An Automated Method for Evaluating the Accuracy of ASL Static Gestures

Veronica Y. Flamenco
Department of Engineering and Technology
Western Carolina University
Cullowhee, NC 28723, USA

Paul M. Yanik
Department of Engineering and Technology
Western Carolina University
Cullowhee, NC 28723, USA

Robert D. Adams
Department of Engineering and Technology
Western Carolina University
Cullowhee, NC 28723, USA

Martin L. Tanaka
Department of Engineering and Technology
Western Carolina University
Cullowhee, NC 28723, USA

Abstract— **Within the past few years, research involving gesture recognition has flourished and has led to new and improved methods assisting people who communicate with sign language. Although numerous approaches have been developed for recognizing gestures, very little attention has been focused on correcting the placement of the fingers after the gesture has been performed. In ASL the placement of the fingers is very important considering a slight misplacement conveys a completely different word, letter, or meaning. We present a new method in correcting the placement of static American Sign Language (ASL) gestures using existing algorithms and feature recognition techniques.**

***Keywords*—ASL, edge detection, correlation, static gestures, gesture recognition**

I. Introduction

When researching new computer program approaches to help teach American Sign Language (ASL), numerous methods have been successfully developed [1-7]. Most of these methods involve displaying correct ASL gestures. However, little work has been done with ASL training for correcting the placement of individual fingers. Considering that corrective instructions are an important part of ASL training, researchers are beginning to find methods to help improve this with gesture recognition [4,5,8]. Although it is easy to mimic gestures, it may be difficult for the learner to know whether or not they are signing them correctly. This research involves developing a computer program that will assist in teaching the correct placement of the fingers when doing American Sign Language. Considering sign language has a wide range of gestures, focus will be mainly be on static gestures which include a few letters of the alphabet. By using image capture, ASL hand gestures made by the user will be compared to standard images in a database. The program will provide feedback concerning how close the user is to the reference gesture as well as specific instructions concerning how to correct the fingers' placement. The importance of this research is that most gestures, if made slightly incorrectly, can have a different meaning [9]. This would be the first step towards making a training/teaching program to help teach sign language accurately and precisely without the need of face-to-face instruction. Future studies could lead to more accurate training techniques for a wider range of ASL gestures.

II. Method

The research was begun by making a database of the alphabetical sign language gestures. Images were retrieved online from the database at lifeprint.com [10]. The database was necessary in order for the users to have a visual representation of the letters that they would be signing. It also provided a reference image in order to compare certain information to the user's image. Some of the information that would be needed from the database images are the position of the midpoint on each fingertip, as well as the angle direction and distance between two fingers, and the angles formed between three fingers (Finger 1=pinky,…,Finger 5= thumb). This information was used to compare the database and user gestures. Testing was performed on twelve letters (a, b, e, h, i, L, n, r, t, u, w, and y) selected from the database. These specific letters were chosen because all fingertips were visible in the images.

Considering that one of the objectives for the program was to recognize fingertip colors automatically when taking snapshot images, a set color range was going to be needed. In order to help distinguish one finger from the other, we decided to make the user (Veronica Flamenco) wear colored latex balloons on the tips of the fingers. We took images of unshaded and shaded gestures performed by the user in order to gather a set of color ranges to be used to detect the five colors under different lighting conditions. Unshaded images of these letters were taken in a well-lit room with natural and artificial lighting. Shaded images of these letters were taken in a room with moderate lighting using a poster to block most of the light hitting the hand. All twelve letters were performed as close to the reference image as possible. A program was developed to gather color ranges from the unshaded and shaded images. This allowed for us to click on each of the colored fingertips (three times per finger) of the unshaded and shaded images of the user. These clicks provided a minimum and maximum value of red, green, and blue (RGB) values (as well as x-y positions). This program also stored these six values in a color range matrix for later use. Another program used the

RGB values stored in the matrix to color the fingertips black wherever it found colors that fell within the selected RGB ranges. This was a visual representation to show if each fingertip color was being recognized. This program also found the midpoint of each fingertip based on the average of all rows and columns of the black pixels found.

