Letter of Transmittal

4431 Union Street
Burnaby, British Columbia V5C 2X7

March 19, 2011

Randall Kerr
English Coordinator
Department of Applied Science
University of British Columbia
Vancouver, British Columbia

Dear Mr. Kerr,

Please find attached a copy of my final report entitled “EECE 496 Hand Gesture Recognition Software Project”.

The main purpose for this report is to document the processes, my findings, and implementation results in developing hand gesture recognition software.

I hope that this report will merit your approval

Respectfully yours,

Derick Hsieh

Derick Hsieh

Hand Gesture Recognition Software

EECE 496 Final Report

Supervisor: Dr. Philippe Kruchten

Derick Hsieh <11978079>

March 28, 2011

ABSTRACT

Gesture control takes advantage of the most natural form of interaction between human and hardware systems. With the rapid advance of technology in the gaming industry including Nintendo Wii, PlayStation Move, and Microsoft Kinect, user and hardware interaction has become a mainstream requirement to system usability. This project will take a step forward to investigate object detection methods and develop a program that bridges the usability of daily personal computers with its software system.

Table of Contents

LIST OF ILLUSTRATIONS

LIST OF TABLES

GLOSSARY

Blobs –	Cluster of object of interests
Threads –	The smallest processing unit that can concurrently run two or more different tasks
Synchronization –	The coordination of events that operate or share the same resources
Global –	A type of variable declaration that is accessible in every scope

LIST OF ABBREVIATIONS

HGR	– Hand Gesture Recognition
RGB	– Red, Green, and Blue
HSB	– Hue, Saturation, and Brightness
OpenCV	– Open Source Computer Vision
MIPF	– Marvin Image Processing Framework
FR	– Functional Requirements
NFR	– Non-Functional Requirements
ROI	– Region of Interest
CPU	– Central Processing Unit
GUI	– Graphical User Interface
API	– Application Programming Interface
JRE	– Java Runtime Environment

1. INTRODUCTION

This report presents investigations and development of hand gesture recognition software. This software will allow in-depth analysis of hand gesture detection systems using consumer integrated webcams found in portable personal computers.

My objectives for this project are to evaluate the different object detection techniques and filtering algorithms to perform hand gesture detections. Upon successful detection, I should develop software solutions that will take the detection results and convert them to interactive features. Some of these usable features include gesture recognition of previous and next page, and normal mouse cursor movement control.

As technology advances from keyboards and mouse to touch inputs, usability of personal computers becomes an exciting and inevitable area of research and future trend. Many leading gaming industry such as Nintendo Wii, PlayStation Move, and Microsoft Kinect are using new ways to improve user interaction with their software gaming systems. These systems allow users to interact freely during games; however, this project aims to take usability to the next level where hand gestures can be used for everyday operations on personal computers. In addition, this project unlike previously mentioned gaming systems will utilize inexpensive consumer grade webcams for hand detection.

Hand gesture recognition software's purpose is to perform hand detection from webcams. The hand detection will involve several image processing modules such as the skin colour detection method. Based on user's choice of operation modes and dominant hand settings, the users should be able to navigate through different features to fully interact with Internet browsers or software applications.

Due to the use of low quality consumer webcams, there are few constraints to operating this software. The user is restricted to running the program in environments not under direct sunlight and bright backgrounds. This report will not go into details in the mathematical calculations behind object detection methods. However, each of the techniques used to detect hands will be discussed in depth. In addition, each components of the software recognition and action control system will be explained with further discussions.

This report divides into the following primary sections: research and design algorithms, software and hardware requirements of HGR software, design and experiment, software versions and application testing, results and conclusion.

2. RESEARCH AND DESIGN ALGORITHMS

In this section, the research in each of the different algorithm designs of the hand gesture recognition (HGR) system is explained in detail.

2.1. Red, Green, and Blue (RGB) Colour Space Detection

The use of RGB colour detection involves pixel-by-pixel comparison between each color value in the RGB colour space [1]. In the RGB color space, the red, green, and blue value of a colour pixel is compared with a threshold value as the basis of filtering parameter. This was the main method in detecting hand indicators for HGR Version 1.0. However, in the second release of this software, the method was replaced with the HSB colour detection to increase the efficiency under different environment lighting conditions.

2.2. Hue, Saturation, and Brightness (HSB) Colour Space Detection

The use of HSB allows a different kind of colour comparison than RGB. Shown on Figure 1 below, it takes into account of the hue, saturation, and brightness levels in the environment [2]. Although this algorithm also uses the threshold technique to filter unwanted parts of the image, it allows more room for error in environment light changes.

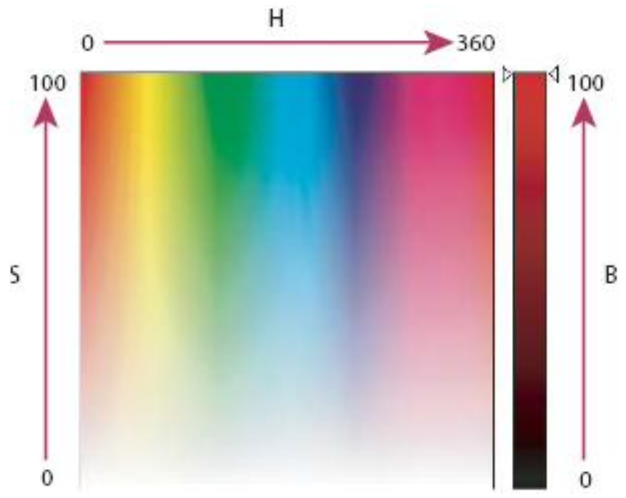


Figure 1 Hue, Saturation, and Brightness Colour Model

2.3. <u>Skin Colour Detection</u>

The skin colour detection method uses a combination of algorithms to detect and filter out non-skin objects. The skin colour detection mechanism used in this project is provided by the Marvin Image Processing Framework (MIPF) [3]. It uses the combination of RGB, HSB and histograms to find skin coloured pixels. In addition, it also performs vector calculations to perform edge detection to allow a better filter of skin coloured features such as faces and hands. An example of skin colour detection is shown in the figure below.

