CSIT 6910 Independent Project

Real-time body tracking of a teacher for automatic dimming of overlapping screen areas for a large display device being used for teaching

Student: ZHAI Yifan

Supervisor: Prof. David Rossiter

Content

1. INTRODUCTION

1.1 Overview

Nowadays, there are numerous large display devices, and traditional blackboard is being replaced gradually. Although it is greener than using eraser and chalk to write on blackboard, and more convenient for the teacher so that he/she does not need use much time writing repeatedly, the power usage still has room for improvement. A 51-inch television would cost nearly 200 Watts in an hour, and a larger one would certainly cost more power. It would be much better if, when someone such as a teacher stands in front of screen, the part of the screen overlapped by the teacher can dim automatically to save unnecessary power usage.

Another situation is that if something completely blocks the screen, then the power usage is entirely wasted, why not the display device just dims the whole screen to save power? Though it does not make difference for a projector because of the way it works, it can actually save power for other devices like televisions. Some modern monitors have already got a similar function – adjust the whole screen brightness according to the image displayed.

1.2 Project Objectives

The final objective of this project is to develop an application which can be put into real usage. In order to realize this goal, the application should be implemented in the follow aspects:

- This application should be accurate, which means it should recognize barrier, and should not regard other places as overlapping.
- This application should be fast in order to provide real-time service.
- This application should have a good availability, which means, it cannot crash at all.

The project includes graph recognition, software programming, and research on related method of real-time body tracking.

2. SYSTEM DESIGN

This chapter introduces the design of the whole system, from an overview to some key algorithms

.

2.1 System Workflow

The application is divided into several steps, which can be described by the flow chart below.

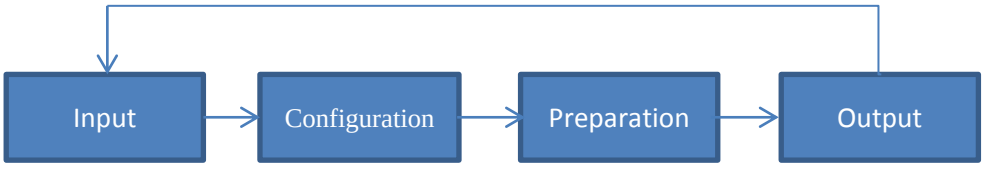

Figure 2.1 System Workflow Chart

The whole application workflow is mainly divided into the four stages:

(1) Input stage. Application would let user choose the output screen's location by four points, which would be used in later recognition. The system also provides a simple auto-recognition.

The system has two inputs: one is (A) the image of the teacher standing in front of the screen. Another is (B) the computer image which is supposed to be projected. Then the system would work out which part of (B) needs to be hidden, hides the appropriate part and sends the resulting image to the output device (television/computer screen/ projector).

Input (A) need a web camera to detect the image of the teacher and television, and then analyze that to determine the overlap. Input (B) is from an existing source, most likely from a computer, laptop, etc.

(2) Configuration stage. The application makes configuration for the "black" color which would be used in later output stage. It is necessary because the "black" from the view of web camera is different from the pure black which is 0:0:0 represented by RGB.

(3) Preparation Stage. The application just outputs the screenshot image. This stage is to allow user to choose when to start the dimming process.

(4) Output Stage. The application grabs the screenshot and the web-camera image, and uses the two images to calculate the barrier in front of the screen. After getting the result image, the application would process the image to clear noise points and then output the final image.

2.2 Class Diagram

The basic idea of this part is to introduce the class design of this system. Figure 2.2 is the class diagram.

