop Technologies
Retrieved on 17th of April, 2009, from Java Sun website
Website: http://java.sun.com/javase/technologies/desktop/media/jmf/

[5] GUI definition
Retrieved on 22nd of April, 2009
Website: http://www.linfo.org/gui.html on 22 april 2009-04-22

[6] Receiving Audio and Video Using RTP
Retrieved on 1st of December 2009, from Sun Development Network
Website:
http://java.sun.com/javase/technologies/desktop/media/jmf/2.1.1/solutions/AVReceive.html

[7] Java Media Framework - Re: Here is the source code to JMF Webcam app + saves jpeg

Retrieved on 23th of October 2009, from forums.sun.com

Website: http://forums.sun.com/thread.jspa?threadID=247253&start=0@tstart=0

[8] Class Bufferedimage

Retrieved on 23th of October 2009, from java.sun.com

Website: http://java.sun.com/j2se/1.5.0/docs/api/java/awt/image/BufferedImage.html

[9] Creating Gray-Level Images

Retrieved on 23th of October 2009, from Java Image Processing Cookbook

Website: http://www.lac.inpe.br/~rafael.santos/JIPCookbook/1200-create-gl.jsp

[10] Ultimate Java Image Manipulation

Retrieved on 23th of October 2009, from Javalobby

Website: http://www.javalobby.org/articles/ultimate-image/

[11] Show Other Image Effects

Retrieved on 23th of October 2009, from RoseIndia website

Website: http://www.roseindia.net/java/example/java/swing/graphics2D/other-image.shtml

[12] Class ConvolveOp

Retrieved on 1st of December 2009, from Java 2 Platfrom SE v1.4.2

Website: http://java.sun.com/j2se/1.4.2/docs/api/java/awt/image/ConvolveOp.html

[13] Show Image Effects

Retrieved on 23th October 2009, from RoseIndia website

Website: http://www.roseindia.net/java/example/java/swing/graphics2D/image-effect.shtml

[14] Color Effect On Image

Retrieved on 23th October 2009, from RoseIndia website

Website: http://www.roseindia.net/java/example/java/swing/color-effect-image.shtml

[15] Sepia tone image filter for Java

Retrieved on 19th July 2009, from comp.lang.java.programmer Google Group

Website:

http://groups.google.com/group/comp.lang.java.programmer/browse_thread/thread/9d20a72c40b119d0

