A2 Project Report

CS 585 A2

Abdelazim Lokma

Team member: Risheet Nair

02/14/24

Problem Definition:

The goal of our project was to successfully recognize multiple Americal Sign Language gestures, using the techniques learned in our CS 585 Image & Video Computing course. We decided to try to recognize the following letters - B.U and C.V:

The ASL sign for B

The ASL sign for U

The ASL sign for C

The ASL sign for V

The applications of this code could be used to aid in the creation of an 'ASL translator' program that translates ASL speaker's signs into english audio. Allowing users with impaired hearing to communicate with others during online meetings without having to type.

Due to the simple nature of some the algorithms used in our project (such as the skin detection algorithm), the project assumes that users will be seated in good lighting conditions, with a dark background (to allow for easy skin detection), and that the user's hands remain at a consistent and reasonable distance away from the screen.

Immediately just by looking at the different hand signs shown above, Risheet and I were able to recognize the difficulty of correctly detecting the B and U hand signs, due to their similarity. However, rather than selecting different hand signs that were more distinct from one another, we decided to implement multiple Computer Vision techniques in order to overcome these challenges.

Method & Implementation:

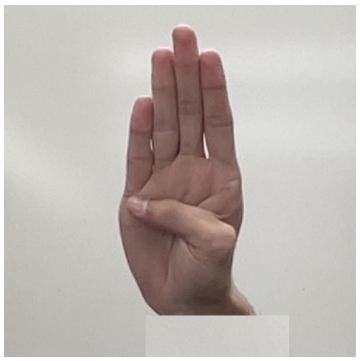
Our program utilizes multiple computer vision methods in order to correctly detect the user's hand sign. The main aspect of our program relies on template matching in order to correctly classify the user's hand signal. After which we also extract a

contour of the user's hand, in order to measure the circularity of the hand sign, which aids in narrowing down the possible letters the user could be signing.

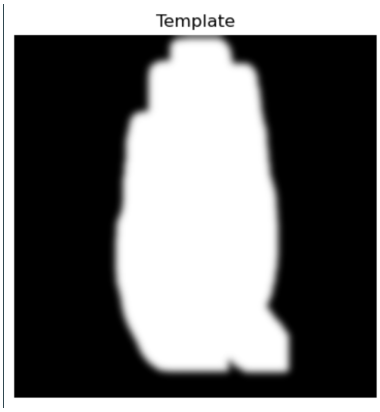
Template Matching Algorithm:

The base of our hand recognition program relies on template matching in order to classify the user's hand signals. This required us to store templates of Risheet's hand, signing out the different letters with a clear white background. For each frame read from the video camera, we attempt to match all four templates (B, U, C & V) to the frame, creating a similarity map for each one, after which, we scan through the similarity map in order to find the highest similarity score, allowing us to determine which template was most similar to the frame in question.

An example of the original template for the B hand sign.



The altered template, a cropped section of the background to hide the wrist and arm.



Final template for B, after erosion, dilation and blurring.

However, one issue we noticed is that the presence of Risheet's exposed wrist had a noticeable impact on the circularity calculations that would be done later. We decided to edit the templates so that the wrist was no longer exposed, allowing the template matching and circularity algorithms to only focus on the part of the hand that matters. This meant that it was imperative for the user's wrists to also be concealed during testing, as the circularity measurements could be affected.

These templates were then fed through our skin detection algorithm, which outputted a binary grayscale image, pixels representing skin were displayed as

white, and all else as black. Erosion was used to eliminate some of the jagged edges on the template, as well as any background noise that was misidentified as skin, and Dilation was used to fill in any hollow parts of the hand. Finally, a gaussian blur was applied to the templates in order to allow the template matching algorithm to better generalize, making it less sensitive to changes in distance and orientation.

However, at this stage of development, our template matching algorithm was not able to reliably distinguish between the different hand signs, often confusing B for U, and V for U due to their similarities. As a result, we had to supplement our template matching with other methods, that could help identify the distinguishing features of each hand sign.

Circularity Measurement Algorithm:

We decided that we could distinguish between the similar hand signs through measuring their circularity. This measurement could help resolve the conflicts that our template matching algorithm faced. For example, since our algorithm often confused U for B, we could use circularity to rule out one of the two options, as the hand sign for B was more circular than the hand sign for U, the same approach is applied to resolve the conflict that would occur between U and Y.