Image enhancement was added to the images to help find a more defined range for the RGB values. Three different enhancement methods were tested on the original image: decorrelation, image color scale adjustment, and decorrelation with linear contrast stretching. The way the decorrelation methods work is that the stretching of the color bands "enhances the color separation of an image with significant band to band correlation" [11] which enhances the difference between colors in a digital image. The image color scale adjustment method provided contrast enhancement.

Figure 1 shows an example of how the three methods mentioned compare to the original (unshaded) image of the letter 'a'. Visually, looking at the enhanced images, the two decorrelation methods were seen to provide more defined fingertip colors. After further examination of the data, the RGB values provided by the decorrelation with linear contrast stretching gave smaller ranges for each individual color. This means that when viewing the enhanced images, this method provided exaggerated color which improved visual interpretation making the features on the image easier to distinguish. This shows that the five fingertip colors were not overlapping as much for this method, which means each color was less likely to be mistaken with one of the other colors.

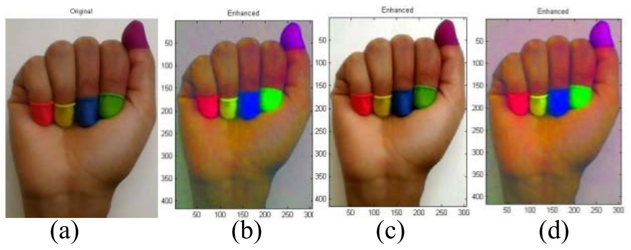

Figure 1: Original image of 'a' compared to the three methods used: (a) Original image (b) Decorrelation (c) Image color scale adjustment (d) Decorrelation with linear contrast

By examining the data of the RGB values we noticed 6 different cases of overlap between both images (shaded and unshaded), which depended on the enhancement method chosen. From these cases we would then get a set range of RGB values for every color on each fingertip. Comparisons of these six cases can be seen in Figures 2-7. For each of these figures range(1) is the minimum intensity for a given finger in the unshaded image, range(2) is the corresponding maximum intensity, range(3) is the minimum intensity for a given finger in shaded image, and range(4) is the corresponding maximum intensity. Case 1 shows that if range(4) ≤ range(1), then it will have two min and max set values which create two non-overlapping ranges as seen in Figure 2. Case 2 shows that if range(2) ≤ range(3) then it will also have two min and max set values which create two non-overlapping ranges as seen in Figure 3. Case 3 shows that if range(3) ≥ range(1) and range(4) ≤ range(2) are both true then if range(4) – range(3) ≤ $\frac{1}{4}$(range(2) - range(1)) the min and max will be found using (1) shown in Figure 4. If range(4) – range(3) > $\frac{1}{4}$(range(2) - range(1)) the min and max will be found using (2) shown in Figure 4. Case 4 shows that if range(1) ≥ range(3) and range(2) ≤ range(4) are both true then if range(2) – range(1) ≤ $\frac{1}{4}$(range(4) - range(3)) the min and max will be found using (1) shown in Figure 5. If range(2) – range(1) > $\frac{1}{4}$(range(4) - range(3)) the min and max will be found using (2) shown in Figure 5. Case 5 shows that if range(2) – range(3) ≤ $\frac{1}{2}$ (range(4) - range(3)) then the min will be found by doing $\frac{2}{3}$ ((range(1) - range(3)) + range(3) and the max will be found by doing $\frac{2}{3}$((range(4) - range(2)) + range(2), shown in Figure 6. Case 6 shows that if range(2) – range(3) > $\frac{1}{2}$ (range(4) - range(3)) then the min will be range(3) and the max will be range(2) shown in Figure 7. These set ranges will be used to find the fingertip colors automatically on the user's images, later on.