[16] Image Demo

Retrieved on 23th of October 2009, from Java2s website

Website: http://www.java2s.com/Code/Java/2D-Graphics-GUI/Imagedemo.htm

[17] Class Timer

Retrieved on 23th of October 2009, from java.sun.com

Website: http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/Timer.html

[18] Java Image Comparison/ Motion Detection

Retrieved on 23th of October 2009, from Mind Meat

Website: http://mindmeat.blogspot.com/2008/07/java-image-comparison.html

APPENDICES

APPENDIX 1

SOURCE CODE OF THE SUITE

In this section, the source code of the suite will be included.

```
package suite;

// Import from JAVA
import java.net.MalformedURLException;
import javax.swing.border.*;
import javax.media.*;
import javax.media.Buffer;
import javax.media.control.FrameGrabbingControl;
import javax.media.format.VideoFormat;
import javax.media.util.BufferToImage;
import java.awt.event.*;
import javax.swing.*;
import javax.imageio.*;
import java.io.*;
import java.util.*;
import java.awt.*;
import java.awt.image.*;
import com.sun.image.codec.jpeg.*;

public class Video extends JFrame implements ControllerListener
{
		/**
		 *
		 */
		private static final long serialVersionUID = 1L;

		// Variables declaration
		private JPanel contentPane;
		//-----
		private JPanel jPanel1;
		//-----
		private JPanel jPanel2;
		//-----
		private JButton jButton2;
		private JButton jButton3;
		private JButton jButton4;
		private JButton jButton5;
		private JButton jButton6;
		private JButton jButton7;
		private JButton jButton8;
		private JButton jButton9;
		private JButton jButton10;
		private JButton jButton11;
		private JButton jButton12;
		private JButton jButton13;
		private JButton jButton14;
		private JButton jButton15;
		private JButton jButton16;
		private JButton jButton17;
		private JButton jButton18;
		private JButton jButton19;
		private JButton jButton20;
		private JButton jButton21;
		private JButton jButton22;
		private JButton jButton23;
		 private JButton jButton24;
		private JButton jButton25;
		private JButton jButton26;
		private JButton jButton27;
		private JButton jButton28;
		 private JButton jButton29;
private JPanel jPanel3;
//-----
private JPanel jPanel4;
private JPanel FrameCapturePanel;
private JPanel FrameProcess;
private JPanel VideoPanel;
 //tabbed
private JPanel InputPanel;
private JPanel InputPanel0;
 private JPanel InputPanel1;
private JTabbedPane tab1;
//radiobutton
private JRadioButton radiobutton1;
private JRadioButton radiobutton2;

// RTP Variable Declaration
private String rtpAdd   = new String();
private MediaLocator mlr = null;

private  Player player   = null;
// End of RTP Variable Declaration;

// FrameGrabber Variable Declaration
private FrameGrabbingControl frameGrabbingControl= null;
private Buffer buffer = new Buffer();
private Image bufferedToImage = null;
private BufferToImage bufferToImage = new BufferToImage ((VideoFormat)buffer.getFormat());
private Image capturedImage;
private BufferedImage photo;
private BufferedImage capturedphoto;
private BufferedImage backgroundimage;

// End of FrameGrabber Variable Declaration;
//test show edges
private BufferedImage photo1;
private BufferedImage photo2;

Raster raster;
WritableRaster writableRaster;

 public javax.swing.Timer timerbinary;
public javax.swing.Timer timergray;
public javax.swing.Timer timerred;
public javax.swing.Timer timergreen;
 public javax.swing.Timer timerblue;
public javax.swing.Timer timernegative;
public javax.swing.Timer timersepia;
public javax.swing.Timer timerdetection;

public Video()
{
                super();
                initializeComponent();

                setUrlAdd();
                setMediaLocator();

                this.setVisible(true);
}
```

```
/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always regenerated
 * by the Windows Form Designer. Otherwise, retrieving design might not work properly.
 * Tip: If you must revise this method, please backup this GUI file for JFrameBuilder
 * to retrieve your design properly in future, before revising this method.
 */
private void initializeComponent()
{
contentPane = (JPanel)this.getContentPane();
//-----
jPanel1 = new JPanel();
//-----
jPanel2 = new JPanel();
//-----
jButton2 = new JButton();
jButton2.setPreferredSize(new Dimension(140,30));
jButton3 = new JButton();
 jButton3.setPreferredSize(new Dimension(140,30));
jButton4 = new JButton();
jButton4.setPreferredSize(new Dimension(110,30));
jButton5 = new JButton();
jButton5.setPreferredSize(new Dimension(110,30));
jButton6 = new JButton();
jButton6.setPreferredSize(new Dimension(110,30));
jButton7 = new JButton();
jButton7.setPreferredSize(new Dimension(110,30));
jButton8 = new JButton();
jButton8.setPreferredSize(new Dimension(110,30));
jButton9 = new JButton();
jButton9.setPreferredSize(new Dimension(110,30));
jButton10 = new JButton();
jButton10.setPreferredSize(new Dimension(110,30));
jButton11 = new JButton();
jButton11.setPreferredSize(new Dimension(110,30));
jButton12 = new JButton();
jButton12.setPreferredSize(new Dimension (110,30));
jButton13 = new JButton();
jButton13.setPreferredSize(new Dimension (110,30));
jButton14 = new JButton();
jButton14.setPreferredSize(new Dimension (110,30));
jButton15 = new JButton();
jButton15.setPreferredSize(new Dimension(140,30));
jButton16 = new JButton();
jButton16.setPreferredSize(new Dimension(110,30));
jButton17 = new JButton();
jButton17.setPreferredSize(new Dimension(110,30));
jButton18 = new JButton();