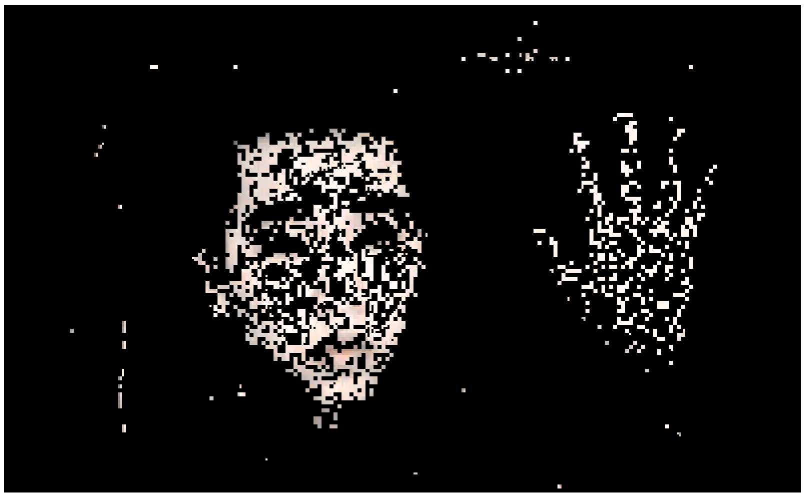


Figure 2 Example of skin colour detection

2.4. <u>Haar Training</u>

In Open Source Computer Vision, Haar Training allows user to train a system to detect objects through training on thousands of positive and negative samples [4]. The positive samples consist of pictures that include the object of interest and the negative samples consist of background images without the object of interest [4]. The training involves reshaping the object of interest with different backgrounds to allow the system to easily detect the object when paired with other OpenCV (discussed in section 4.1.1.) functions.

2.5. Background Subtraction

Background subtraction is a method in image processing where one can filtered out objects from stationary objects. The algorithm works as follow: the system takes two pictures at two different times and compares both of the images [5]. Then, the system will replace any unchanged pixel colour values with the black colour. The result of the comparison and filter will show objects that have changed or moved; this is shown on Figure 3 below. Therefore, in the hand gesture recognition software, this is an important component in motion detections.

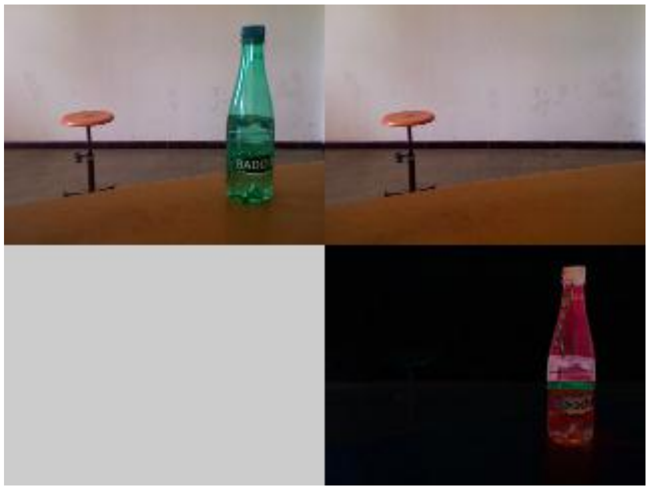


Figure 3 Example of background subtraction [3]

2.6. Histogram

In image processing, calculating the histogram of an image will show the distribution of colours inside an image. The histogram can be built for both RGB and HSB colour space. By calculating the histogram inside a picture, it will show the number of pixels that have colours in the fixed colour range that spans the image’s specific colour space [6]. This method is particularly useful such that a cluster of related colour, in this case, the skin colour, can be calculated inside the region of interest (ROI) and be used as the colour value for the threshold filter.

3. SOFTWARE AND HARDWARE REQUIREMENTS

This section will document the specific functional and non-functional software requirements of the hand gesture recognition software as required by Dr. Philippe Kruchten. These requirement statements will follow the format as in the "Software Requirements Engineering" process. The hardware components required to operate this software will be specified as well.

3.1. Functional Requirements (FR)

Functional requirements specify the main technical functionalities and specifications that the system should incorporate.

3.1.1. FR-01 Face Detection

This software shall utilize a face detection system to filter out faces from the video capturing device. By applying face detection, the system can disregard the region where the face is located and thus reducing the amount of calculation needed to perform hand detection. The face detection unit will be implemented with the help of OpenCV (discussed in section 4.1.1.).

3.1.2. FR-02 Skin Detection Module

This software shall perform skin colour detection and filter out all objects that do not contain the colour of skin. By filtering object of non-skin coloured, the system can then use its remaining resources and focus on hand detection and gesture recognitions. This also allows the system to pinpoint possible locations of user's hands. The skin detection module can be achieved by using the Marvin Image Processing Framework (discussed in section 4.1.2).

3.1.3. FR-03 Filtered Object Detection

Once the program has filtered out most of the unwanted parts of the picture after using the skin detection module, the software shall read and recognize “clusters” skin coloured objects also known as “**blobs**”.

3.1.4. FR-04 Object Location

Upon detection, the system shall be able to compute the location of the object using simple trigonometry math.

3.1.5. FR-05 Hand Calibration

Depending on user’s preferences, the system shall perform adjustments according to user’s dominant hand. This means that if the user is right handed, the mouse control gesture mode should be recognized near the right side of the face instead of the whole field of view. This is achieved through trigonometry math conversions.

3.1.6. FR-06 Mouse Movement Gesture Control Mode

After obtaining the location of the hand, the software shall use the detected location as the mouse cursor point. As the user moves his/her hands, the mouse should follow promptly on the screen.

3.1.7. FR-07 Browsing Gesture Control Mode

The software shall allow the user to use the “Browsing Gesture Mode”. In this mode, user’s hand gesture will only be recognized for commands including previous page, next page, scroll up and scroll down.

3.2. <u>Non-Functional Requirements (NFR)</u>

Non-functional requirements specify the criteria in the operation and the architecture of the system.

3.2.1. NFR-01 Efficiency in Computation

This software shall minimize the use of Central Processing Unit (CPU) and memory resources on the operating system. When HGR is executing, the software shall utilize less than 80% of the system's CPU resource and less than 100 megabytes of system memory.

3.2.2. NFR-02 Extensibility

The software shall be extensible to support future developments and add-ons to the HGR software. The gesture control module of HGR shall be at least 50% extensible to allow new gesture recognition features to be added to the system.

3.2.3. NFR-03 Portability

The HGR software shall be 100% portable to all operating platforms that support Java Runtime Environment (JRE). Therefore, this software should not depend on the different operating systems.

3.2.4. NFR-04 Performance

This software shall minimize the number of calculations needed to perform image processing and hand gesture detection. Each captured video frame shall be processed within 350 milliseconds to achieve 3 frames per second performance.

3.2.5. NFR-05 Reliability

The HGR software shall be operable in all lighting conditions. Regardless of the brightness level in user’s operating environment, the program shall always detect user’s hands.

3.2.6. NFR-06 Usability

This software shall be easy to use for all users with minimal instructions. 100% of the languages on the graphical user interface (GUI) shall be intuitive and understandable by non-technical users.

3.3. <u>Hardware Requirements</u>

HGR software does not require special equipment except for a personal computer (PC) and a webcam. The CPU of this computer should have at least two cores to handle the enormous amount of calculations needed for the image processing unit.

4. DESIGNS AND EXPERIMENTS

In this section, the dependencies from open source image processing libraries, software logic cycle, software constraints, and design components of the HGR software are discussed in detailed.

