**Final Project:**

**Gesture Trainer for ASL Alphabet**

**Jason Mann & Lindsey Heller**

**05/14/2013**

**COMS 4735: Visual Interfaces for Computers**

Table of Contents

**Introduction 3**

**BackGround 4**

**Measurements 5**

**Detection of statistics 6**

**Detection of Gestures 8**

**Training 9**

**Challenges and Limitations 10**

**conclusion 11**

**Code 12**

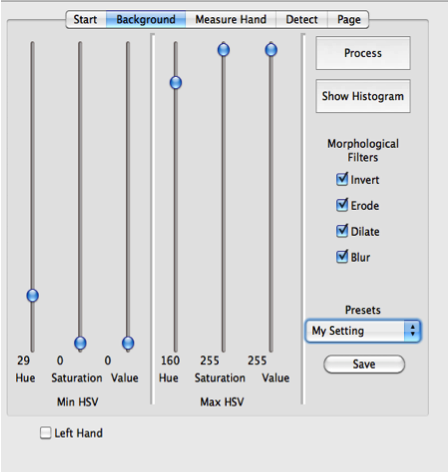
# Introduction

**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; its purpose being to train a user in the use of a specific gesture set in real time. This is 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 the first assignment of the course, in 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 a selection of static letters from the American Sign Language (ASL) alphabet to provide a proof of concept on a well known set of gestures.**

**We implemented this project in C++ (stdlib c++11), relying on OpenCV 2.4.3 for computer vision algorithms, and QT 4.8 (with 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 processes live video capture from any common webcam and can be used in varying environments with the use of our manual background filtering tool.**

# Background

**The skin regions in the live capture are differentiated from the background by manual configuration of an HSV thresholding mask (setting min and max for each value) using sliders in a configuration pane. Further processing options are provided by checkboxes that control the morphological filters made available—invert, erode, dilate, and blur. [Figure 1] These sliders and filter settings are found on the form on the “Background” tab of our tool. The masks of these settings are implemented using cv::Range() to provide a binary thresholded image of skin regions, and the morphological filters are then applied to close and blur the image. A workable mask can be found using the real time view of the filters at work. Once a suitable mask is found for the background environment, the user can save it as a preset, preventing the user from having to re-find an optimal mask each time the Gesture Tool is opened, since often the perfect mask can be difficult to discover. After a mask is set, the user should then proceed to the “Measure Hand” tab, which measures the user’s hand attributes that are necessary for distinguishing correct from incorrect gestures later on.**

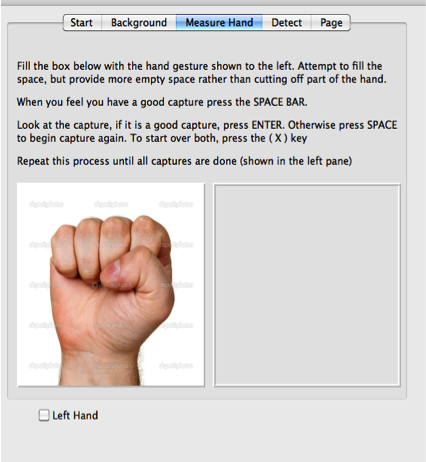


**Figure 1: The “Background” Tab**

**Any faces in the frame are identified and eliminated from consideration immediately, using OpenCV’s built-in Haar cascade classifier (frontalface\_alt\_tree). Any other thresholded regions on the frame not intersecting the face bounding rectangle are considered by their contours using cv::findContours. The largest of these contours is chosen as the hand/arm, while the background configuration step should remove any regions that could be larger.**

# 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 our algorithms to function as important attributes and traits of the users hand are recorded in order to properly identify and distinguish the gestures used in the Gesture Library.**



**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. This box can be positioned for either left or right hand captures, and once a handed-ness is chosen it should not be changed without re-measuring the user’s hand. The user is given the option to try multiple times to emulate the image provided, or even redo the entire process after it has completed (if they are unsatisfied with their results in later steps), until they are comfortable with the captures they have made.**

*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, we do not currently use any traits captured from the fist gesture, though they would be helpful for follow on gesture sets or for creating masks of the users hand, while it provides a decent measurement of the size of the user’s palm. However, from the capture of the open palm we determine the center of the palm and then measure the angle from the tip of each finger this point. These measurements are the basis of how we distinguish the gesture set provided, though we explored many others. To provide a more generalized tool, we also measure many other traits of the users hand, explained below.**