jButton18.setPreferredSize(new Dimension(110,30));
jButton19 = new JButton();
jButton19.setPreferredSize(new Dimension(140,30));
jButton20 = new JButton();
jButton20.setPreferredSize(new Dimension(140,30));
jButton21 = new JButton();
jButton21.setPreferredSize(new Dimension(140,30));
jButton22 = new JButton();
jButton22.setPreferredSize(new Dimension(140,30));
jButton23 = new JButton();
jButton23.setPreferredSize(new Dimension(140,30));
jButton24 = new JButton();
jButton24.setPreferredSize(new Dimension(140,30));
jButton25 = new JButton();
jButton25.setPreferredSize(new Dimension(140,30));
jButton28 = new JButton();
jButton28.setPreferredSize(new Dimension(140,30));
jButton29 = new JButton();
```

```
jButton29.setPreferredSize(new Dimension(140,30));
//radio button
radiobutton1 = new JRadioButton("Embedded Webcam", false);
radiobutton1.setPreferredSize(new Dimension(140,30));
radiobutton2 = new JRadioButton("Network Webcam", false);
radiobutton2.setPreferredSize(new Dimension(140,30));

jPanel3 = new JPanel();
//-----
Panel4 = new JPanel();
//-----
FrameCapturePanel = new JPanel();
FrameProcess = new JPanel();
VideoPanel = new JPanel();
//tab
InputPanel = new JPanel();
 InputPanel0 = new JPanel();
InputPanel1 = new JPanel();
Tab1 = new JTabbedPane();
tab1.setPreferredSize(new Dimension(383,257));

//
// contentPane
//
contentPane.setLayout(new GridLayout(2, 2, 0, 0));
contentPane.add(jPanel1, 0);
contentPane.add(jPanel2, 1);
contentPane.add(jPanel3, 2);
contentPane.add(jPanel4, 3);
contentPane.setBorder(BorderFactory.createLoweredBevelBorder());
//
// jPanel1
//
jPanel1.setLayout(new BorderLayout());
jPanel1.setBorder(new TitledBorder("Webcam Input"));
jPanel1.add(VideoPanel,BorderLayout.CENTER);
//
// jPanel2
//
jPanel2.setLayout(new BorderLayout());
jPanel2.setBorder(new TitledBorder("Captured Image"));
jPanel2.add(FrameCapturePanel,BorderLayout.CENTER);
//
// FrameCapture
//
FrameCapturePanel.setLayout(new BorderLayout());
//
//
// FrameProcess
//
FrameProcess.setLayout(new BorderLayout());
//
// jButton2
//
jButton2.setText("Start Camera");
jButton2.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton2_actionPerformed(e);
}

});
//
// jButton3
//
jButton3.setText("Capture");
```

```
jButton3.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton3_actionPerformed(e);
}

});
//
// jButton4
//
jButton4.setText("Binary");
jButton4.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton4_actionPerformed(e);
}

});
//
// jButton5
//
jButton5.setText("Gray Scale");
jButton5.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton5_actionPerformed(e);
}

});
//
// jButton6
//
jButton6.setText("Show Edges");
jButton6.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton6_actionPerformed(e);
}

});
//
 // jButton7
//
jButton7.setText("Negative");
jButton7.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton7_actionPerformed(e);
}

});
//
 // jButton8
//
jButton8.setText("Red Band");
jButton8.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton8_actionPerformed(e);
}

});
//
// jButton9
//
jButton9.setText("Green Band");
```

```
jButton9.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton9_actionPerformed(e);
}

});
//
// jButton9
//
jButton10.setText("Blue Band");
jButton10.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton10_actionPerformed(e);
}

});
jButton11.setText("Flip(Horizontal)");
jButton11.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton11_actionPerformed(e);
}

});

jButton12.setText("Flip(Vertical)");
jButton12.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton12_actionPerformed(e);
}

});

jButton13.setText("Sharpen");
jButton13.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton13_actionPerformed(e);
}

});

jButton14.setText("Blur");
jButton14.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton14_actionPerformed(e);
}

});

jButton15.setText("Binary Video");
jButton15.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton15_actionPerformed(e);
}

});

jButton16.setText("Average Band");
jButton16.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
```

```
{
jButton16_actionPerformed(e);
}

});
jButton17.setText("Sepia");
jButton17.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton17_actionPerformed(e);
}

});
jButton18.setText("Posterize");
jButton18.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton18_actionPerformed(e);
}

});
jButton19.setText("Stop Video");
jButton19.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton19_actionPerformed(e);
}

});
jButton20.setText("Gray-Scale Video");
jButton20.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton20_actionPerformed(e);
}

});
jButton21.setText("Red-Band Video");
jButton21.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton21_actionPerformed(e);
}

});
jButton22.setText("Green-Band Video");
jButton22.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton22_actionPerformed(e);
}

});
jButton23.setText("Blue-Band Video");
jButton23.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton23_actionPerformed(e);
}

});
jButton24.setText("Negative Video");
jButton24.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton24_actionPerformed(e);
}

});
jButton25.setText("Sepia Video");
jButton25.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton25_actionPerformed(e);
}

});
jButton28.setText("Background Image");
jButton28.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton28_actionPerformed(e);
}

});
jButton29.setText("Motion Detection");
