**Final Project:**

**Gesture Trainer for ASL Alphabet**

**Jason Mann & Lindsey Heller**

**05/14/2013**

**COMS 4735: Visual Interfaces for Computers**

Table of Contents

**Application OverView 4**

**Assignment Detail 5**

**Part 1: Building Features and Descriptions 5**

**Part 2: Describing Compact Spatial Relations 6**

**Part 3: Source and Goal Description 7**

**Part 4: Path Generation 8**

**OutPut 9**

**Appendix A: Part one 9**

**Appendix B: Part Two 10**

**Appendix C: Part Three 20**

**Code 22**

**MapReader.pro 22**

**main.cpp 22**

**mapreader.h 23**

**mapreader.cpp 25**

**mapreader.ui 38**

**clickablelabel.h 39**

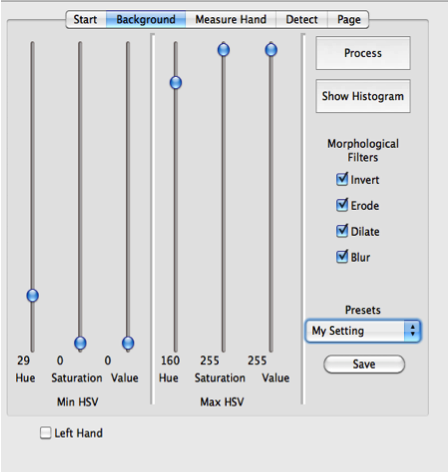
# Introduction

**This project was designed for an assignment in COMS 4735 Visual Interfaces to Computers under the supervision of Professor John Kender at Columbia University in the Spring of 2013. The goal of this project was to create a Gesture Library Training Tool; the purpose of this training tool is to train a user to use a specific Gesture Library, in order to ensure a user has the ability to correctly perform the set of standard gestures of the system on which the user is being trained. The idea to create this tool stemmed from Assignment 1 of the course, of which the goal was to filter out skin regions in imagery, detect regions belonging to hands, and then interpret specific pre-defined gestures from the captures. We chose examples from the American Sign Language (ASL) alphabet to be the gestures in our example Gesture Library.**

**The project is coded in C++, relying on OpenCV 2.4.3 for computer vision algorithms, and QT 4.8 (and QT Creator) for the design of the GUI forms. Some code and inspiration was found in the OpenCV2 Cookbook by Robert Laganiere, specifically the design of the SkinDetector and Controller classes (though not necessarily the implementation). The program uses live video capture to analyze static hand gestures.**

# Background

**The skin regions are differentiated from the background by manual configuration of an HSV thresholding mask (setting min and max for each value) using sliders and by using the morphological filters made available—invert, erode, dilate, and blur. [Figure 1] These sliders and filter setting are found on the form on the “Background” tab of the Gesture Tool. The masks of these settings are implemented using cv::Range() to provide a binary image of skin regions, the filters are then applied to this binary image. Once the correct mask is found, the user can save this preferred preset to eliminate requiring the user to find this optimal mask each time the Gesture Tool is opened, since often the perfect mask can be hard to find. After the mask is set, the user should then proceed to the “Measure Hand” tab, which will take measurements of the user’s hand that are necessarily for distinguishing correct form incorrect gestures later on.**

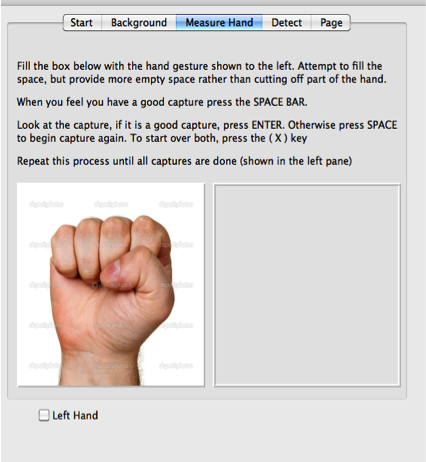


**Figure 1: The “Background” Tab**

**Any faces in the frame are identified and eliminated from consideration immediately, using OpenCV’s built-in Haar cascae classifier (frontalface\_alt\_tree). Any other skin regions on the frame are considered to be contours using cv::findContours. The frames that will be analyzed to ensure the user is making the correct hand gesture are captured at regular intervals (hard coded to 3 seconds) to allow the user time to configure the next gesture.**

# Measurements

**Once on the measurement tab the user is prompted to display two gestures for the tool to capture: a fist and an open palm. This step is crucial for the program to functions as important measurements necessary to properly identifiy the gestures in the Gesture Library are taken at this point from these captures.**



**Figure 2: The “Measure Hand” Tab**

*Directions*

**When the user proceeds to this page they are prompted to fill the red box that appears on the screen with a specific hand gesture. The user is given the option to try multiple times to capture their hand, or even redo the entire process, the until they are comfortable with the captures.**