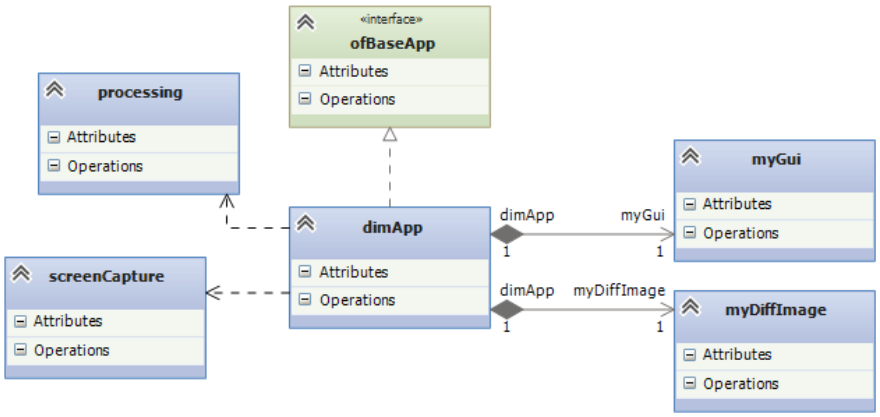

Figure 2.2 Class Diagram in Design Stage

- Class dimAmpp inherits ofBaseApp and is the core of the whole application, and it manages the workflow of the system.
- Class screenCapture is responsible for grabbing the screenshot using system-related API.
- Class processing is a utility class, which provides some useful functions like map and constrain.
- Class myGui is a panel for setting, including threshold, screen size, etc.
- Class myDiffImage is the image processing class, which takes screenshot and webcam images as input. Most algorithms are in this class.

2.3 Image Warping

In the input stage, with the given four points from the user, the webcam image's perspective would be warped, like figure 2.3, because the screen from the view of the webcam usually is not a standard rectangle, which would cause problems during later comparing.

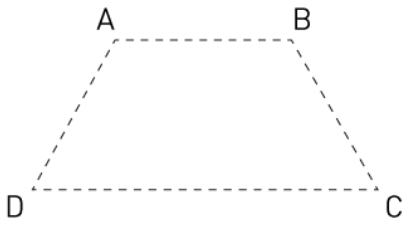

Figure 2.3 Warp Image According To 4 Points

We can see in figure 2.4, the image in the right is the warping result of the image in the left according to the four points. What we do here is to use a perspective transformation, which uses the equations of a ratio of linear polynomials may be expressed as:

$$U = (a_0 + a_1X + a_2Y)/(1+c_1X+c_2Y),\ V = (b_0 + b_1X + b_2Y)/(1+c_1X+c_2Y)$$

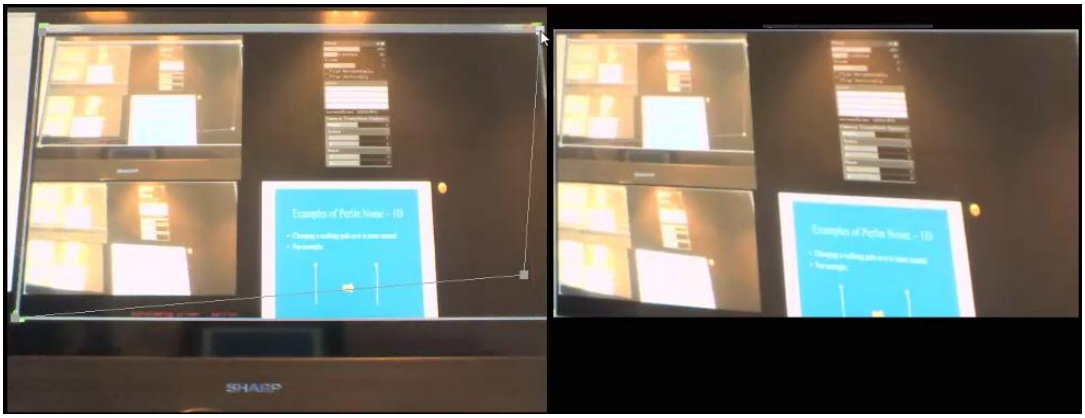

Figure 2.4 Image Warping from Irregular Quadrilateral

Figure 2.5 illustrates a real example. Because the webcam and the center of the screen is not on a horizontal line, we choose the exact border of the application window, and the webcam image is warped to the image in the right.

