Assignment 4

Aleyna Alper, 21827024
Department of Computer Engineering
Hacettepe University
Ankara, Turkey
b21827024@cs.hacettepe.edu.tr

January 7, 2023

1 Introduction

In this assignment, we will use cross correlation to define American sign languages images to a set of template images and classify the ASL images based on the highest degree of correlation. Cross correlation is a measure of similarity between two images. It is used to find the degree of correlation between two images or to determine the degree to which one image is similar to a template image.

2 Experiment

2.1 About the code

First, I added the representation of the methods in my code to the table below:

Table 1: Methods

Method Name	Input(s)	Output(s)	Info
ncc	target, template	ncc result	calculate two images' ncc val
classification	letter, target image, pixel	-	make classification
binaryClassification	letter, target image, pixel	-	make binary classification
offset	targetName, target image	-	find offset images and display

In the first stage of the assignment, I read all the target photos given to us one by one. Then I converted every photo I read into black and white. After gray scaled, I did the hog process requested from us.

I checked the names of all the photos in the target file before doing the action. Their names were going as a string of letters + a number. This series

of numbers started from 1 and continued as 101,201,301 and continued until 29001. For this, I first defined a list with all these elements and used this list in the reading phase. I did the reading in a for loop and used the list I mentioned above while doing this. I sent each photo to the classification function in order to understand which photos it is suitable with.

Classifation function takes two different parameters. These parameters are a photo and the letter element, which specifies which letter corresponds to the actual value of this photo. It is aimed to make comparisons with all template photo values by taking these values into a loop first. I used the NCC function requested from us to perform the operation. Actually, at first, I was cropping because the size of the template was 200x200 and the size of the target photos was smaller than that each time, but I couldn't get a healthy result in this way and my accuracy value was very low. I got an average of 6-7 percent results and then I realized that I had to use another method. According to this formula, it is explained that we need to find the area where it is the maximum. The algorithm that appears in the pdf is as follows:

$$[\hat{u}, \hat{v}] = argmax_{u,v}\rho_{12}(u, v)$$

In the other method, I realized that I had to scan the area piece by piece as mentioned in the pdf. In this part, I applied a 200x200 template size navigating process. While applying the process, I did not touch the target size and only did a pixel-based scrolling. While doing this process, I first performed it by skipping 3 pixels. Afterwards, I did this by skipping 1 pixel to increase the accuracy value I obtained a little more. Printing the Accuracy value took much longer than 3 pixels, but after running the code, I added it directly to the jupyter notebook with the result.

After scanning and finding all the parts separately, I threw each one into the Normalized Cross Correlation function that I used in my previous assignments. I recorded the highest Correlation value. I did this for all the letters and put the value of each one in the template list I collected. At the end of the transaction, I received the highest correlation value in return. In this way, I made a recognition process by passing a total of 780 data through the classification function.

2.2 Comparing Data

First, when I ran the code, I printed the output of all the photos as the desired result and the result after the algorithm. An example of this output is as follows:

```
Desired val: C Result: C
Desired val: C Result: A
Desired val: C Result: C
Desired val: C Result: C
Desired val: C Result: C
Desired val: C Result: C
Desired val: C Result: L
Desired val: C Result: C
Desired val: C Result: C
Desired val: C Result: C
Desired val: C Result: Q
Desired val: C Result: C
Desired val: C Result: X
Desired val: C Result: Z
Desired val: C Result: E
Desired val: C Result: G
Desired val: C Result: C
```

When I examined the outputs seen above, I saw that a very large part of the letter C was correctly detected by the algorithm I used. I thought it was because the letter C was understood much more prominently in American Sign Language than other letters. When I looked at other letters, I realized that some of them were very difficult to recognize, unlike the letter C. The output of some of the letter E photos created by my algorithm is as follows:

```
Desired val: E Result: N
Desired val: E Result: E
Desired val: E Result: Z
Desired val: E Result: Z
Desired val: E Result: S
Desired val: E Result: Z
Desired val: E Result: N
Desired val: E Result: N
Desired val: E Result: G
Desired val: E Result: M
```

As seen in the output above, most of the results were wrong because the letter E is more difficult to recognize. This is because there are a lot of American Sign Language representations that look like the letter E.

```
print("Desired val: "+targetName[0]+" Result val: "+string.ascii_uppercase[peak_index])
```

I was running this part with the code I added above, but since it was not requested in this assignment, I deleted the piece of code.

2.3 Accuracy

In this section, I will talk about the accuracy I have achieved. First of all, I defined two different variables, trueResults and allResults, in my code to find the accuracy value. To find the Accuracy value, I divided the trueResults value by the allResults value and multiplied by 100 to determine the accuracy value as a percentage.