jButton29.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e)
{
jButton29_actionPerformed(e);
}

});

ButtonGroup radiobuttongroup = new ButtonGroup();
radiobuttongroup.add(radiobutton1);
radiobuttongroup.add(radiobutton2);

//
// jPanel3
//
jPanel3.setLayout(new FlowLayout(FlowLayout.CENTER, 0, 0));
//
InputPanel.add(radiobutton2,0);
InputPanel.add(radiobutton1,1);
InputPanel.add(jButton2, 2);
InputPanel.add(jButton3, 3);

InputPanel0.add(jButton11, 0);
InputPanel0.add(jButton12, 1);
InputPanel0.add(jButton13,2);
InputPanel0.add(jButton14,3);
InputPanel0.add(jButton4, 4);
InputPanel0.add(jButton5, 5);
InputPanel0.add(jButton6, 6);
InputPanel0.add(jButton7, 7);
InputPanel0.add(jButton8,8);
InputPanel0.add(jButton9,9);
InputPanel0.add(jButton10,10);
InputPanel0.add(jButton16, 11);
InputPanel0.add(jButton17, 12);
InputPanel0.add(jButton18,13);

InputPanel1.add(jButton15, 0);
InputPanel1.add(jButton20, 1);
InputPanel1.add(jButton21, 2);
InputPanel1.add(jButton22, 3);
InputPanel1.add(jButton23, 4);
InputPanel1.add(jButton24, 5);
InputPanel1.add(jButton25, 6);
InputPanel1.add(jButton19, 7);
InputPanel1.add(jButton28, 8);
```

```
InputPanel1.add(jButton29, 9);

tab1.addTab("Input", null ,InputPanel,"Input Options");

tab1.addTab("Image Processing", null ,InputPanel0,"Image Processing Options");

tab1.addTab("Video Processing", null ,InputPanel1,"Video Processing Options");

jPanel3.add(tab1);

jPanel3.setBorder(new TitledBorder("Control Panel"));
//
// jPanel4
//
jPanel4.setLayout(new BorderLayout());
jPanel4.setBorder(new TitledBorder("Processed Image"));
jPanel4.add(FrameProcess,BorderLayout.CENTER);
//
// Video
//
this.setTitle("Image Analysis");
this.setLocation(new Point(30, 30));
this.setSize(new Dimension(800, 600));
this.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
this.setResizable(false);

}

private void jButton2_actionPerformed(ActionEvent e)
{

createPlayer();

}

private void jButton3_actionPerformed(ActionEvent e)
{

capturedImage = grabFrameImage();

if ( capturedImage != null ){

capturedphoto = new BufferedImage(FrameCapturePanel.getWidth(),
FrameCapturePanel.getHeight(),BufferedImage.TYPE_INT_RGB);

 Graphics G = capturedphoto.createGraphics();

 G.drawImage(capturedImage, 0, 0,FrameCapturePanel.getWidth(),FrameCapturePanel.getHeight(), null);

 G.dispose();

 Graphics2D g = (Graphics2D) FrameCapturePanel.getGraphics();

 g.drawImage(capturedphoto, 0, 0, null);

 g.dispose();

}

else
{
System.err.println ("Error : Could not grab frame" );
}

}
```

```
private void jButton4_actionPerformed(ActionEvent e)
{

if ( capturedImage != null ){

photo = new BufferedImage(FrameProcess.getWidth(),
FrameProcess.getHeight(),BufferedImage.TYPE_BYTE_BINARY);
 Draw1();

 Draw2();

}
else
{
System.err.println ("Error " );
}

}

private void jButton5_actionPerformed(ActionEvent e)
{
if ( capturedImage != null ){

photo = new BufferedImage(FrameProcess.getWidth(),
FrameProcess.getHeight(),BufferedImage.TYPE_BYTE_GRAY);

Draw1();

Draw2();

}
else
{
System.err.println ("Error " );
}

}

private void jButton6_actionPerformed(ActionEvent e)
{

if ( capturedImage != null ){

photo = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);
 //
photo1 = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);

Graphics G = photo1.createGraphics();

G.drawImage(capturedImage, 0, 0,FrameProcess.getWidth(),FrameProcess.getHeight() , null);
 G.dispose();

photo2 = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);
Graphics G1 = photo2.createGraphics();

G1.drawImage(capturedImage, 0, 0,FrameProcess.getWidth(),FrameProcess.getHeight() , null);
 G1.dispose();

float value[] = { 1.0f, 0.0f, -1.0f, 1.0f, 0.0f, -1.0f, 1.0f, 0.0f,-1.0f };
Kernel kernel = new Kernel(3, 3, value);
ConvolveOp convolve = new ConvolveOp(kernel, ConvolveOp.EDGE_NO_OP,null);
convolve.filter(photo1, photo2);
photo = photo2;

Draw2();
```

```
}
else
{
System.err.println ("Error " );
}

}

private void jButton7_actionPerformed(ActionEvent e)
{
if ( capturedImage != null ){

photo = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);

Draw1();

LookupTable lookup;
byte reverse[] = new byte[256];
for (int i = 0; i < 256; i++) {
reverse[i] = (byte) (255 - i);
}
lookup = new ByteLookupTable(0, reverse);
LookupOp lop = new LookupOp(lookup, null);
lop.filter(photo, photo);

Draw2();

}
else
{
System.err.println ("Error " );
}

}

private void jButton8_actionPerformed(ActionEvent e)
{
if ( capturedImage != null ){

float RED_MATRIX[][] = { { 1.0f, 0.0f, 0.0f },{ 0.0f, 0.0f, 0.0f }, { 0.0f, 0.0f, 0.0f } };

photo1 = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);

 Graphics G = photo1.createGraphics();

 G.drawImage(capturedImage, 0, 0,FrameProcess.getWidth(), FrameProcess.getHeight(), null);

 G.dispose();
raster = photo1.getRaster();

photo2 = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);
writableRaster = (WritableRaster) photo2.getRaster();

float combineMatrix[][];
combineMatrix=RED_MATRIX;
BandCombineOp band = new BandCombineOp(combineMatrix, null);
band.filter(raster, writableRaster);
photo = photo2;

 Draw2();

}
else
{
System.err.println ("Error " );
```

```
}

}

private void jButton9_actionPerformed(ActionEvent e)
{
if ( capturedImage != null ){

float GREEN_MATRIX[][] = {{ 0.0f, 0.0f, 0.0f }, { 0.0f, 1.0f, 0.0f }, { 0.0f, 0.0f, 0.0f }};

photo1 = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);

 Graphics G = photo1.createGraphics();
 G.drawImage(capturedImage, 0, 0,FrameProcess.getWidth(), FrameProcess.getHeight(), null);