4.1. Special Application Program Interface (API) Used

This sub-section describes the libraries and open source programming API that the hand gesture recognition software relies on.

4.1.1. OpenCV – Open Source Computer Vision

Open Source Computer Vision, also known as OpenCV, is a real time computer vision library with many image processing functions developed by Intel for the C++ or Java programming platform [7]. This API provides many functions to perform calculation on captured video sources and special image processing on each frame of the video to support the HGR software. The video capture and the face detection components of this project are heavily supported by some of the functions built into the OpenCV library.

4.1.2. Marvin Image Processing Framework (MIPF)

The Marvin Image Processing Framework (MIPF) is also an open source image processing framework developed in Java. The initial version of MIPF is developed by Danilo Munoz, Fabio Andrijauskas, and Gabriel Archanjo [3]. Similar to OpenCV, the MIPF provides many image manipulation functions to images and captured videos frames. One of the built in plugin features is the skin colour detection function. This is used in the HGR skin detector component to filter user's hands.

4.2. <u>HGR Software Logic Cycle</u>

Figure 4 illustrated on next page shows the flow chart of how the program operates. The following are the steps from start to end of the HGR software:

1. Initialize webcam and capture video sources.
2. The system performs face detection.
3. Any face in the image is filtered and the frame is blurred.
4. The frame is then passed into the skin colour detection module to filter out backgrounds.
5. With only the image of hands remain in the image, the system will locate the hand and perform geometry translations.
6. Perform action:
    a. Browsing mode: detect gestures and perform the following actions:
        i. Previous page
        ii. Next page
        iii. Scroll up
        iv. Scroll down
    b. Mouse control mode: detect hand movements and translate the coordinates to user’s screen.
7. Repeat whole cycle until system pause or system end.

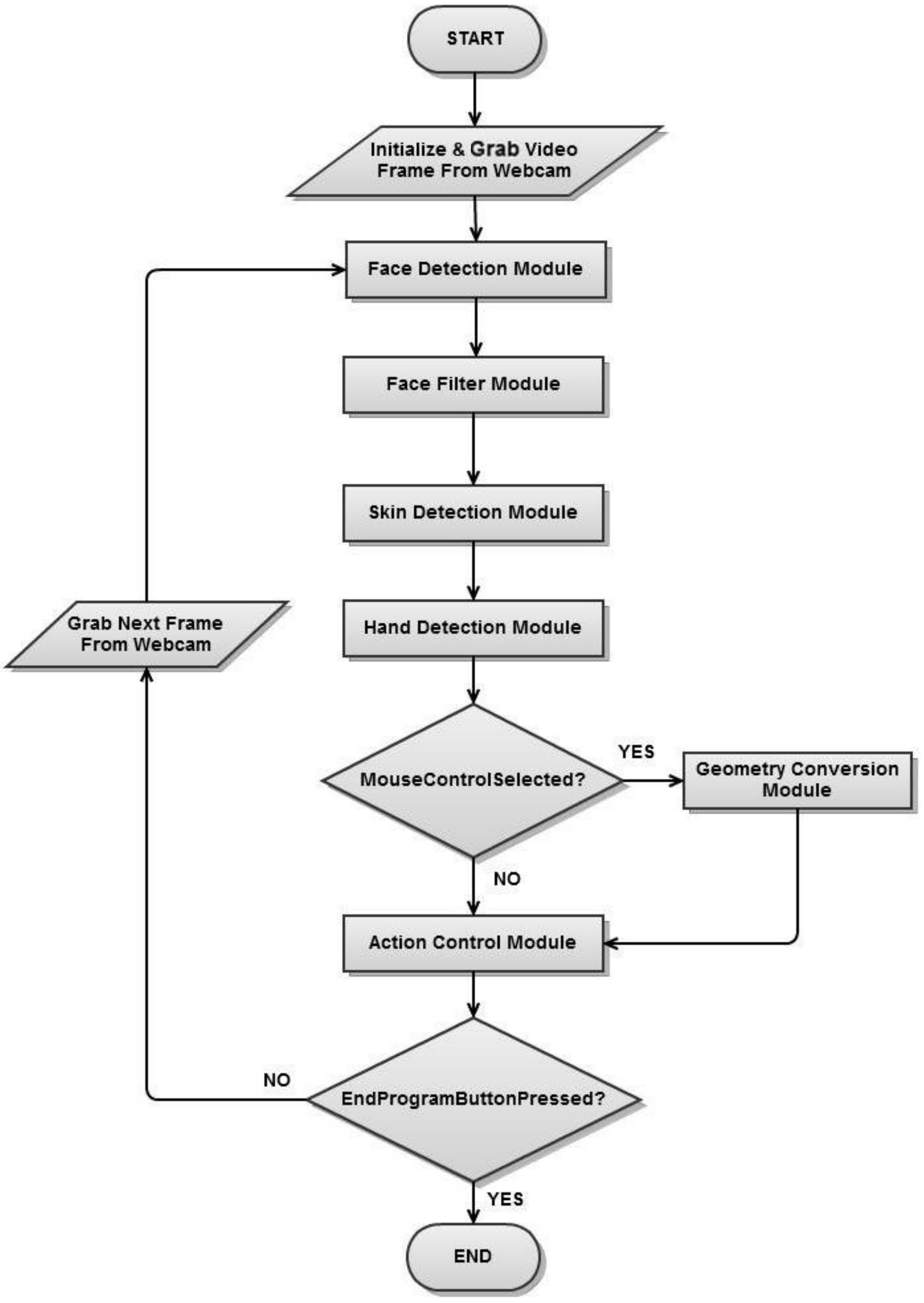


Figure 4 HGR Software Operation Flow Chart

4.3. <u>Software Constraints</u>

Although the software utilizes different methods of filtering and object detections, the system still varies under different environmental light. This system cannot guarantee to perform correct hand detection under very bright light source such as direct sunlight or having bright lights in the background in the case of windows. These factors affect the algorithm due to the fact that the light softens the skin colour needed for detection, thus rendering the skin detector ineffective to detect accurately.