Finding the Accuracy value part, I classified all the data according to their ncc values. In this process, we wanted to go over the target value piece by piece and compare them by finding the most suitable results accordingly. First, I added an extra parameter to my classification method to determine how many pixels I would like to navigate. Thanks to this parameter, I have provided the opportunity to show how many pixel intervals I want to navigate while running the function.

2.3.1 Output 1

While doing this process, I first performed this process by walking around 3 pixels by 3 pixels. This code I created is the output of the first line in Jupyter notebook. When I circulated with three pixels, the process took an average of 20-30 minutes. The accuracy value that comes out with the code I run with three pixels is as follows:

```
Accuracy is: %13.461538461538462in 3 pixel wandering
```

2.3.2 Output 2

Secondly, I wanted to circulate this process as one pixel instead of three pixels. I aimed to perform this process for better accuracy, but this process took too long unlike the first example. There was a cooldown of over an hour, but the resulting accuracy seems to be much higher when you look at the photo below.

```
Accuracy is: %15.897435897435896in 1 pixel wandering
```

2.3.3 Output 3

In this part, unlike the other two outputs, there is an extra situation. Normally, we wanted to convert the photos to binary values before doing the HOG process in PDF. But since I found a lower result than usual every time I did this operation, I wanted to examine this situation under a different heading. In the third line of Jupyter Notebook, I created a binary classification method, apart from the classification method I used above. Although this method looks the same as the other method, it is aimed to convert the extra photos into binary images after the gray scale process and then calculate the HOG. All the steps after this process are the same as the method under the classification name, and this is how I handled the desired situation.
When converting to binary operation, in case of specifying threshold, I had to set a threshold between 0 and 255. I found the most suitable one of these values as 50 by trying all of them first. As a result of all operations, the accuracy value obtained by using the binary conversion process is as follows:

```
Accuracy is: %7.435897435897436in 3 pixel wandering with converting binary
```

As can be seen in the output value in the photo, the probability of making the correct classification has decreased and the resulting accuracy value has decreased a lot.

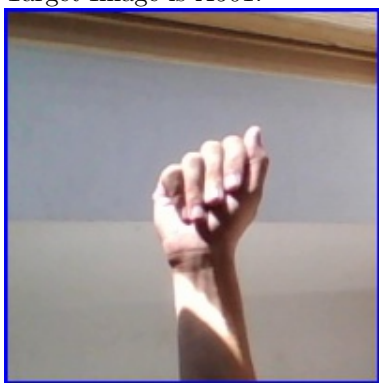
3 Results

3.1 True Results

In this section, I will talk about the correct results I have achieved while making classification.

3.1.1 Example 1

Target Image is A501:



Defined Image is A:

Offset Representation of A501:

As seen in this example 1, classification methods have been applied to a target

instance of the letter a and the correct result has been achieved. In the third photo, the part with the highest NCC found when browsing part by part is yellow frame as desired.

3.1.2 Example 2

Target Image is C401:

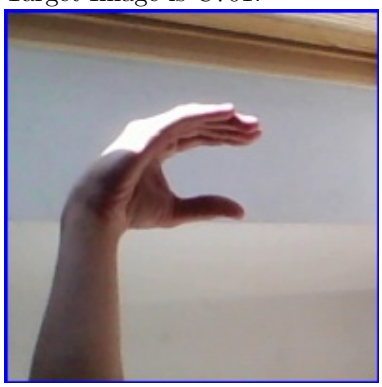
Defined Image is C:

Offset Representation of C401:

When I looked at the results of the algorithm I created, I discovered that the classification results of the letter c were much more than the other letters. This is due to the fact that the letter c in the American sign language is very distinctly different from the others.

3.1.3 Example 3

Target Image is C701:



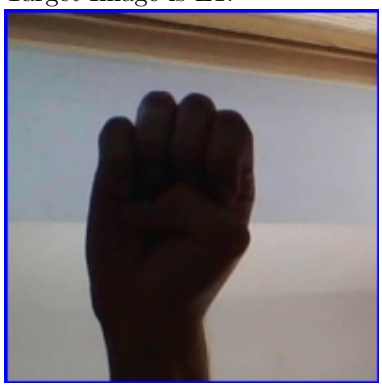
Defined Image is C:

Offset Representation of C701:

In order to show that you can recognize the letter c very easily in the algorithm I created, I defined another example from the letter c. Classification methods were applied to a target sample of the letter C and the correct result was achieved. The part with the highest NCC found while navigating piece by piece in the third photo is framed in yellow, as desired.