 G.dispose();
raster = photo1.getRaster();
photo2 = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);
writableRaster = (WritableRaster) photo2.getRaster();

float combineMatrix[][];
combineMatrix=GREEN_MATRIX;
BandCombineOp band = new BandCombineOp(combineMatrix, null);
band.filter(raster, writableRaster);
photo = photo2;

 Draw2();

}
else
{
System.err.println ("Error " );
}

}

private void jButton10_actionPerformed(ActionEvent e)
{
if ( capturedImage != null ){

float BLUE_MATRIX[][] = { { 0.0f, 0.0f, 0.0f },{ 0.0f, 0.0f, 0.0f }, { 0.0f, 0.0f, 1.0f } };

photo1 = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);

Graphics G = photo1.createGraphics();
G.drawImage(capturedImage, 0, 0,FrameProcess.getWidth(), FrameProcess.getHeight(), null);
G.dispose();
raster = photo1.getRaster();
 photo2 = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);
writableRaster = (WritableRaster) photo2.getRaster();

float combineMatrix[][];
combineMatrix=BLUE_MATRIX;
BandCombineOp band = new BandCombineOp(combineMatrix, null);
band.filter(raster, writableRaster);
photo = photo2;

 Draw2();

}
else
 {
System.err.println ("Error " );
}

}
```

```
private void jButton11_actionPerformed(ActionEvent e)
{

if ( capturedImage != null ){


photo = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);

 Graphics G = photo.createGraphics();
 G.drawImage(capturedImage, 0, 0,FrameProcess.getWidth(), FrameProcess.getHeight(), null);
G.drawImage(photo, 0, 0, FrameProcess.getWidth(), FrameProcess.getHeight(), FrameProcess.getWidth(), 0, 0,
FrameProcess.getHeight(), null);
G.dispose();

 Draw2();

}
else
{
System.err.println ("Error " );
}

}

private void jButton12_actionPerformed(ActionEvent e)
{

if ( capturedImage != null ){


photo1 = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);

 photo = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);
Graphics G = photo1.createGraphics();
G.drawImage(capturedImage, 0, 0,FrameProcess.getWidth(), FrameProcess.getHeight(), null);
 G.dispose();

 Graphics G1 = photo.createGraphics();
G1.drawImage(photo1, 0, 0, FrameProcess.getWidth(), FrameProcess.getHeight(), 0, FrameProcess.getHeight(),
FrameProcess.getWidth(), 0, null);
G1.dispose();

Draw2();

}
else
{
System.err.println ("Error " );
}

}

private void jButton13_actionPerformed(ActionEvent e)
{

if ( capturedImage != null ){

photo = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);

//
photo1 = new BufferedImage(FrameProcess.getWidth(),
FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);
Graphics G = photo1.createGraphics();
G.drawImage(capturedImage, 0, 0,FrameProcess.getWidth(),FrameProcess.getHeight() , null);
G.dispose();
photo2 = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);
Graphics G1 = photo2.createGraphics();

G1.drawImage(capturedImage, 0, 0,FrameProcess.getWidth(),FrameProcess.getHeight() , null);

G1.dispose();

float value[] = { -1.0f, -1.0f, -1.0f, -1.0f, 9.0f, -1.0f, -1.0f, -1.0f,-1.0f };
Kernel kernel = new Kernel(3, 3, value);
ConvolveOp convolve = new ConvolveOp(kernel, ConvolveOp.EDGE_NO_OP,null);
convolve.filter(photo1, photo2);
photo = photo2;

Draw2();

}
else
{
System.err.println ("Error " );
}

}

private void jButton14_actionPerformed(ActionEvent e)
{

if ( capturedImage != null ){

photo = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);

//
photo1 = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);
Graphics G = photo1.createGraphics();
G.drawImage(capturedImage, 0, 0,FrameProcess.getWidth(),FrameProcess.getHeight() , null);
 G.dispose();

photo2 = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);
Graphics G1 = photo2.createGraphics();

G1.drawImage(capturedImage, 0, 0,FrameProcess.getWidth(),FrameProcess.getHeight() , null);
G1.dispose();

float value[] = { 0.0625f, 0.125f, 0.0625f, 0.125f, 0.25f, 0.125f, 0.0625f, 0.125f, 0.0625f };
Kernel kernel = new Kernel(3, 3, value);
ConvolveOp convolve = new ConvolveOp(kernel, ConvolveOp.EDGE_NO_OP,null);
convolve.filter(photo1, photo2);
photo = photo2;

Draw2();


}
else
{
System.err.println ("Error " );
}

}

private void jButton15_actionPerformed(ActionEvent e)
{

timerbinary = new javax.swing.Timer(200, BinaryVideo);
timerbinary.start();

}
```

```
private void jButton16_actionPerformed(ActionEvent e)
{

if ( capturedImage != null ){

float AVG_MATRIX[][] = { { 0.5f, 0.0f, 0.0f }, { 0.0f, 0.5f, 0.0f }, { 0.0f, 0.0f, 0.5f } };

photo1 = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);

 Graphics G = photo1.createGraphics();

 G.drawImage(capturedImage, 0, 0,FrameProcess.getWidth(), FrameProcess.getHeight(), null);

 G.dispose();
raster = photo1.getRaster();
 photo2 = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);
writableRaster = (WritableRaster) photo2.getRaster();

float combineMatrix[][];
combineMatrix=AVG_MATRIX;
BandCombineOp band = new BandCombineOp(combineMatrix, null);
band.filter(raster, writableRaster);
photo = photo2;

 Draw2();

}
else
{
System.err.println ("Error " );
}
}

private void jButton17_actionPerformed(ActionEvent e)
{

if ( capturedImage != null ){


photo = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);

Draw1();

int sepiaDepth = 20;
int w = photo.getWidth();
int h = photo.getHeight();
 writableRaster = photo.getRaster();

 int[] pixels = new int[w*h*3];
writableRaster.getPixels(0, 0, w, h, pixels);

 for (int i=0;i<pixels.length; i+=3)
{
int r = pixels[i];
int g = pixels[i+1];
int b = pixels[i+2];
int gry = (r + g + b) / 3;
r = g = b = gry;