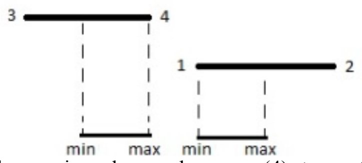

Figure 2: Case 1 shows min and max when range(4) ≤ range(1)

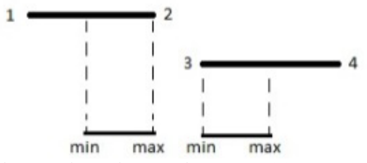

Figure 3: Case 2 shows min and max when range(2) ≤ range(3)

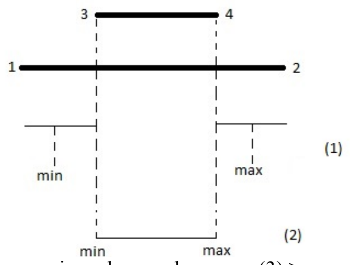

Figure 4: Case 3 shows min and max when range(3) ≥ range(1) and range(4) ≤ range(2)

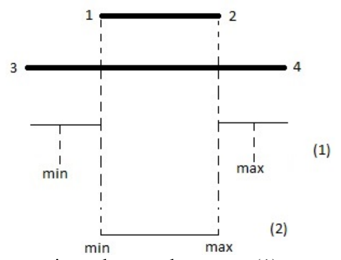

Figure 5: Case 4 shows min and max when range(1) ≥ range(3) and range(2) ≤ range(4)

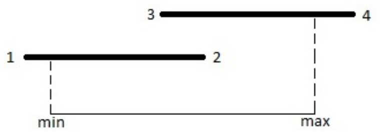

Figure 6: Case 5 shows the min and max when range(2) – range(3) ≤ $\frac{1}{2}$ (range(4) - range(3))

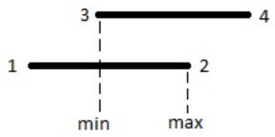

Figure 7: Case 6 shows the min and max when range(2) – range(3) > $\frac{1}{2}$ (range(4) - range(3))

After having found set ranges of RGB values, we made a program that would ask the user what letter they would like to sign. Then it asked the user what method of image enhancement he/she would like to apply to the saved user images (unshaded and shaded). The program re-colored the fingertips black by using the ranges to search for the pixels meeting the criteria of the given case. This provided a visual representation to show if each fingertip color was being recognized. It also found the midpoint based on the average of the row and column values of the re-colored pixels. An example of this can be seen in Figure 8. After finding the midpoint, this program then calculated the angle direction between two fingers and the angles formed between three fingers.

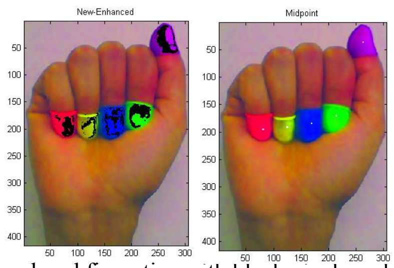

Figure 8: Re-colored fingertips with black pixels and midpoints (shown with white pixels) of the letter 'a'

The user's images need to be scaled to match the size of the database images prior to testing whether the gesture is correct or not. Given the distances and angles generated in the previous steps, we used this information of the user's image (whichever is being used) and compared it to the one in the database. Before comparing the distances of both the database and the user's images we had to find a scaling factor to use on the user's distance measurements. This way both images would be proportional in size. Considering angles were not dependent on the images being proportional, they were not scaled. A set scaling factor was found by having the user perform a "correct gesture." A new scaling factor will not be found or changed each time a different variation of the letter is signed. This is because the hand perimeter of the gestures will be off, such as a finger extended too far out, and will make the edge detection process skew the width for the scaling factor. User images were scaled to match the width of the database images. Table 1 shows the fifteen key angles that were compared to one another for all twelve letters. The x's represent the angles that resulted in an angle difference greater than twelve degrees (when performing correct gestures). From Table 1, we can see that each individual letter had about five or fewer angles that were greater than twelve degrees off (refer to the count at the bottom of Table 1). We can also see that three out of the twelve letters, a, b, and n were considered extremely close matches because they did not have angles greater than twelve degrees.