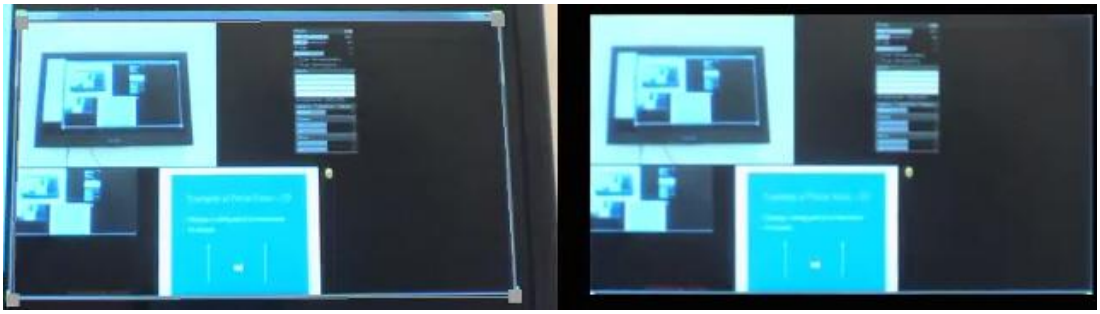

Figure 2.5 Real Image Warping of the System

2.4 Two Detection Method Comparison

In this section, I will introduce two algorithms I have used to detect the barrier. The first one is motion detection, which is discarded because of its limitation. The latter one is pixel comparison, which is used in the final result.

Actually, barrier recognition is not as easy as other detection like faces and people, which can be done by machine learning like HoG Descriptors, SVM, etc. Because barrier maybe anything, a box, a hand, an upper body, etc. What's more, it is possible that what we have to display contains a person's face or a full body image, and we certainly do not want the application to dim that. I used two algorithms and chose one of them as my basic detection method.

2.4.1 Motion Detection

The idea came from My little piece of Privacy project by Niklas Roy[1], which used motion detection to move the curtain. Motion detection is similar to background subtraction, while they are different output from the same processing. There have been some projects using background estimation to extract the human body[2][3].

Motion Detection is done by comparing the current picture to the previous one to figure out

the movement of pixels.

2.4.2 Pixel Comparison

Pixel comparison, which is finally chosen in the system, is to compare two corresponding pixel of two input, and decides whether it is overlapped by the "difference" of the two pixels. Suppose we have two images, and we figure out two corresponding points P1(x1, y1) and P2(x2, y2). We have

$$D_1 = abs(R_{P1} - R_{P2}) + abs(G_{P1} - G_{P2}) + abs(B_{P1} - B_{P2})$$

If the difference is over the threshold, we regard this pixel as a barrier pixel.
However, D_1 is not enough, because different color may have close RGB, and a smaller threshold would produce too many noisy pixels. So we add a further assessment:

$$D_2 = abs(Hue_{p1} - Hue_{p2}),\ Saturation_{P1} > threshold \text{ and } Saturation_{P2} > threshold.$$

The new value puts HSV color space into consideration, which performs well when two colors have close RGB distance. Saturation threshold here is necessary because Hue value is meaningless when saturation is too low.

The two assessments are used together to compensate the limitation of each other.

2.5 Denoise

When we compare pixels from two inputs, and regard them as different from the method I introduced in last section, we set the output pixel to purely black. However, the two inputs cannot which actually have one-to-one relationship, which means an object at pixel(100, 200) in webcam image might be at pixel (101, 199) in screenshot image. This offset may produce some noisy points at the edge.

In my system, I first convert the result image into gray scale image so that the boundary can be clear[4]. Then, I find contour in the gray scale image using border following[5].

Figure 3.6 shows an illustration of the process of border following algorithm. The left-hand figures show the pixel values and the right-hand the extracted structures among borders (ob for outer border, and hb for hole border). The circled pixels are the starting points of border following.

The input binary picture is given in (a). When the raster scan reaches the circled l-pixel, we find that it satisfies the condition for the starting point of an outer border. The sequential number 2 is assigned to the border, because the number 1 is set aside for the frame of the picture. We also find out from Table 1 that its parent border is the frame.