r = r + (sepiaDepth * 2);
g = g + sepiaDepth;
if (r>255) r=255;
 if (g>255) g=255;
if (b>255) b=255;

if (b<0) b=0;
if (b>255) b=255;
 pixels[i] = r;
 pixels[i+1]= g;
pixels[i+2] = b;
}
 writableRaster.setPixels(0, 0, w, h, pixels);

Draw2();


}
 else
   {
 System.err.println ("Error " );
}


}

private void jButton18_actionPerformed(ActionEvent e)
{
if ( capturedImage != null ){



photo = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);

Draw1();

 LookupTable lookup;
short[] posterize = new short[256];
 for (int i = 0; i < 256; i++) {
posterize[i] = (short) (i - (i % 32));
}
 lookup = new ShortLookupTable(0, posterize);
LookupOp lop = new LookupOp(lookup, null);
lop.filter(photo, photo);

Draw2();

}
 else
 {
System.err.println ("Error " );
}

}

private void jButton19_actionPerformed(ActionEvent e)
{
if(timerbinary != null){
 timerbinary.stop();}
if(timergray != null){
timergray.stop();}
if(timerred != null){
timerred.stop();}
 if(timergreen != null){
timergreen.stop();}
f(timerblue != null){
timerblue.stop();}
 if(timernegative != null){
timernegative.stop();}
 if(timersepia != null){
timersepia.stop();}
if(timerdetection != null){
timerdetection.stop();}
```

```
}

private void jButton20_actionPerformed(ActionEvent e)
{

timergray = new javax.swing.Timer(200, GrayVideo);
 timergray.start();

}

Private void jButton21_actionPerformed(ActionEvent e)
{

timerred = new javax.swing.Timer(200, RedVideo);
timerred.start();

}

private void jButton22_actionPerformed(ActionEvent e)
{

timergreen = new javax.swing.Timer(200, GreenVideo);
timergreen.start();

}

private void jButton23_actionPerformed(ActionEvent e)
{

timerblue = new javax.swing.Timer(200, BlueVideo);
 timerblue.start();

}

private void jButton24_actionPerformed(ActionEvent e)
{

timernegative = new javax.swing.Timer(200, NegativeVideo);
timernegative.start();

}

private void jButton25_actionPerformed(ActionEvent e)
{

timersepia = new javax.swing.Timer(200, SepiaVideo);
timersepia.start();

}

private void jButton28_actionPerformed(ActionEvent e)
{

capturedImage = grabFrameImage();

if ( capturedImage != null ){



backgroundimage = new BufferedImage(FrameCapturePanel.getWidth(),
FrameCapturePanel.getHeight(),BufferedImage.TYPE_INT_RGB);

Graphics G = backgroundimage.createGraphics();
G.drawImage(capturedImage, 0, 0,FrameCapturePanel.getWidth(),FrameCapturePanel.getHeight(), null);
G.dispose();
```

```
Graphics2D g = (Graphics2D) FrameCapturePanel.getGraphics();
g.drawImage(backgroundimage, 0, 0, null);
 g.dispose();

}

else
 {
 System.err.println ("Error : Could not grab frame" );
 }

}

private void jButton29_actionPerformed(ActionEvent e)
{
timerdetection = new javax.swing.Timer(500, DetectionVideo);
timerdetection.start();

}

private void Draw1(){

Graphics G = photo.createGraphics();
G.drawImage(capturedImage, 0, 0,FrameProcess.getWidth(), FrameProcess.getHeight(), null);
G.dispose();
}

private void Draw2(){

Graphics2D g = (Graphics2D) FrameProcess.getGraphics();
 g.drawImage(photo, 0, 0, null);
g.dispose();
}

protected static BufferedImage imageToBufferedImage(Image img) {
BufferedImage bi = new BufferedImage(img.getWidth(null), img.getHeight(null), BufferedImage.TYPE_INT_RGB);
Graphics2D g2 = bi.createGraphics();
g2.drawImage(img, null, null);
return bi;
}

ActionListener BinaryVideo = new ActionListener() {
 public void actionPerformed(ActionEvent evt) {
capturedImage = grabFrameImage();
if ( capturedImage != null ){
photo = new BufferedImage(FrameProcess.getWidth(),
FrameProcess.getHeight(),BufferedImage.TYPE_BYTE_BINARY);
Graphics G = photo.createGraphics();
G.drawImage(capturedImage, 0, 0,FrameProcess.getWidth(),FrameProcess.getHeight(), null);
G.dispose();

Graphics2D g = (Graphics2D) FrameProcess.getGraphics();
g.drawImage(photo, 0, 0, null);
  g.dispose();

 }

 else
{
System.err.println ("Error : Could not grab frame" );
}
}
 };

ActionListener GrayVideo = new ActionListener() {
public void actionPerformed(ActionEvent evt) {
```

```
capturedImage = grabFrameImage();
if ( capturedImage != null ){

photo = new BufferedImage(FrameProcess.getWidth(),
FrameProcess.getHeight(),BufferedImage.TYPE_BYTE_GRAY);

Graphics G = photo.createGraphics();
G.drawImage(capturedImage, 0, 0,FrameProcess.getWidth(),FrameProcess.getHeight(), null);
G.dispose();

Graphics2D g = (Graphics2D) FrameProcess.getGraphics();
 g.drawImage(photo, 0, 0, null);
g.dispose();

}

 else
{
 System.err.println ("Error : Could not grab frame" );
}
 }
};

ActionListener RedVideo = new ActionListener() {
 public void actionPerformed(ActionEvent evt) {
 capturedImage = grabFrameImage();
if ( capturedImage != null ){

float RED_MATRIX[][] = { { 1.0f, 0.0f, 0.0f },{ 0.0f, 0.0f, 0.0f }, { 0.0f, 0.0f, 0.0f } };

photo1 = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);

Graphics G = photo1.createGraphics();
G.drawImage(capturedImage, 0, 0,FrameProcess.getWidth(), FrameProcess.getHeight(), null);
G.dispose();
    raster = photo1.getRaster();

photo2 = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);
writableRaster = (WritableRaster) photo2.getRaster();

 float combineMatrix[][];
combineMatrix=RED_MATRIX;
 BandCombineOp band = new BandCombineOp(combineMatrix, null);
band.filter(raster, writableRaster);
photo = photo2;

Draw2();

}

else

 System.err.println ("Error : Could not grab frame" );
 }
}
};

ActionListener GreenVideo = new ActionListener() {
public void actionPerformed(ActionEvent evt) {
apturedImage = grabFrameImage();