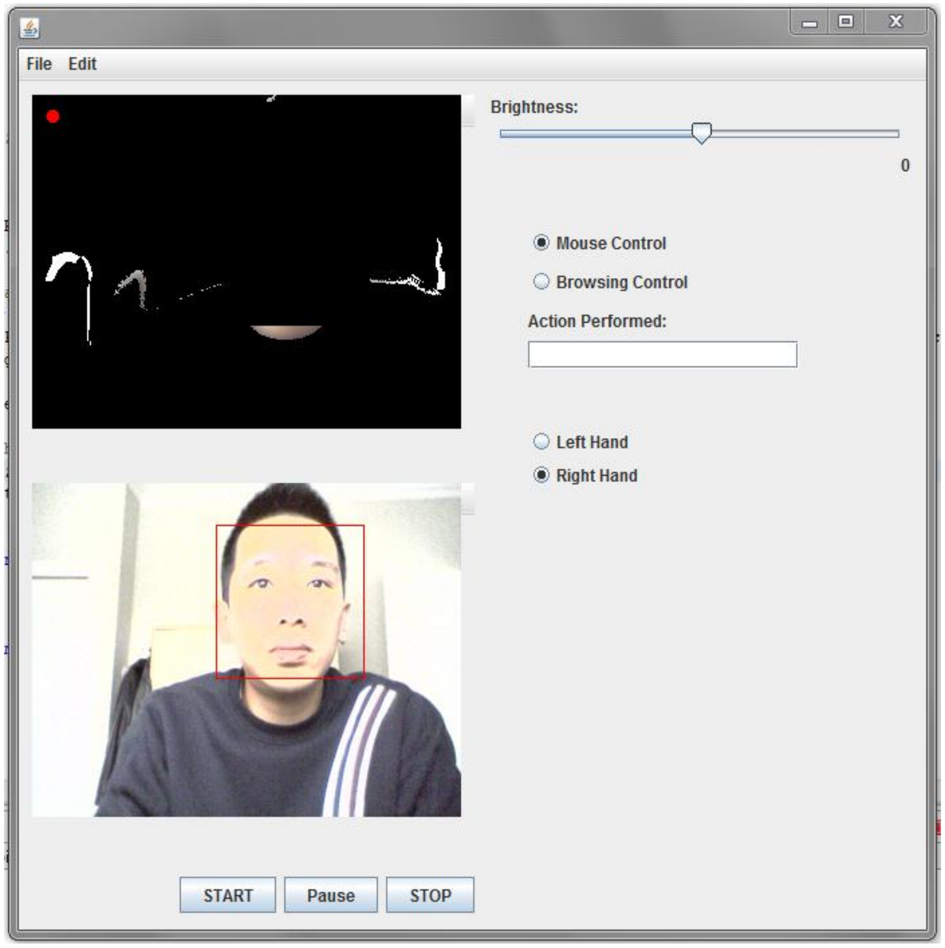
4.4. <u>Software Components</u>

This sub-section describes the features and functionalities of the different components of the HGR software.

4.4.1. Graphical User Interface (GUI)

HGR's final release version (version 3.0), as presented on Figure 5 on the next page, has three buttons for normal operations, a slider for brightness adjustments, radio buttons for different modes, and a text box to show the action performed. The difference between the two video frames is that the top video frame shows the hand detection whereas the bottom video frame shows the face detection in real-time. In addition, depending on the mode chosen by the user, if the Mouse Control is selected, then the action performed textbox would display the location coordinates of the mouse cursor. If the Browsing Control is selected, then the actions such as "Previous Page", "Next Page", "Scroll Up", and "Scroll Down" will be displayed based on the gestures recognized.

Figure 5 Graphical User Interface of HGR Release Version 3.0



4.4.2. Face Detection Module

The face detection module is developed using parts of the OpenCV library. The facial recognition is done by loading the ".xml" descriptor file that is generated by the Haar Training [4]. This file has been previously included by the OpenCV API community and that it contains details which allow the OpenCV function to perform face detection efficiently in short amount of time.

4.4.3. Face Filter Module

The face filter module takes the coordinates of the detected faces from the face detection module and applies the filter for that specific frame. Not only does this black out the face area but it also allow the skin detection module to operate without searching the area with the user’s face.

4.4.4. Skin Detection Module

Skin detection module is implemented by using the help from MIPF. This module, as the name suggests, filters out all objects that do not include the skin colour. The algorithm in filtering out the unwanted object includes using RGB, HSB, and histogram method to calibrate the right colour threshold [8]. The disadvantage of using this method is that under extreme lighting changes, the system cannot detect the skin colour accurately.

4.4.5. Hand Detection Module

The hand detection module scans through the image processed frames and distinguishes the different blob (objects) detected. This method is called image segmentation [9]. Once each blob is assigned a number, the software can then decide which blob is likely to be more similar to a hand based on its size. Figure 6 on the next page shows an example of blob distinction.

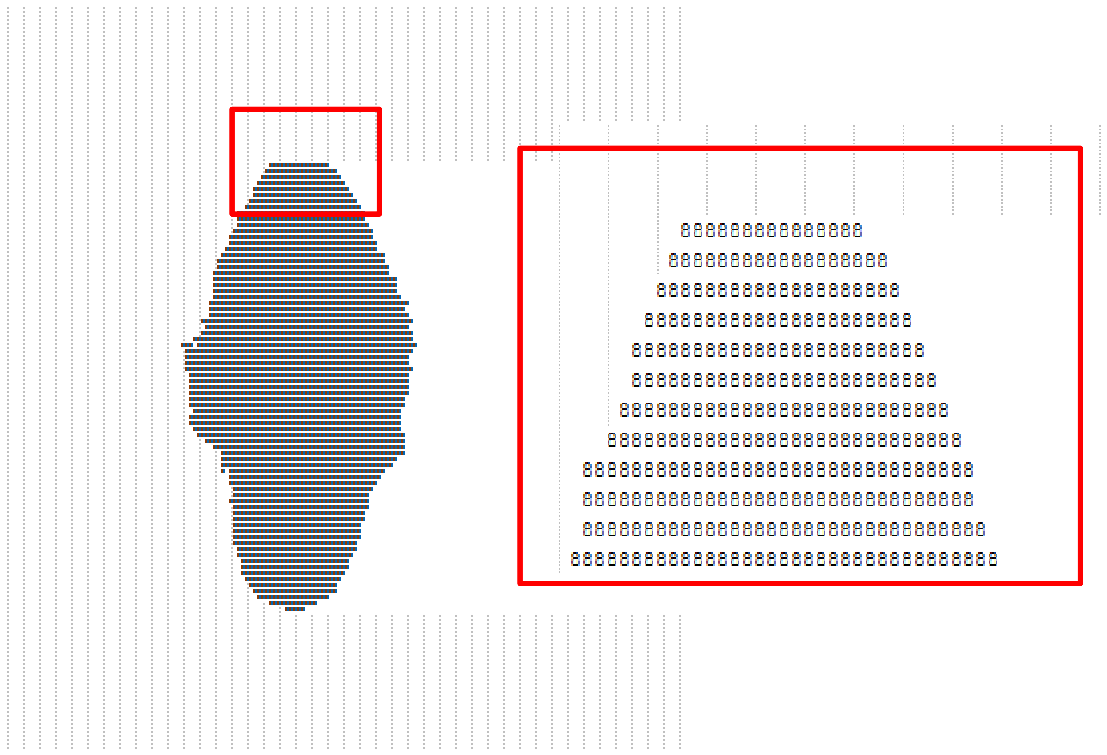


Figure 6 The enlarged section shows the number 8 assigned to this particular "blob". Other blobs have been filtered out due to their sizes not meeting the system bounds.