TABLE I. 15 KEY ANGLES THAT RESULTED IN GREATER THAN A 12° DIFFERENCE BETWEEN USER AND DATABASE IMAGES

Fingers forming angles			a	b	e	h	i	L	n	r	t	u	w	y
2	1	3				x				x	x			
3	1	4						x						
4	1	5						x					x	
1	2	5			x	x				x				x
5	2	4					x	x						
4	2	3					x	x			x			
2	3	1					x							x
1	3	5			x						x			x
5	3	4			x			x		x		x		
3	4	2					x			x				
2	4	1					x							x
1	4	5			x						x			
4	5	3												
3	5	2			x									
2	5	1											x	
Count (>12°)			0	0	5	2	5	5	0	4	4	1	2	4

After using one of our programs to find the differences of the user's "correct gestures" and database images, we now had reference data, in which we could use to compare with incorrect gesture images. To clarify, we did not compare the incorrect gestures to the information obtained from the database gestures. This is because we know that a zero degree difference is ideal; however, what we need is to get data of "correct" gestures performed in order to have reference data when comparing to "incorrect" gestures. Although all the images of the twelve letters were made as correct as possible, some angles were considerably off. This could have been caused by the location of the midpoint and where it was placed when the program was finding the RGB ranges on the fingertips. Considering the letter 'a' was a very close match, based on the information in Table 1, we decided we would use this letter as the preliminary testing image. Then, we took five different shots performing the letter 'a' incorrectly. These five different versions of the letter 'a' can be seen in Figure 9. After calculating all the angles for all five incorrect versions of the letter 'a', we then placed all these angles in a table along with the angles of the correct 'a' gesture. These results are summarized in Table 2. When comparing all five versions to the "correct" one, we decided that we were not going to look for angles that were close to zero, however we were looking to make sure all angles were fairly close to the difference in angles found of the 'Correct A.' It can be seen in Figure 9 that fingers one and five (Pinky and Thumb) were moved. The information in Table 2 shows that the angles that involve finger one and five have more deviation than other angles. Angles highlighted in red show angle deviations when finger five was an outer segment of the angle. Angles highlighted in green show angle deviations when finger one was an outer segment of the angle. This confirmed that the program was correctly detecting the angles which were incorrect.

III. Conclusion

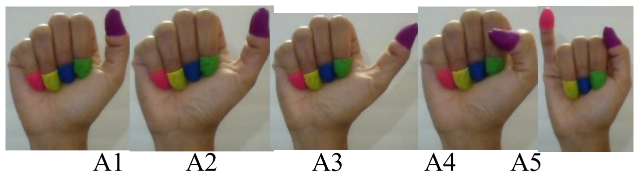

Figure 9: Versions of ASL gesture 'a' performed incorrectly

In this paper we used static images of ASL gestures in developing a program that would recognize the user's colored fingertips automatically. After creating a database of the twelve letters we would use (retrieved from [10]), we then took pictures of the user performing the letters in a shaded and a lighted environment. Then we enhanced the intensities of the images to get a better range of RGB colors. After retrieving all the RGB ranges of the enhanced shaded and unshaded images, we analyzed both images of the same letter to produce a set range. This was useful because it provided a color range that would be more accurately recognized in a normal environment when taking snapshots of images. We then computed the midpoint for each finger based on the pixels found using the set range. From these midpoints we computed distances and angles between fingers. We then compared the user's correctly made gestures to the database. Now we had set angles for all the letters in order to start comparing intentionally incorrectly made gestures to the "correct" gestures that were performed by the user. From the angles that were obtained, we could confirm that the program was indeed detecting the fingers that were misplaced correctly.