Then the border following starts and all the pixel values on that border are changed to either 2 or - 2, as shown in (b). After following the entire border, raster scan resumes.

The next border following starts at the circled pixel in (b), which satisfies the condition for the starting point of a hole border. Note that a point having a negative value cannot be a starting point of border following. The parent border of this border (with the number 3) is found to be the outer border with the number 2. During the border following, two pixels with value 2 are changed to - 3, and two pixels with value 1 are changed to 3. All the other already visited pixels (with value 2) are unchanged. The steps go on in a similar manner and we obtain the final results (e).

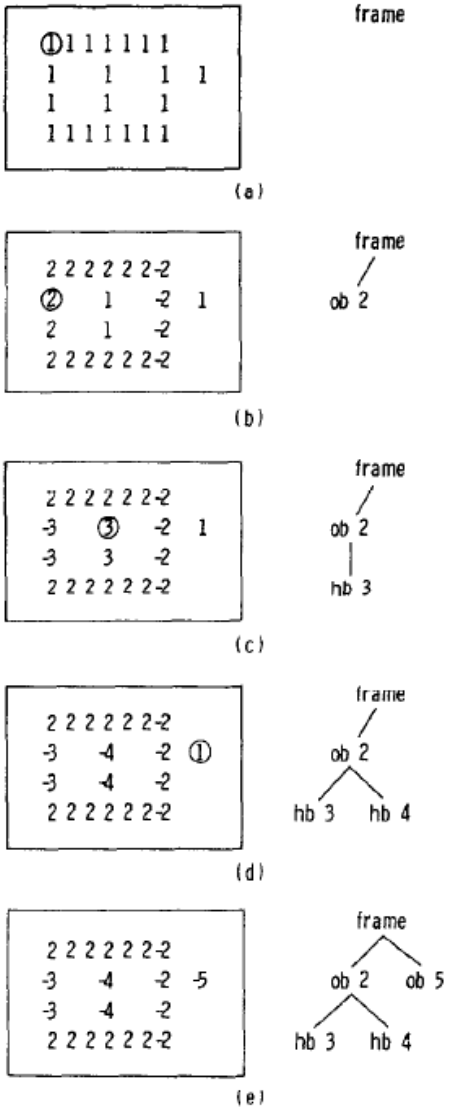

Figure 2.6 an illustration of the process of Border Following Algorithm

As the contour in our system has some conventions that they should not be too small or too big (block the whole screen), I set a minimum area and maximum area to the contour. Furthermore, I limit the contour number to be less than 5 to prevent noise.

2.6 Window Structure

This section introduces the window structure of the software, mainly focus on the structure of the first stage.

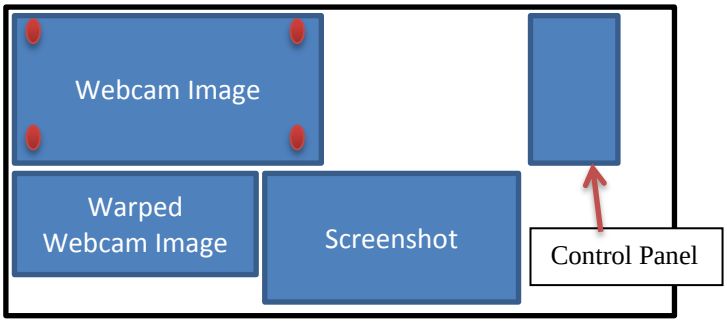

Figure 2.7 Input Stage – Window Structure

Figure 2.7 shows the window structure for input stage, which consists of four areas and provides usability to user.

In the first stage – input stage, user can see an interface like the figure 3.4 which allows input including:

(1) Location of output screen, which is specified by four points. The webcam image would then be warped and displayed in the bottom-left area. The location would first be automatically detected by the software. The four points would move according to the webcam image, and they would only stop moving automatically after the first mouse click, which implies the user starts to select the specific area.
(2) Control panel, which is displayed by a list and some bars allow user to choose a value from a range.