4.4.6. Geometry Conversion Module

The geometry conversion module’s only task is to convert the coordinates of the detected object and translate them proportionally to the PC screen. This module has two different calculation schemes: trigonometry calculations for the left and the right hand. Based on the left or right hand selection on the GUI, the software will identify the appropriate choice for coordination conversion in the mouse control mode. A high level view of this module is displayed by Figure 7 on the next page.

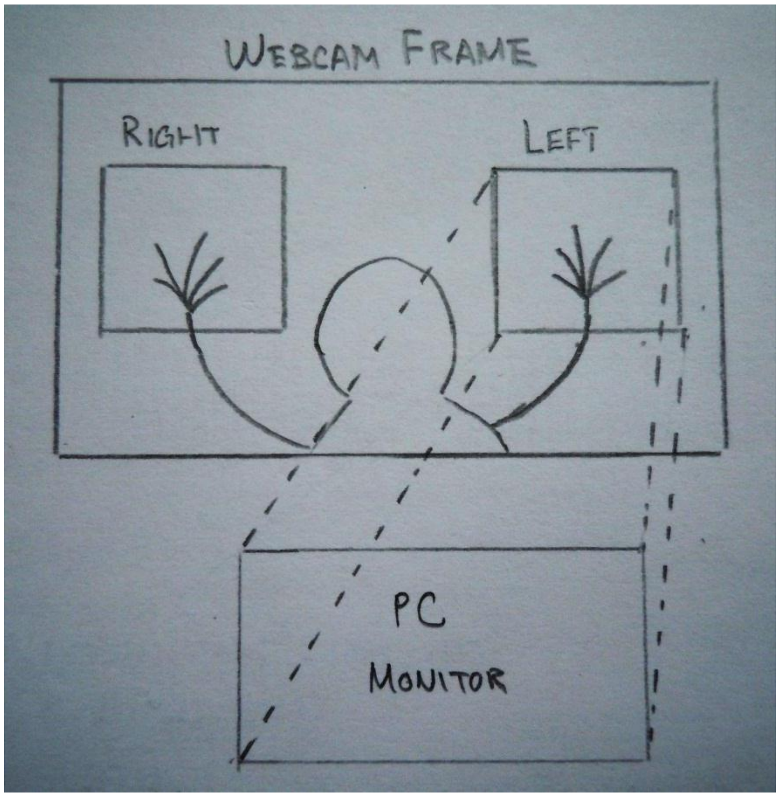


Figure 7 This shows the concept of how the Geometry Module translates the area of interest to the PC monitor.

4.4.7. Action Control Module

This module depends on the selection made by the user. If the Mouse Control mode is selected, the software will utilize the geometry conversion module to translate hand coordinates to the correct position on the screen. This way the user can control mouse cursor movements. With two hands, the system will simulate the left mouse click. In the Browsing Control mode, the software will calculate the amount of left, right, up, or down distance the hand has traveled. Based on that direction and distance, the software will issue commands to the corresponding mappings shown in Table 1 below.

Table 1 Browsing mode's gesture and action mapping

Movement	Action
Left	Previous Page
Right	Next Page
Up	Page Up
Down	Page Down

4.4.8. Class Diagram

The figure below shows the class diagram of the system. The class diagram is a detailed view that encompasses all the software modules previously mentioned. A larger version of the class diagram is available in the appendix.