A method that will be looked into in the future will be to find the centroid of the midpoints of all five fingers. With this we can calculate the distances between each finger in relation to the centroid and have set distances when comparing the user's images to the images in the database. This will lead to using these distances and angles to determine whether or not the finger needs to be relocated by comparing them to database image features. If gestures are not correct, we will use this information to provide the user with feedback on how to correct finger position.

References

[1] K. Kadam, R. et al. "American Sign Language Interpreter," in IEEE Fourth International Conference on Technology for Education, 2012, pp. 157-159.

[2] Y. N. Khan and S. A. Mehdi, "Sign Language Recognition Using Sensor Gloves," in Proceedings of the 9th International Conference on Neural Information Processing ICONIP 2002, vol.5, pp. 2204-2206.

[3] T. Messer, "Static Hand Gesture Recognition," M.S. Thesis, Department of Informatics, Univ. of Fribourg, Fribourg, Switzerland, 2009.

[4] A. Samir and M. Aboul-Ela, "Error Detection and Correction Approach for Arabic Sign Language Recognition," in Seventh International Conference on Computer Engineering & Systems (ICCES), 2012, pp. 117-123.

[5] T.E. Starner and A. Pentland, "Real-Time American Sign Language Recognition from Video using Hidden Markov Models," International Symposium on Computer Vision, Coral Gables, Fl. pp. 265-270, 1995.

[6] N. Tanibata, N. Shimada and Y. Shirai, "Extraction of Hand Features for Recognition of Sign Language Words," Proc. Int',l Conf. Vision Interface, pp. 391-398, 2002.

[7] N. Adamo-Villani, J. Doublestein and Z. Martin, "The MathSigner: An Interactive Learning Tool for American Sign Language," in Eighth International Conference on Information Visualization Proceedings, 2004, pp. 713-716.

[8] A. Licsar and T. Sziranyi, "Dynamic Training of Hand Gesture Recognition System," in Proceedings of the 17th International Conference on Pattern Recognition, ICPR 2004, vol.4, pp. 971-974.

[9] C. T. Best et al. "Effects of Sign Language Experience on Categorical Perception of Dynamic ASL Pseudo-signs," The Psychonomic Society, vol.72, 2010, pp. 747-762.

[10] W. Vicars, "American Sign Language,"ASLU, 2012, Internet: http://lifeprint.com/

[11] MATLAB version 7.8.0. Natick, Massachusetts: The Math Works Inc., 2013,http://www.mathworks.com/help/images/adjusting-pixel-intensity-values.html#f11-23223

TABLE II. DIFFERENCE OF ANGLES BETWEEN THE FINGERS OF 5 DIFFERENT VERSIONS OF 'A' COMPARED TO THE 'CORRECT A'

Fingers Forming Angles			Difference of angles between the fingers (degrees)					
Finger 1	Finger 2	Finger 3	"Correct A"	A1	A2	A3	A4	A5
2	1	3	8.2	7.6	9.0	7.2	9.0	0.7
3	1	4	1.8	2.2	2.3	2.9	0.7	7.4
4	1	5	7.2	6.5	15.5	21.9	11.5	2.6
1	2	5	1.7	1.4	10.9	21.6	0.3	72.9
5	2	4	12.3	7.8	19.2	26.0	11.7	5.3
4	2	3	1.6	3.1	3.9	5.0	1.7	4.8
2	3	1	7.4	6.5	7.7	7.7	5.9	63.5
1	3	5	9.4	12.4	26.6	37.1	9.3	76.9
5	3	4	13.6	4.8	18.8	27.8	9.6	4.9
3	4	2	1.6	2.1	2.8	3.3	1.7	3.7
2	4	1	4.1	1.0	1.6	1.2	2.4	60.9
1	4	5	13.8	8.1	29.3	43.0	1.9	75.2
4	5	3	2.7	2.2	4.9	8.8	8.1	5.7
3	5	2	2.9	3.4	6.7	5.6	0.6	3.4
2	5	1	1.1	0.3	2.2	4.0	2.1	63.6