*What is a user is left handed? The box is on the right!*

**Not to worry, we have taken care of that! At the bottom of the pane there is a check box with a label Left hand. Checking this box will move the red box over, so that a user can use their left hand. Checking this box also lets the tool know which hand you are using (if the box is unchecked it assumes a user is using their right hand) and is necessary for making the proper calculations later on in the tool.**

*Why Do We Require This*

**Truthfully for what is implemented currently, nothing comes of the capture of th fist. However, from the capture of the open palm a measure of the angle from the tip of each finger to the center of the palm is determined. This is basis of how certain calculations and recognition of hand gestures are done in this gesture tool.**

# Detection of Statistics

*Circle Time*

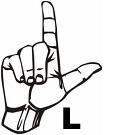
*Fingerlings*

# Detection of Gestures

**The gestures from the ASL alphabet that we currently are able to teach and determine correctness are displayed below:**

|  |
| --- |
| **I** |
| **T** |
| **W** |

|  |
| --- |
| **A** |
| **L** |
| **V** |
| **Y** |

****

**As you can see from the images above, there are generally two classes that these gestures fall under: *FIST* or *PALM*. The gesture to sign A and T belong to the *FIST* class and the gesture to sign I, L, V, W, and Y belong to the *PALM* class. To try out any of these gestures and view that our gesture tool can in fact distinguish between the different letters, checkout the “Detect” Tab. Statistics used for debugging and gathering information is printed out in the text browser to the left and the gesture detected printed in the upper left corner of the right hand side, above the picture of the live capture.**

*Fist Class*

**A gesture is classified to be within the *FIST* class if there are no significant figures detected outside of the minimum enclosing circle detected hand. The change in slope along the top of the fist is used to determine the difference between an A and a T gesture.**

*Palm Class*

**A gesture is classified to within the *PALM* class if there are significant figures detected above the minimum enclosing circle of the detected hand. Both the number of fingers detected and a comparison between the difference of the angles of the detected fingures calculated from the image captured in the “Measure Hand” Tab is used to determine which gesture the user is attempting.**

# Training

**In the “Trainer” tab is where the user can actually go through the training process. The images of the gestures the training tool aims to teach (the same displayed above in the “Detection of Gestures” section) is displayed for the user to see and try to emulate. The program is satisfied that a user is correctly displaying the gestures if after 3 captures, the program resolves that the user is making the gesture requested. Once the user has successfully emulated the gesture requested, the program tells the user of their success and then moves on to the next image. If there is a failure, the program encourages the user to try again.**

# Challenges & Limitations

Background & Lighting

**Good lighting and background was a crucial part in developing this gesture tool, as it generally determined how well the background mask would be configured. In light of this we faced two problems in creating our Gesture Training Tool in some buildings where there wasn’t good lighting and/or background to work with:**

1. **Sometimes, the bad lighting and/or a less than ideal background led to making incorrect assumptions and measurements while developing that later had to be revisited and changed when in a place with better lighting.**
2. **Sometimes the lighting was so horrible, that it was even impossible to develop in the room and we had to move elsewhere.**

**The areas that generally caused a problem for us were generally sunlit areas, areas with uni-directional lighting, and areas that have flesh colored tones, or light birch wood.**

Auto-focus and Auto-exposure

**On modern web-cameras, the auto-focus and auto-exposure caused us some problems along the way as it caused a once perfect mask to no longer be a valid fit.**

My Partner and I… We are VERY Different.

**Lindsey is a lefty, Jason is a righty. Lindsey has small hands, and Jason has bigger hands not mention a deformed pinky that caused some trouble. Truthfully, this was both a blessing and a curse. We were able to catch many bugs by testing the work that we each did using our own hands, but it challenged us to really make the detection more general.**

Version Control Couldn’t Help Us Here

**It was hard to split up the work in creating this Gesture Tool so that we were each able to work on things where we wouldn’t affect the other. Unfortunately, many times we had to revisit problems we though we solved since the work another did interfered with the solution to the problem.**

Brain Geometry does not always Measure to Actual Geometry

**We had awesome ideas on how to implement the gesture detection for this Gesture Tool, however, the great ideas we had often did not translate well into real life measurements. For example, we initially though we could tell the different between a fist and a palm using several different aspect ratios we figured would work well… to our dismay we found that in reality these ratios had no significance in determining the difference between the two classes that we were trying to distinguish between.**

# Conclusion

**While during development we had to decrease the scope of the project that we initially set out to accomplish, we nevertheless ended up developing a really cool program that is a great start for the initial goal that we set out to accomplish. Given more time and coffee we believe that we would have had the ability to finish the entirety of our proposed project.**

**For the most part, our program is successful in recognizing different gestures. Factors such as background, rotation of the hand, etc. sometimes make the recognition unstable although eventually the program is usually able to resolve all issues.**