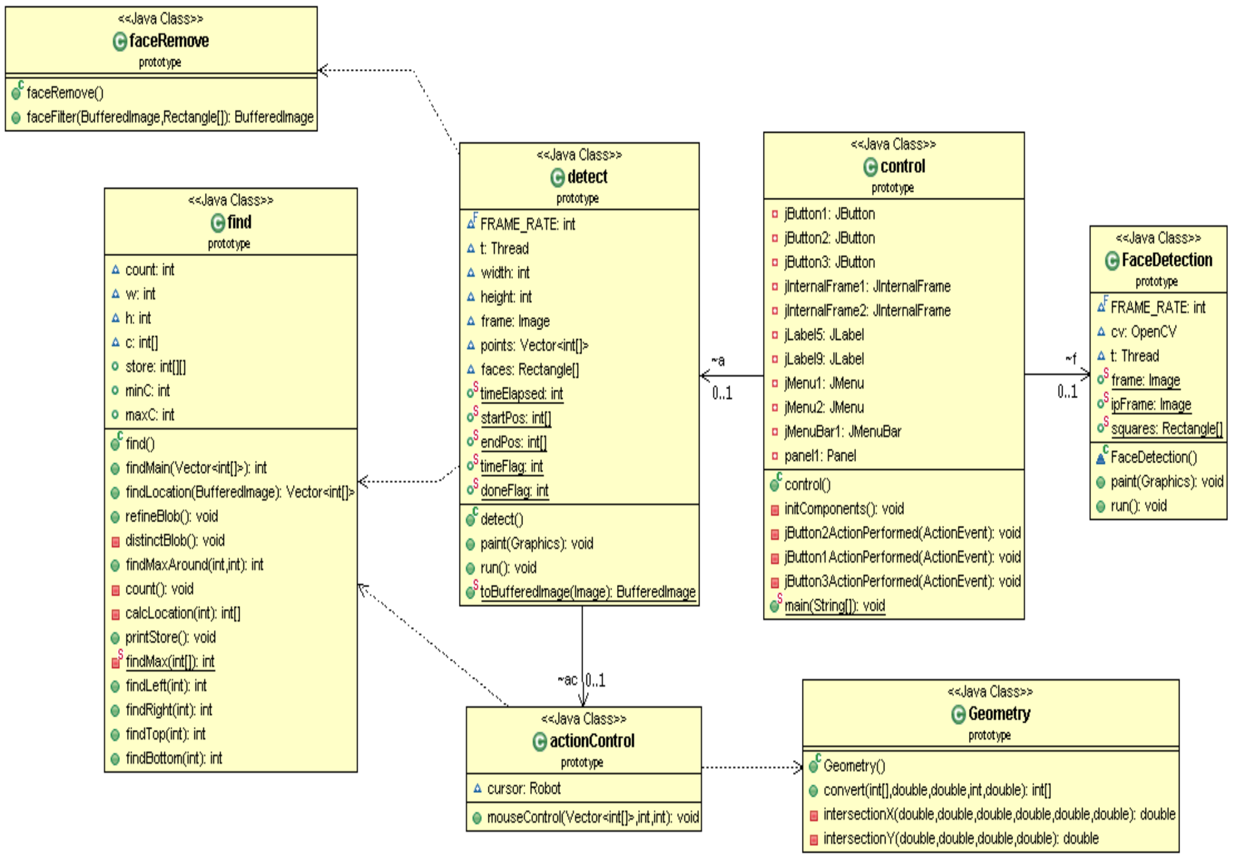


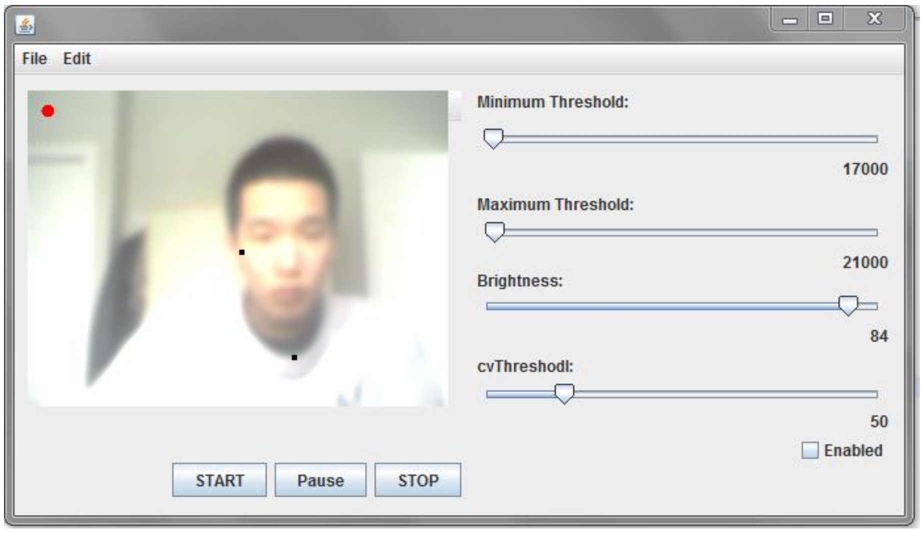
Figure 8 Class Diagram for HGR Software

5. SOFTWARE VERSIONS AND APPLICATION TESTING

In this section, the three versions of the software and the application testing are described in detail for further understanding of the software development process.

5.1. HGR Release Version 1.0

HGR Release Version 1.0 is the first prototype of the project. This version relied on red indicators on fingers. As introduced in section 2.1, the RGB algorithm allowed the system to find the red indicators on user's fingers by varying the colour threshold and comparing pixel by pixel from the captured video frame. Setting the RGB minimum and maximum and brightness values are the prerequisite to configure this detection system. Once the mouse control mode is started, the user can use his/her finger to control the mouse movement by waving the finger with the red indicator in front of the webcam. To perform mouse clicks, the user needs to raise the second finger with the red indicator. This method of detection performed very well; however, it relies on the fact that the user needs to wear indicators on one's fingers for the detection to work. Since the indicator colour is arbitrary chosen, if it coincides with other background objects of the same colour, the detection system will be affected and thus fail to recognize the controller finger. Figure 9 below shows the graphical user interface of HGR Release Version 1.0.



Minimum Threshold: 17000

Maximum Threshold: 21000

Brightness: 84

cvThreshodl: 50

Enabled

Figure 9 HGR Release Version 1.0 User Interface

5.2. <u>HGR Release Version 2.0</u>

HGR Release Version 2.0 has a new detection algorithm in place. This version allows detection of user's hands without the use of indicators. The system applies both RGB and HSB algorithm to filter out unwanted objects in the background. It requires initial user calibration for the hand colour and then compares the colour threshold through both RGB and HSB comparisons. After the comparison, the system will distinct the number of the hands in the field of view based on the size of the object. If the object size is within a lower and upper bound value for size, it will be considered as a hand. Furthermore, a mathematical conversion module is added to this release. This module allows the user to perform hand gestures in a smaller area than the previous version. Although the dependency on indicators has been removed, the hand detection is still very dependent of environment light. Figure 10 below shows hand detection in a room at night with desk lamps on.

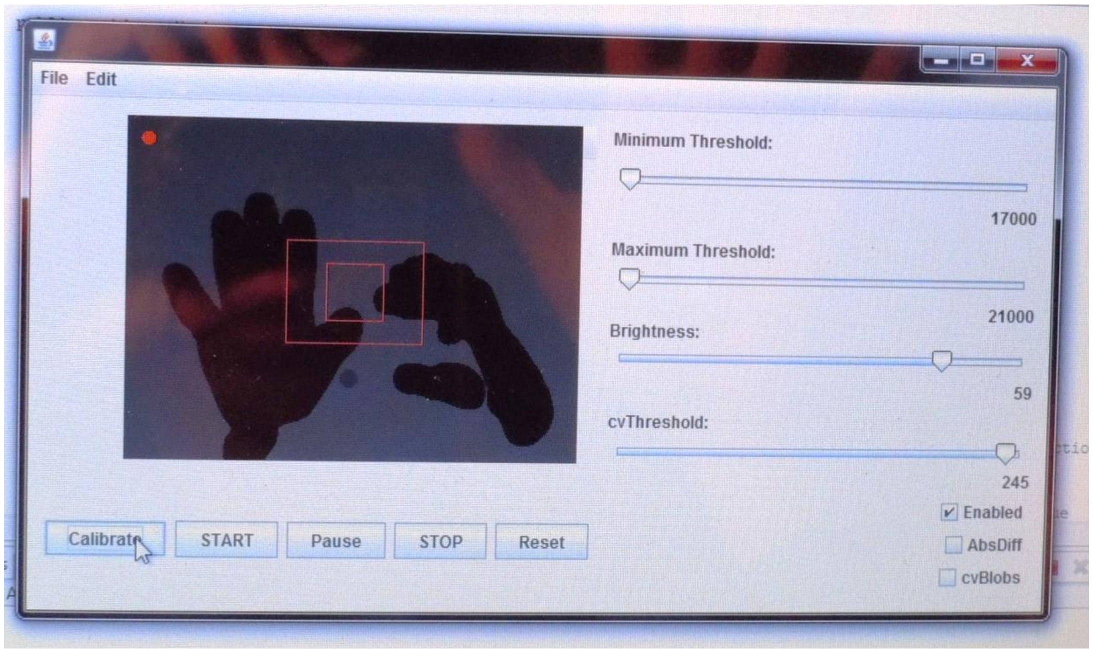

Minimum Threshold: 17000
Maximum Threshold: 21000
Brightness: 59
cvThreshold: 245
[✓ Enabled]
AbsDiff
cvBlobs

Figure 10 HGR Release Version 2.0 Hand Detection Module in operation

5.3. HGR Release Version 3.0

As shown from Figure 5 in section 4.4.1, version 3.0 has a face detection module added to the software. The addition of this module allows extra skin coloured objects to be filtered thus reducing the area needed to calculate for the hand detection. Furthermore, different modes of operation have been added. The user can choose between Mouse Control and Browsing Control. The features of version 3.0 are as described in section 4.4.