The right-bottom area simply outputs a resized screenshot in the main monitor, which might be playing a PPT file. The top-left area is the webcam image, and the image below is that warped image according to the four points from the webcam image.

3. LIBRARIES USED

3.1 OpenFrameworks

OpenFrameworks is an open source C++ toolkit for creative coding[6]. OpenFrameworks is written in C++ and runs on Windows, Mac OS X, Linux, iOS, and Android. It is maintained by Zachary Lieberman, Theo Watson and Arturo Castro with contributions by other members of the openFrameworks community.
Its emphasis on "creative" uses draws parallels to Processing[7] as both projects present a simplified interface to powerful libraries for media, hardware and communication. openFrameworks's main difference from Processing is that it is written in C++, instead of Java.

3.2 OpenCV

OpenCV is a library of programming functions mainly aimed at real-time computer vision, developed by Intel, and now supported by Willow Garage and Itseez. It is free for use under the open source BSD license. The library is cross-platform. It focuses mainly on real-time image processing. If the library finds Intel's Integrated Performance Primitives on the system, it will use these proprietary optimized routines to accelerate itself. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics[8].

4. SOFTWARE WALK-THROUGH

In this chapter, I will demonstrate the final result of the project.

4.1 Configuration

We can see in the figure 4.1 below, the interface is divided into four pieces.

The top-left one is the picture from webcam, and there are four points inside the picture which allows user to specify the screen's location. Here the four points can form an irregular quadrilateral, and the software would transform it into a rectangle picture, which is shown in the bottom-left area.

In the right of the webcam image is a control panel, which allows user to set some parameters for later usage, like RGB threshold, HSV threshold, dim color, and so on. User can also drag the panel and put it somewhere else.

The bottom-middle area shows the screenshot, which is what we want to display in the second displayer, and the base image of later calculation.

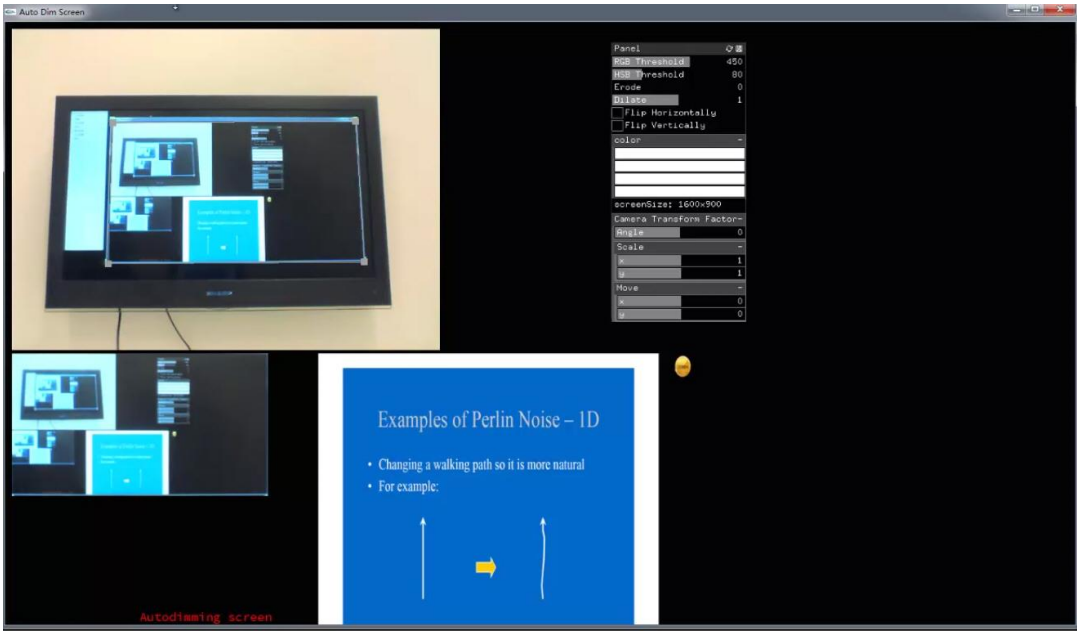

Figure 4.1 First Stage of Application

The second configuration stage is quite direct. The software just draws two pictures one by one and remembers the dim color from the view of webcam. We need two pictures because some displayers would change their light according to the image, while the brightness would decrease significantly, and the dimming color of each pixel would thus be remembered as a quite different color from that would be in later output stage. So what I do here is to draw two images respectively, while the first one has a pure black in its left side, and the second one has a pure black in its right side.