3.1.4 Example 4

Target Image is E1:



Defined Image is E:

Offset Representation of E1:

As seen in the fourth example, although the target photo is dark, it was recognized by the algorithm I created and it was able to find the letter e with ncc. In this example, it is possible to identify that the shadows are unimportant in classification, just by looking at the figure.

3.1.5 Example 5

Target Image is G2601:

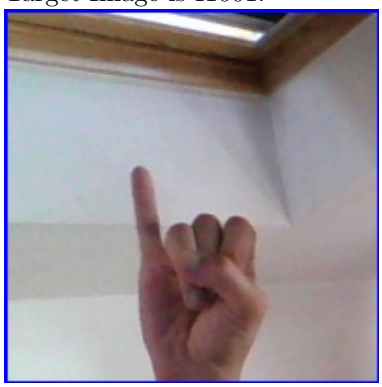
Defined Image is G:

Offset Representation of G2601:

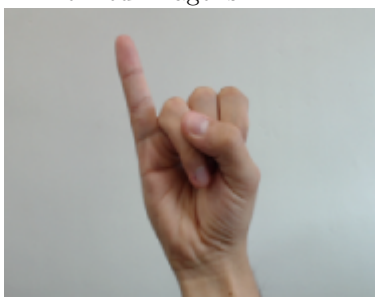
The same situation is here, as shown in the fourth example. Although the target photo is dark, the algorithm was able to find the result correctly. In fact, this is because we prevented the color transitions from affecting the result by performing the hog operation.

3.1.6 Example 6

Target Image is I1001:



Defined Image is I:



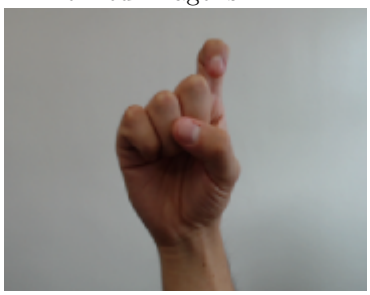
Offset Representation of I1001:

As seen in this example, it is understood that the similarity of target and template photos is very high. For this reason, the algorithm I created has very easily defined which letter this sample belongs to and gave the correct result.

3.1.7 Example 7

Target Image is X101:

Defined Image is X:



Offset Representation of X101:

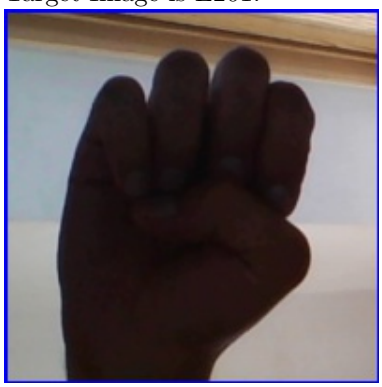
As seen in this example 7, classification methods have been applied to a target instance of the letter X and the correct result has been achieved. In the third photo, the part with the highest NCC found when browsing part by part is yellow frame as desired.

3.2 False Results

In this part, I will take some of the wrong results from the algorithm I created and interpret the reasons why they might be wrong. I will show you which results with wrong letters

3.2.1 Example 1

Target Image is E201:



Defined Image is M:

Offset Representation of E201:

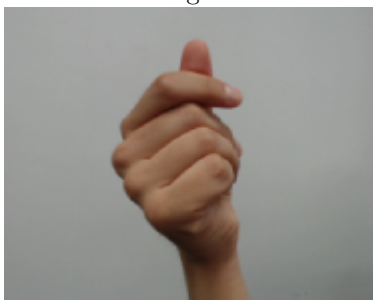
As seen in the first wrong example, the target image was the letter E, while the letter defined after the classification process became the letter M. The reason

for this is that both letters are similar and the details of the target image are not clear due to the dark color of the photo.

3.2.2 Example 2

Target Image is X301:

Defined Image is T:



Offset Representation of X301:

In the second wrong example, the reason why the classification result is wrong is due to the fact that the details of the letter T and the letter X are similar when viewed from the outside.

3.2.3 Example 3

Target Image is S201:

Defined Image is A:

Offset Representation of S201:

In the third wrong example, while the desired value was the letter S, the algorithm resulted in the letter A. This is because the representation of the two letters in American Sign Language is very similar. Naturally, the algorithm could not recognize these two letter representations, which are easy to confuse even under normal life conditions.

4 Conclusion

As a result, in this assignment, we performed the classification of thirty data from each letter using the methods requested from us. I understood the logic of all the methods I used while doing this process, and I gained the opportunity to adapt the image classification process, which is an important topic in image processing, in real life. Thanks to this assignment, which is the last project of the BBM 415 course, I added a new perspective to my problem solving ability.

5 References

https://www.nidcd.nih.gov/health/american-sign-language