# Detection of Statistics

*Contours*

**The first step in acquiring data and statistics on hands in our project was to accurately capture the contours that surround the blobs of the thresholded image. Once the largest contour was acquired with cv::findContours, we used it to calculate the first round of attributes of the users hand, providing: moments (and center of mass), minimum and upright bounding rectangles, a convex hull, and convexity defects. We experimented with many variations of these traits, but quickly found that a subset of the convexity defects (selection based off depth from the hull) was the most useful initial classifier. While we initially thought the aspect ratios of the bounding boxes would also provide useful information, they only confused otherwise dissimilar gestures.**

*Shapes*

**In the exploration phases of our project, we spent a large amount of time and resources exploring different attributes of hand contours and their bounding shapes, like those listed above. First we explored the bounding rectangles, but landed on two different variations of bounding circles for useful differentiation. Using the most concave points of the convexity defects we were able to calculate both the minimum enclosing circle, as well as a circle whose center and radius were both a result of averaging the defects distances. We then smoothed the results of these circles by keeping a cache of past radii and centers, averaging them in a weighted fashion based on recent occurrence. This process added stability to otherwise rapidly changing defect results. The benefit of having these circles was such that we could have a good estimation of the size of the palm, but also a marker for the region we needed to remove in order to isolate fingers from it.**

*Fingers*

**Finding the averaged and enclosing circles described above provided a useful tool for isolating the individual fingers from the hand contours, by deleting any area contained in the circle. [Figure 3] For another measure of stability, if less than 5 fingers were found, the number of fingers found was compared to running the algorithm once more with a circle of larger size. To completely eliminate the wrist, anything below the top three quarters of the circle was also deleted (obviously this would change depending on expected arm orientation). Through this fashion, we were able to successfully isolate the hand from the arm/wrist, but also determine with a certain measure of stability a reliable count of individual fingers present in the image. Fingers that are joined without a gap were not considered in this gesture set, though follow on work could distinguish them further based off of the contours garnered for each finger in the measurement step. These fingers were then analyzed for their angle from the center point of the palm, their general length to palm radius ratio, and their overall contour.**

**Figure 3:**

**Palm Elimination**

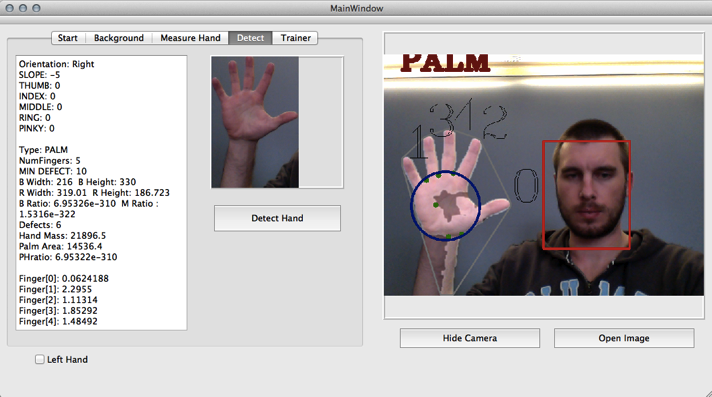


*Other Useful Statistics*

**For the implementation of a more general data set we have provided a pane [Figure 4] that shows the results of our explorations explained above, and many of the other attributes we uncovered along the way. This we hope will help anyone wanting to examine possible gestures for the inclusion in a library, allowing them to find easily determinable discrepancies for training in the final phases of the program.**

**Figure 4:**

**Attributes Pane**

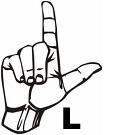


# Detection of Gestures

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

|  |
| --- |
| **T** |
| **I** |
| **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,* differentiating based on the presentation of fingers or not. 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 as within the *PALM* class if there are significant figures detected outside the minimum enclosing circle of the detected hand after the palm and wrist are eliminated. Both the number of fingers detected and a comparison between the differences of the angles of the detected fingers calculated from the image captured in the “Measure Hand” tab is used to determine which gesture the user is attempting.**

# Training

**The “Trainer” tab is where the user participates in the training process. The images of the gestures the training tool aims to teach (the same as displayed above in “Detection of Gestures”) are displayed for the user to emulate. To correctly perform a gesture, the user must hold their pose (or one still accepted as that same gesture) for a specific set of frames, that can be adjusted based on the purpose of the system. The program is satisfied that a user is displaying the correct gesture if after this number of captures, the program resolves that the user has continuously expressed it. 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 the same gesture again, and restarts the count of frames.**

# 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 accomplish all the lofty goals of our proposal (which we were rightly told to limit).**

**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 by providing the user time to shift his gesture and learn through repetition. We have developed primitive hand masks that emulate the user’s specific hand in different positions, and in the future hope to improve these in order to provide active feedback during the training phase of the program.**

# Code

**Our code has been omitted in this printed form, though this document and all the code attached to the project can be viewed on GitHub at:**

[**http://github.com/on2valhalla/GestureTrainer**](http://github.com/on2valhalla/GestureTrainer)