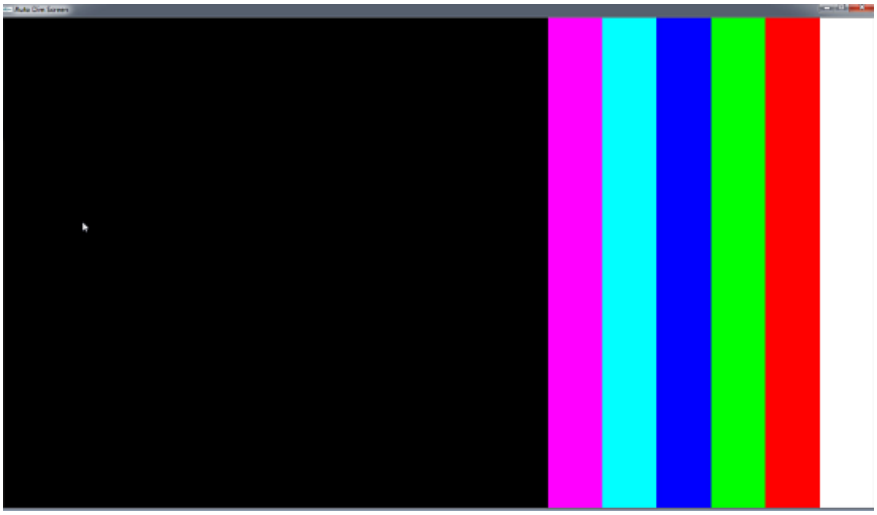

Figure 4.2 Configuration Stage A

Figure 4.2 and 4.3 show how the two configuration stages work, while the first one configures dim color in the left and the latter one configures dim color in the right. After entering the configuration stage, the first image would stay for 2 seconds, and then the application would automatically display the second image. After another 2 seconds, the application would enter the output stage.

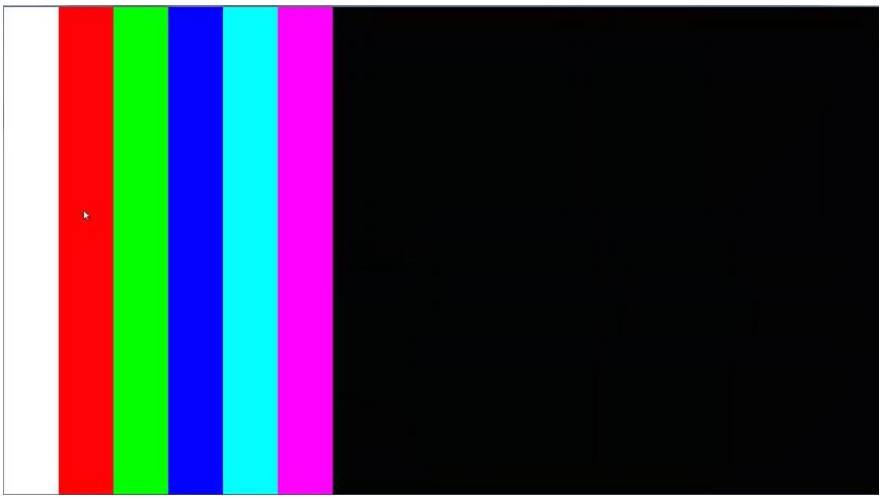

Figure 4.3 Configuration Stage B

The final stage before output stage is Prepare Stage. It just displays the screenshot, and waits for user's input to enter output stage.