 if ( capturedImage != null ){

float GREEN_MATRIX[][] = { { 0.0f, 0.0f, 0.0f },{ 0.0f, 1.0f, 0.0f }, { 0.0f, 0.0f, 0.0f } };

photo1 = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);
```

```
Graphics G = photo1.createGraphics();
G.drawImage(capturedImage, 0, 0,FrameProcess.getWidth(), FrameProcess.getHeight(), null);
G.dispose();
 raster = photo1.getRaster();
photo2 = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);
writableRaster = (WritableRaster) photo2.getRaster();

float combineMatrix[][];
combineMatrix=GREEN_MATRIX;
 BandCombineOp band = new BandCombineOp(combineMatrix, null);
band.filter(raster, writableRaster);
photo = photo2;

Draw2();

 }

else
{
System.err.println ("Error : Could not grab frame" );
 }
}
};

ActionListener BlueVideo = new ActionListener() {
public void actionPerformed(ActionEvent evt) {
capturedImage = grabFrameImage();
if ( capturedImage != null ){

float BLUE_MATRIX[][] = { { 0.0f, 0.0f, 0.0f },{ 0.0f, 0.0f, 0.0f }, { 0.0f, 0.0f, 1.0f } };

photo1 = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);
Graphics G = photo1.createGraphics();
G.drawImage(capturedImage, 0, 0,FrameProcess.getWidth(), FrameProcess.getHeight(), null);
G.dispose();
raster = photo1.getRaster();
photo2 = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);
writableRaster = (WritableRaster) photo2.getRaster();

float combineMatrix[][];
combineMatrix=BLUE_MATRIX;
BandCombineOp band = new BandCombineOp(combineMatrix, null);
band.filter(raster, writableRaster);
photo = photo2;

Draw2();

 }

else
{
System.err.println ("Error : Could not grab frame" );
}
}
};

ActionListener NegativeVideo = new ActionListener() {
public void actionPerformed(ActionEvent evt) {
capturedImage = grabFrameImage();
if ( capturedImage != null ){


photo = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);
Draw1();

 LookupTable lookup;
```

```
byte reverse[] = new byte[256];
for (int i = 0; i < 256; i++) {
 reverse[i] = (byte) (255 - i);
}
lookup = new ByteLookupTable(0, reverse);
                                    LookupOp lop = new LookupOp(lookup, null);
 lop.filter(photo, photo);

Draw2();

}

      else
 {
 System.err.println ("Error : Could not grab frame" );
}
}
 };

ActionListener SepiaVideo = new ActionListener() {
public void actionPerformed(ActionEvent evt) {
 capturedImage = grabFrameImage();
if ( capturedImage != null ){

photo = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);
Draw1();

int sepiaDepth = 20;
 int w = photo.getWidth();
int h = photo.getHeight();
writableRaster = photo.getRaster();

int[] pixels = new int[w*h*3];
writableRaster.getPixels(0, 0, w, h, pixels);

for (int i=0;i<pixels.length; i+=3)
{
int r = pixels[i];
int g = pixels[i+1];
int b = pixels[i+2];
 int gry = (r + g + b) / 3;
r = g = b = gry;
r = r + (sepiaDepth * 2);
g = g + sepiaDepth;
if (r>255) r=255;
if (g>255) g=255;
 if (b>255) b=255;

if (b<0) b=0;
if (b>255) b=255;
pixels[i] = r;
pixels[i+1]= g;
pixels[i+2] = b;
}
writableRaster.setPixels(0, 0, w, h, pixels);

Draw2();

}

else
{
 System.err.println ("Error : Could not grab frame" );
}
}
};
```

```
ActionListener DetectionVideo = new ActionListener() {
public void actionPerformed(ActionEvent evt) {
capturedImage = grabFrameImage();

if ( capturedImage != null ){



photo = new BufferedImage(FrameProcess.getWidth(), FrameProcess.getHeight(),BufferedImage.TYPE_INT_RGB);


Draw1();
Draw2();

 ImageCompare ic = new ImageCompare(backgroundimage, photo);
 ic.setParameters(8, 6, 5, 10);
 ic.setDebugMode(0);
ic.compare();

Graphics2D g = (Graphics2D) FrameProcess.getGraphics();

g.drawImage(ic.getChangeIndicator(), 0, 0, null);

g.dispose();
}

 else
{
 System.err.println ("Error : Could not grab frame" );
}
}
};

public String[] getFormats() {
String[] Formats = ImageIO.getWriterFormatNames();
TreeSet<String> FormatsSet = new TreeSet<String>();
for (String s : Formats) {
FormatsSet.add(s.toLowerCase());
 }
return FormatsSet.toArray(new String[0]);
 }

private void setUrlAdd() {

rtpAdd = "rtp://224.123.111.101:22224/video/1";


}

private void setMediaLocator() {
//below the code to get video across the internet/broadcast
 radiobutton1.addActionListener(
 new ActionListener() {
public void actionPerformed(ActionEvent e)
{
 JRadioButton rbutton = (JRadioButton) e.getSource();
if (rbutton.equals(radiobutton1)) {
mlr = new MediaLocator("vfw://0");
}
                                   }

}
);

 radiobutton2.addActionListener(
 new ActionListener() {
public void actionPerformed(ActionEvent e)
{
JRadioButton rbutton = (JRadioButton) e.getSource();
if (rbutton.equals(radiobutton2)) {
mlr = new MediaLocator(rtpAdd);
 }
}

}
 );

}


private void createPlayer() {

if (mlr == null) {
 System.err.println("Can't build MRL for RTP:"+ rtpAdd);
System.exit(1);
}

try {

player = Manager.createPlayer(mlr);
player.addControllerListener(this);
player.start();


} catch (NoPlayerException e) {
System.err.println("Error:" + e);

} catch (MalformedURLException e) {
System.err.println("Error:" + e);

} catch (IOException e) {
System.err.println("Error:" + e);

}

}

public synchronized void controllerUpdate(ControllerEvent event) {
Component comp = null;

if (event instanceof RealizeCompleteEvent ){


comp = player.getVisualComponent();
if (comp != null){
jPanel1.add ( comp, BorderLayout.CENTER );
validate();

}
}

if (event instanceof StopEvent){

try{
playerClose();
}catch(Exception e){
JOptionPane.showMessageDialog(this,"Problem closing CAM:\nPlayer Exception: "+e);
}
}
```