5.4. Informal Software Performance Testing

The HGR software has been tested before each release of the software. Version 1.0 relied on users wearing indicators to allow accurate detections. Although the detection and tracking were accurate, it required the background to have no other similar coloured objects. This led to the change in detection method for version 2.0.

Version 2.0 implemented the hand detection without indicators. Under low lighting condition, the system performs exceptionally accurate as seen on Figure 10. This version, however, was not usable since the environment lighting changes the result of detections constantly. Furthermore, the system also detected user’s face. As a result, the software crashed and required a restart each time.

Version 3.0 of the HGR software removed many problems presented in version 2.0 but not everything. The use of face detection allowed the system to filter out user’s face as a result of extra image processing on each frame of video. The only problem that remained in version 3.0 was the constant change in lighting. In this version, the skin colour detection had improved in the aspects of operating environment. The only constraint left is that user’s background must not contain any windows. This is due to the fact that the light coming from the windows greatly affects the visibility of the low-ended consumer webcam.

In addition, all versions of HGR have been tested with users of different ethnicity. The only constraint to successful skin detection relies on the amount of light is reflected from the user; too little or too much light on the user will stop the image processing module from recognizing the subject.

As a result of adding the face detection module to version 3.0, the processing speed of each captured video frame also decreased by 500-700 milliseconds. The increase in processing time diminishes the flow of cursor movements; however, it removes the problem of recognizing the user’s face as a hand.

6. RESULTS

In this section, the objectives achieved, the difficulties and problems faced, and suggestions for future development will be described in detail.

6.1. Objectives Achieved

As outlined in the project proposal, the purpose of this project is to allow users to execute hand gestures that a regular consumer PC webcam can recognize. The primary objective is to have hand detections without using indicators and special webcams. This is achieved during the second release of HGR where the software uses colour filtering techniques to obtain hand recognition. However, after version 3.0, when skin colour detection method is added, the usability of the software is further enhanced. Users no longer need to manually calibrate the system at each start up for hand recognition and do not require special coloured indicators for tracking.

The secondary objective for this project is to convert hand detection results to user beneficial operations. This is achieved by having the two built-in control modes: mouse control and browsing control modes. Users can control mouse cursor movements by just moving one's hands in front of the webcam. By raising two hands, the user can simulate a click on the mouse. In addition, in browsing mode, users can switch, scroll up and scroll down pages in Internet browsers.

Through three different versions of the software, both objectives discussed in the proposal are accomplished. The software is also built in many modules where future add-ons to the project can be easily attained.

6.2. Difficulties and Problems Faced

There are two main problems that occurred throughout this project. The changes in environment lighting mentioned do not allow the program to be usable in all environments. In low light areas, the webcam cannot receive enough reflected light from users' hands. Under intensive lighting conditions such as direct sunlight or in the case shown in Figure 11 below, the webcam cannot detect the skin due to the similarity in colour of the hand with the background. By combining all the algorithms discussed in section 2 of this report, the problem still persists.

The second problem arose within the gesture recognition module. Initially, the gesture module used two separate **threads** to recognize actions. However, a **synchronization** problem existed when both threads were trying to read and write to the same variable storing the distance that the hand had traveled. This problem was removed in version 3.0 when the threads were simplified to a single thread and the distance variable converted to a **global** type.

Both of the major problems faced in this project involved heavy research of different methods to produce the correct output. The problem of the environment lighting that remains to be unsolved likely requires advanced mathematical analysis that cannot be learned during the time frame of this project.

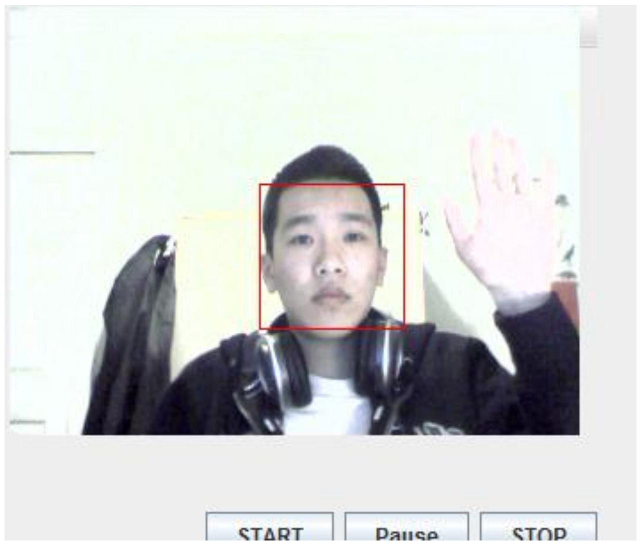


Figure 11 A captured frame shows the hand that is hard to distinguish from the background under the naked eye.

6.3. Suggestion for Future Development

For future development, it will require further research in advanced mathematical methods in image processing. This includes new detection algorithms from a mathematical point of view and methods to calculate lighting offsets and adjustments based on the user operating environment. Additional research and investments into hardware components such as infra-red cameras or two side-by-side webcams will also help to solve the problem that I faced in this project.

7. REPORT CONCLUSIONS

This report investigated the alternative solution to using integrated hardware components with the personal computer. The project presented a program that allowed user to perform hand gestures for easy software control. Through various algorithms and detection techniques, the program achieved simple hand detection and identified the accurate gestures executed by the user.

In achieving the hand detection module, the program utilized the combination of RSB, HSB, skin colour and face detection methods to determine user hand locations. After the hand location was defined, geometry point conversion and gesture recognition modules were called upon to execute user gesture commands.

Under extensive testing done in different locations and lighting conditions, the program showed that:

- Environment lighting changes could dramatically decrease the robustness of the software system in detecting hands
- The addition of face detection module could decrease the performance speed of image processing but increase the accuracy of hand detections

For future developments, it would be ideal to research into advanced mathematical materials for image processing and investigate on different hardware solutions that would result in more accurate hand detections. Not only did this project show the different gesture operations that could be done by the users but it also demonstrated the potential in simplifying user interactions with personal computers and hardware systems.