4.2 Output Stage

The output stage fills one image into the software, which is the result image. The software would make the area which is regarded as barrier dark color (which is default set as black). We can also see that there are two lines of words in the bottom-right corner, which shows the two thresholds, and the user can use keyboard to control these values.

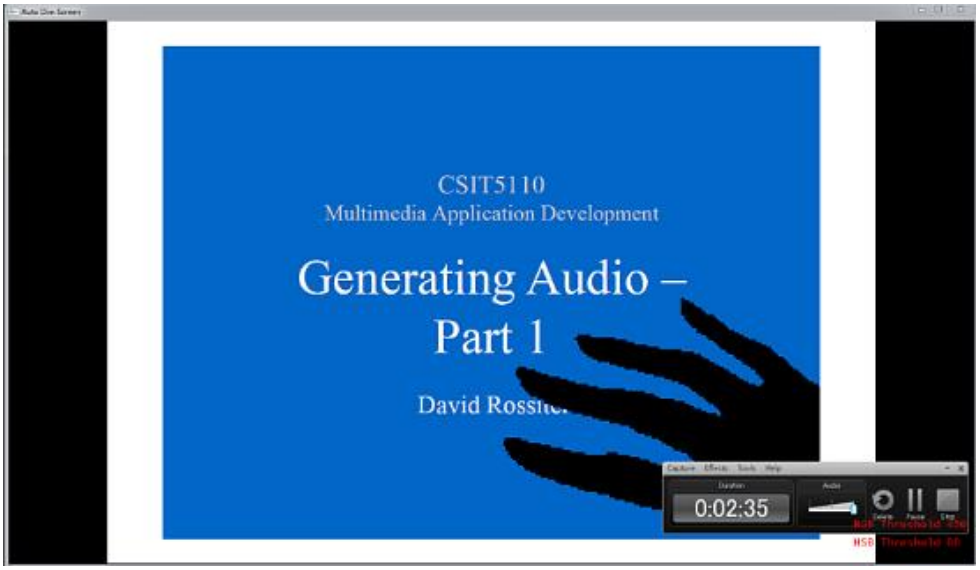

Figure 4.4 Output Stage - Hand

The output stage fills one image into the software, which is the result image. The software would make the area which is regarded as barrier dark color (which is default set as black).

For those dimming area, when later the barrier moves out of it, the software would detect that by comparing the pixel to the dimming color of the pixel which we got from the configuration stage, when the difference is really small, this dimming area would be clear.

5. FURTHER DEVELOPMENT

Since there is limited time, the basic structure of the project is done and there are many possibilities for further development and improvement, such as:

1. The interface of the application can be improved to make it become friendlier.
2. The software can run in full-screen mode, which is now set as 1600*900 window mode.
3. More functions can be added, such as gesture recognition so that teacher can use hand to control the screen to draw something.
4. More accurate barrier recognition and faster speed.

6. CONCLUSION

The aim of this project is to develop a software which can be used for teaching in meeting rooms and classrooms and save power. Currently, the software can be put in real usage and perform quite well, especially when the main monitor is playing some black and white PPT or the image does not have too many color changes. With the help of openFramework, the software has also quite clear workflow, and is able to scale and extend its features easily. The performance is also efficient, while the software only takes near 0.2 - 0.3 seconds delay, which is fast enough to provide real-time usage. However, it still has some minor issues like it sometimes produces noise, and sometimes there are holes in dim contours.

7. REFERENCES

[1] My little piece of Privacy, http://www.niklasroy.com/project/88/, retrieved on 25th Nov 2013.

[2] Yoo, Jang-Hee, et al. "Extracting Human Body based on Background Estimation in Modified HLS Color Space."

[3] Rakibe, Rupali S., and Bharati D. Patil. "Background Subtraction Algorithm Based Human Motion Detection."

[4] Martin, David R., Charless C. Fowlkes, and Jitendra Malik. "Learning to detect natural image boundaries using local brightness, color, and texture cues." Pattern Analysis and Machine Intelligence, IEEE Transactions on 26, no. 5 (2004): 530-549.