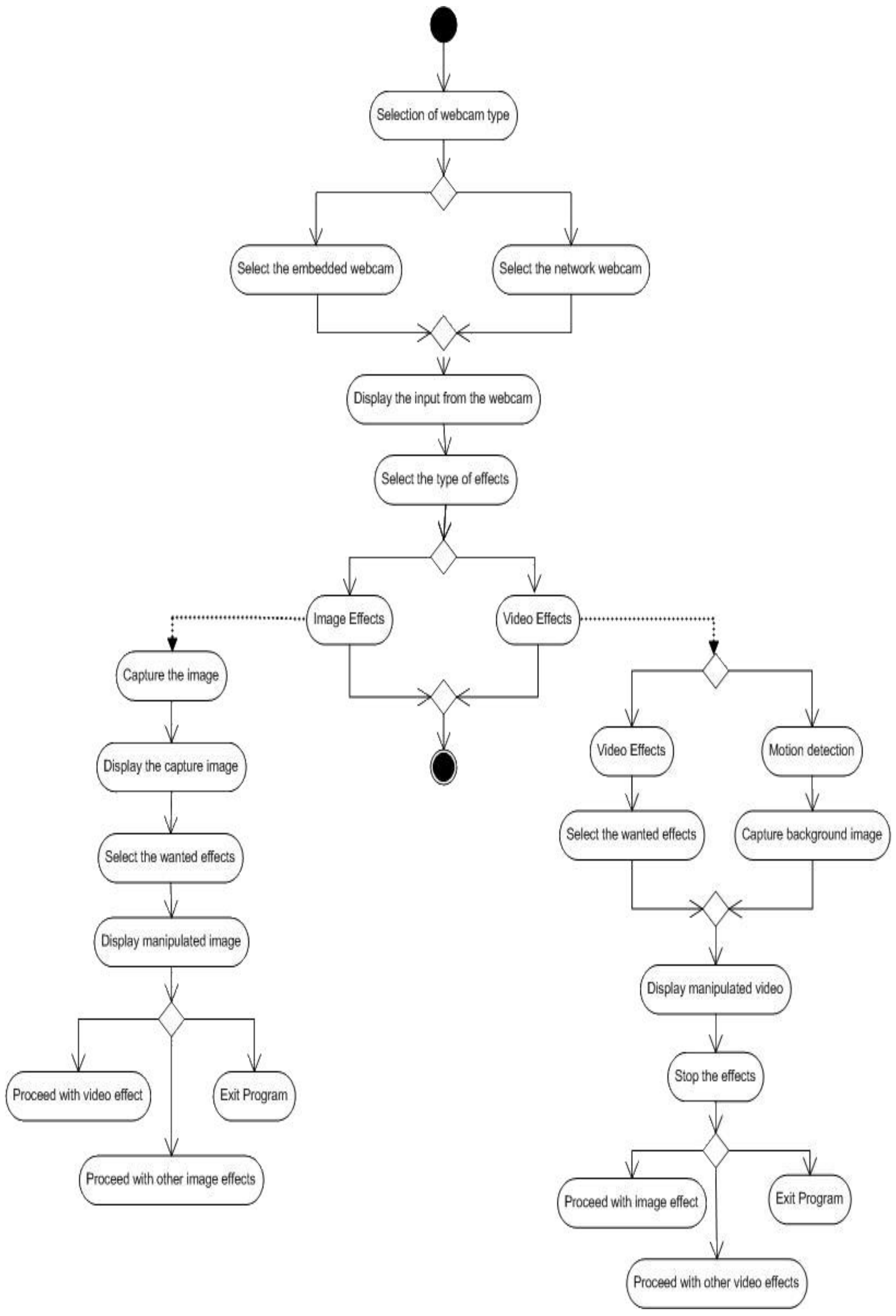
```
if (event instanceof ResourceUnavailableEvent) {

playerClose();
}
}

private void playerClose() {

if ( player != null ){

player.close();
player.deallocate();
player = null;

}
}

private Buffer grabFrameBuffer() {
if ( player != null )
 {
frameGrabbingControl = (FrameGrabbingControl)player.getControl
( "javax.media.control.FrameGrabbingControl" );

if ( frameGrabbingControl != null )
{
return ( frameGrabbingControl.grabFrame() );
}
else
{
System.err.println ("Error : FrameGrabbingControl is null");
return ( null );
}
}
else
{
System.err.println ("Error : Player is null");
return ( null );
}

}

private Image grabFrameImage() {

buffer = grabFrameBuffer();

if ( buffer != null )
{
 // Convert it to an image
bufferToImage = new BufferToImage ( (VideoFormat)buffer.getFormat() );
if ( bufferToImage != null )
{
bufferedToImage = bufferToImage.createImage ( buffer );
if ( bufferedToImage != null )
{
return (bufferedToImage);
}
else
{
System.err.println ("Error : BufferToImage cannot convert buffer");
return ( null );
}
}
else
{
System.err.println ("Error : cannot create BufferToImage instance");
return ( null );
```

```
                }
                }
                else
                {
                System.out.println ("Error : Buffer grabbed is null");
                return ( null );
                }
                }

//============================ Testing ===============================//
//=                                                   =//
//= The following main method is just for testing this class you built.=//
//= After testing,you may simply delete it.                    =//
//==================================================================//
                public static void main(String[] args)
                {
                JFrame.setDefaultLookAndFeelDecorated(true);
                JDialog.setDefaultLookAndFeelDecorated(true);
                try
                {
                UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
                }
                catch (Exception ex)
                {
                System.out.println("Failed loading L&F: ");
                System.out.println(ex);
                }
                new Video();
                }
                //= End of Testing =
                }
```

```
package suite;

import javax.swing.*;
import java.io.*;
import java.util.*;
import java.awt.*;
import java.awt.image.*;
import com.sun.image.codec.jpeg.*;

 public class ImageCompare {

protected BufferedImage img1 = null;
protected BufferedImage img2 = null;
protected BufferedImage imgc = null;
protected int comparex = 0;
protected int comparey = 0;
protected int factorA = 0;
protected int factorD = 10;
protected boolean match = false;
protected int debugMode = 0;
 // 1: textual indication of change, 2: difference of factors

public ImageCompare(BufferedImage img1, BufferedImage img2) {
this.img1 = img1;
this.img2 = img2;
autoSetParameters();
}

protected void autoSetParameters() {
comparex = 10;
comparey = 10;
factorA = 10;
factorD = 10;
}

public void setParameters(int x, int y, int factorA, int factorD) {
this.comparex = x;
this.comparey = y;
this.factorA = factorA;
this.factorD = factorD;
}

public void setDebugMode(int m) {
this.debugMode = m;
}

// compare the two images in this object.
public void compare() {
// setup change display image
imgc = imageToBufferedImage(img2);
Graphics2D gc = imgc.createGraphics();
gc.setColor(Color.RED);
// convert to gray images.
img1 = imageToBufferedImage(GrayFilter.createDisabledImage(img1));
img2 = imageToBufferedImage(GrayFilter.createDisabledImage(img2));
// how big are each section
int blocksx = (int)(img1.getWidth() / comparex);
int blocksy = (int)(img1.getHeight() / comparey);
// set to a match by default, if a change is found then flag non-match
this.match = true;
// loop through whole image and compare individual blocks of images
for (int y = 0; y < comparey; y++) {
if (debugMode > 0) System.out.print("|");
for (int x = 0; x < comparex; x++) {
int b1 = getAverageBrightness(img1.getSubimage(x*blocksx, y*blocksy, blocksx - 1, blocksy - 1));
int b2 = getAverageBrightness(img2.getSubimage(x*blocksx, y*blocksy, blocksx - 1, blocksy - 1));
int diff = Math.abs(b1 - b2);
if (diff > factorA) { // the difference in a certain region has passed the threshold value of factorA
// draw an indicator on the change image to show where change was detected.
gc.drawRect(x*blocksx, y*blocksy, blocksx - 1, blocksy - 1);
this.match = false;
}
if (debugMode == 1) System.out.print((diff > factorA ? "X" : " "));
if (debugMode == 2) System.out.print(diff + (x < comparex - 1 ? "," : ""));
}
if (debugMode > 0) System.out.println("|");
}
}

// return the image that indicates the regions where changes where detected.
public BufferedImage getChangeIndicator() {
return imgc;
}

// returns a value specifying some kind of average brightness in the image.
protected int getAverageBrightness(BufferedImage img) {
Raster r = img.getData();
int total = 0;
for (int y = 0; y < r.getHeight(); y++) {
for (int x = 0; x < r.getWidth(); x++) {
total += r.getSample(r.getMinX() + x, r.getMinY() + y, 0);
}
}
return (int)(total / ((r.getWidth()/factorD)*(r.getHeight()/factorD)));
}

// returns true if image pair is considered a match
public boolean match() {
return this.match;
}

// buffered images are just better.
protected static BufferedImage imageToBufferedImage(Image img) {
BufferedImage bi = new BufferedImage(img.getWidth(null), img.getHeight(null), BufferedImage.TYPE_INT_RGB);
Graphics2D g2 = bi.createGraphics();
g2.drawImage(img, null, null);
return bi;
}

}
```

APPENDIX 2

ACTIVITY DIAGRAM OF THE SUITE


Selection of webcam type
Select the embedded webcam
Select the network webcam
Display the input from the webcam
Select the type of effects
Image Effects
Video Effects
Capture the image
Display the capture image
Select the wanted effects
Display manipulated image
Proceed with video effect
Exit Program
Proceed with other image effects
Video Effects
Motion detection
Select the wanted effects
Capture background image
Display manipulated video
Stop the effects
Proceed with image effect
Exit Program
Proceed with other video effects