REFERENCES

[1] Santa Barbara Instrument Group Inc., (2010, August), “Filters for Color Imaging, Narrow Band Imaging and Photometry,” [Online]. Available: http://www.sbig.com/products/filters.htm

[2] (2011, March), “HSL and HSV,” *Wikipedia* [Online], Available: http://en.wikipedia.org/wiki/HSL_and_HSV

[3] D. Munoz, F. Andrijauskas, and G. A. Archanjo, “Marvin Image Processing Framework,” [Online], Available: http://marvinproject.sourceforge.net/en/index.html

[4] C. Sandarenu, “OpenCV – Haar Training Resources,” Sandarenu’s Blog [Online], Available: http://sandarenu.blogspot.com/2009/03/opencv-haar-training-resources.html

[5] S. Cousot, D. E. Stanely, and Intel Corp., “absDiff(),” Processing and Java Library [Online], Available: http://ubaa.net/shared/processing/opencv/opencv_absdiff.html

[6] R. Fisher, S. Perkins, A. Walker and E. Wolfart, (2003), “Intensity Histogram,” [Online], Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/histgram.htm

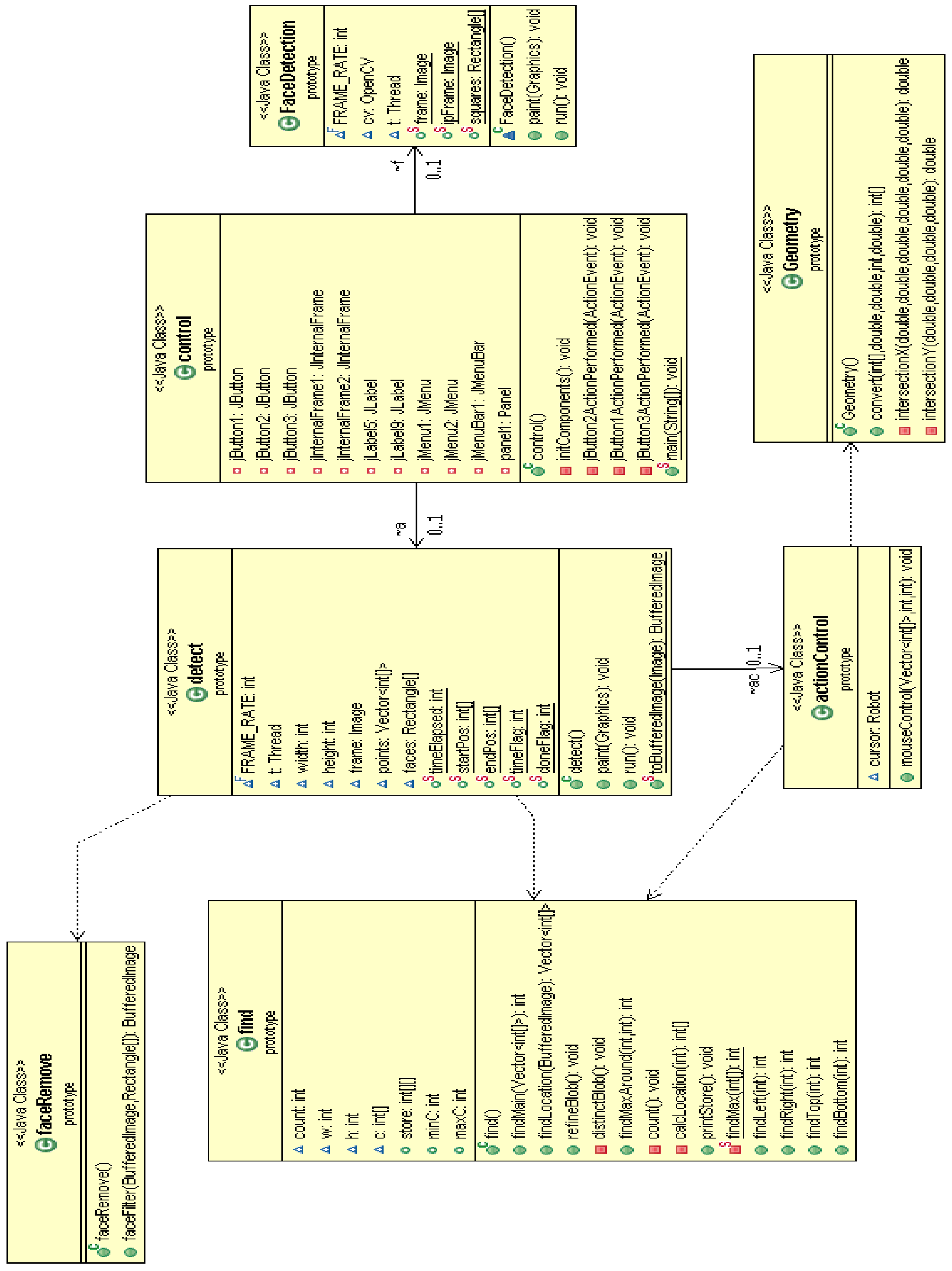
[7] Intel Corp, “OpenCV Wiki,” OpenCV Library [Online], Available: http://opencv.willowgarage.com/wiki/

[8] D. Munoz, F. Andrijauskas, and G. A. Archanjo, “Skin Color Detection,” Marvin Image Processing Framework [Online], Available: http://marvinproject.sourceforge.net/sourceCodes/plugins/image/color/skinColorDetection/SkinColorDetection.java

[9] A. Jepson, “Image Segmentation,” University of Toronto [Online], Available: http://www.cs.toronto.edu/~jepson/csc2503/segmentation.pdf

APPENDIX

HGR Software Class Diagram



<<Java Class>>
FaceDetection
prototype
FRAME_RATE: int
cv: OpenCV
t: Thread
frame: Image
pFrame: Image
squares: Rectangle[]
FaceDetection()
paint(Graphics): void
run(): void
~f
0..1
<<Java Class>>
control
prototype
jButton1: JButton
jButton2: JButton
jButton3: JButton
jInternalFrame1: JInternalFrame
jInternalFrame2: JInternalFrame
jLabel5: JLabel
jLabel9: JLabel
jMenu1: JMenu
jMenu2: JMenu
jMenuBar1: JMenuBar
panel1: Panel
control()
initComponents(): void
jButton2ActionPerformed(ActionEvent): void
jButton1ActionPerformed(ActionEvent): void
jButton3ActionPerformed(ActionEvent): void
main(String[]): void
~a
0..1
<<Java Class>>
detect
prototype
FRAME_RATE: int
t: Thread
width: int
height: int
frame: Image
points: Vector<int[]>
faces: Rectangle[]
timeElapsed: int
startPos: int[]
endPos: int[]
timeFlag: int
doneFlag: int
detect()
paint(Graphics): void
run(): void
toBufferedImage(Image): BufferedImage
~ac
0..1
<<Java Class>>
Geometry
prototype
Geometry()
convert(int[],double,double,int,double): int[]
intersectionX(double,double,double,double,double,double): double
intersectionY(double,double,double,double): double
<<Java Class>>
actionControl
prototype
cursor: Robot
mouseControl(Vector<int[]>,int,int): void
<<Java Class>>
faceRemove
prototype
faceRemove()
faceFilter(BufferedImage,Rectangle[]): BufferedImage
<<Java Class>>
find
prototype
count: int
w: int
h: int
c: int[]
store: int[][]
minC: int
maxC: int
find()
findMain(Vector<int[]>): int
findLocation(BufferedImage): Vector<int[]>
refineBlob(): void
distinctBlob(): void
findMaxAround(int,int): int
count(): void
calcLocation(int): int[]
printStore(): void
findMax(int[]): int
findLeft(int): int
findRight(int): int
findTop(int): int
findBottom(int): int