[5] Suzuki, S. and Abe, K., Topological Structural Analysis of Digitized Binary Images by Border Following. CVGIP 30 1, pp 32-46 (1985)

[6] OpenFrameworks, http://www.openframeworks.cc/ retrieved on 25th Nov 2013.

[7] Processing, http://processing.org/

[8] OpenCV, http://opencv.org/, retrieved on 29th Nov 2013.

8. APPENDIX

Minutes of 1st Meeting

Date: 24/09/2013
Time: 03:40 p.m.
Place: Rm.3512
Attending: Prof. Rossiter, Zhai Yifan
Absent: None
Recorder: Zhai Yifan

1. Approval of Minutes
Since this is the first meeting, there is no approval of minutes of previous meeting.

2. Report on Progress
Since this is the first meeting, there is no progress to be reported.

3. Discussion Items and Things To Do

- Project idea
- Development platform
- Final result
- Project development equipment

4. Meeting Adjournment
The meeting was adjourned at 04:15 p.m.

Minutes of 2nd Meeting

Date: 17/10/2013
Time: 11:30 a.m.
Place: Rm.3512
Attending: Prof. Rossiter, Zhai Yifan
Absent: None
Recorder: Zhai Yifan

1. Approval of Minutes
The minutes of the last meeting were approved without amendment.

2. Report on Progress
Zhai Yifan changed the platform from processing to C++ with openframworks, and made an easy application to show the first stage progress.
He compared the two captured image and figured the moving objects to recognize body.
The method however, cannot work if the person is not moving.

3. Discussion Items and Things To Do

- To try another method which compares screenshot with captured image.
- To modify the GUI so that user can select the screen position in the captured image.

4. Meeting Adjournment
The meeting was adjourned at 12:10 a.m.

Minutes of 3rd Meeting

Date: 21/10/2013

Time: 12:00 a.m.

Place: Rm.3512
Attending: Prof. Rossiter, Zhai Yifan
Absent: None
Recorder: Zhai Yifan

1. Approval of Minutes
The minutes of the last meeting were approved without amendment.

2. Report on Progress
Zhai Yifan tried the new method talked about in last meeting, but it does not show a good result.

3. Discussion Items and Things To Do
- To figure out what is wrong.
- To down sampling the screenshot so that the update-speed can be faster.

4. Meeting Adjournment
The meeting was adjourned at 12:30 a.m.

Minutes of 4th Meeting

Date: 07/11/2013

Time: 11:15 a.m.

Place: Rm.3512
Attending: Prof. Rossiter, Zhai Yifan
Absent: None
Recorder: Zhai Yifan

1. Approval of Minutes
The minutes of the last meeting were approved without amendment.

2. Report on Progress
The update function call for output now takes only 0.1 – 0.2 seconds, while it took almost 1 second in previous version.

The program now can recognizes barrier in front of the screen and set the overlapping area into black color, and figures out which pixel is likely to be dim color pixel by adding a configuration stage before output stage.

The program can also somehow automatically configure the output screen's part in the captured image by finding the contour.

3. Discussion Items and Things To Do
- Add HSB into algorithm so that the application can recognize barrier which has similar RGB with screen pixels.
- To improve the algorithm so that it can show a great result with colorful PPT.

4. Meeting Adjournment
The meeting was adjourned at 12:30 a.m.

Minutes of 5th Meeting

Date: 04/12/2013

Time: 12:00 a.m.

Place: Rm.3512
Attending: Prof. Rossiter, Zhai Yifan
Absent: None
Recorder: Zhai Yifan

1. Approval of Minutes
The minutes of the last meeting were approved without amendment.

2. Report on Progress
The application can now work well with colorful PPT, and it can recognize barrier which has close RGB with pixels behind it by adding a HSB comparing function.
Noise is cleared by converting result image into gray scale image and finding large contours.

3. Discussion Items and Things To Do
- Finish the report.
- Make the video shorter and add speech to explain how it works.

4. Meeting Adjournment
The meeting was adjourned at 13:00 a.m.