To measure circularity, we calculated the contour of the hand, and used that to find the area and perimeter of the hand shape, these could then be used to solve this equation:

$$\text{circularity} = \frac{4\pi \times \text{area}}{\text{perimeter}^2}$$

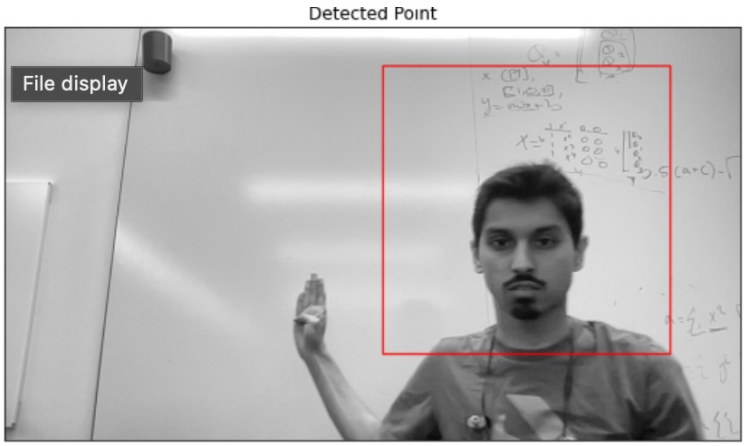
Below is a list of the functions that were implemented:

- `create_template(template_list, grayscaled=True)`: Accepts a list of image filenames representing hand shapes or gestures. Optionally converts them to grayscale and applies Gaussian blurring. Returns a list of preprocessed templates.
- `create_contoured_templates(template_list)`: Takes a list of thresholded templates and extracts contours from each template. Returns a list of contours.
- `get_contour(frame)`: Extracts the contour of the current frame using OpenCV's contour finding functions.

- `circulariy_threshold(circularity, template_list)`: Determines which templates to consider based on predefined circularity thresholds.
- `my_template_matcher(frame, template_list, template_names)`: Matches templates with the current frame using template matching algorithms and visualizes the result. Returns the detected hand sign.
- `calculate_circularity(contour)`: Computes the circularity of a contour using its area and perimeter.
- `determine_hand_sign(frame, template_list)`: Determines the detected hand sign by applying contour comparison and circularity thresholding techniques.

Experiments:

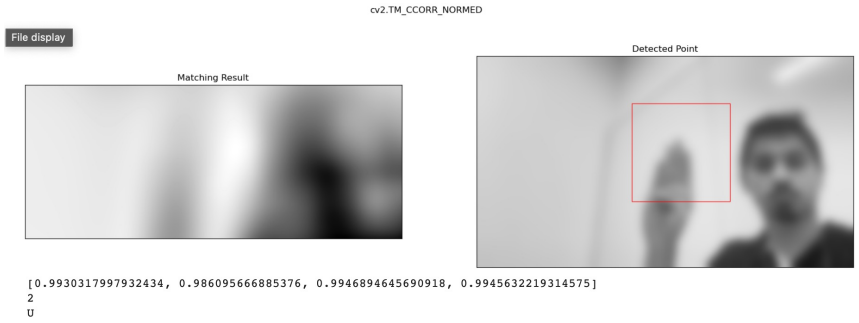
As stated before, we started by attempting to use template matching as our only means of identifying hand signals. Our main metric for these experiments was the success rate when detecting the hand signs. The first experiment was a failure, as our template matching algorithm would match to our face when our hands were too far away. When our hands were at the correct distance away from the screen, the slight differences in lighting and orientation would result in an incorrect match. This test was run 5 times and the result was the same.



This example shows the case where template matching would match to our face rather than recognizing the hand sign.

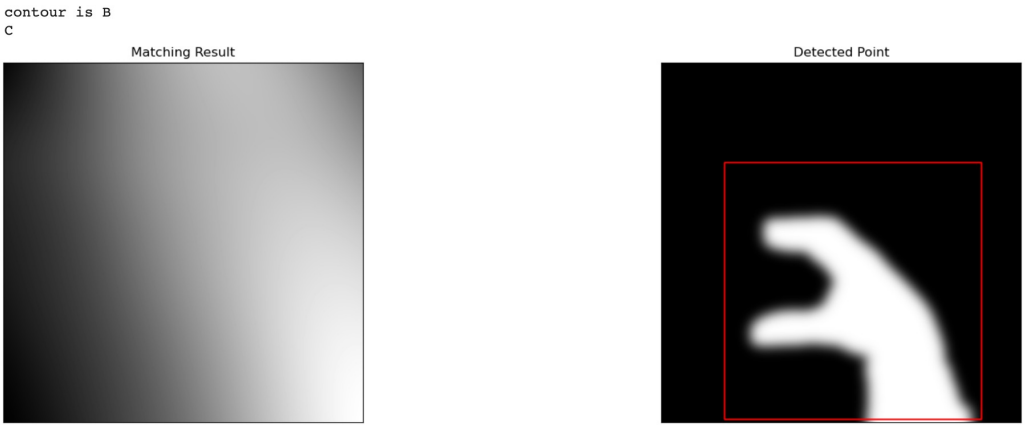
Our second experiment involved implementing thresholding to filter out the user's skin from the rest of the frame. We used the skin detection algorithm to convert both the frame and the templates into a binary greyscale image. This aided the template matching algorithm to become more resilient, allowing it to accurately match a template to the user's hand, rather than their face. A parameter value that we had to fine tune during this experiment was the size of the blurring kernel, which impacted how much blur we would see. We began with kernel size of (151,151) and scaled up and down in order to determine whether more or less blurring would improve results. We discovered that the program performs well with a kernel size of (75, 75).

We tested this case 5 times, and the template matcher succeeded 2 out of 5 times. However, the template matching algorithm still predicted the wrong hand sign, as seen below:



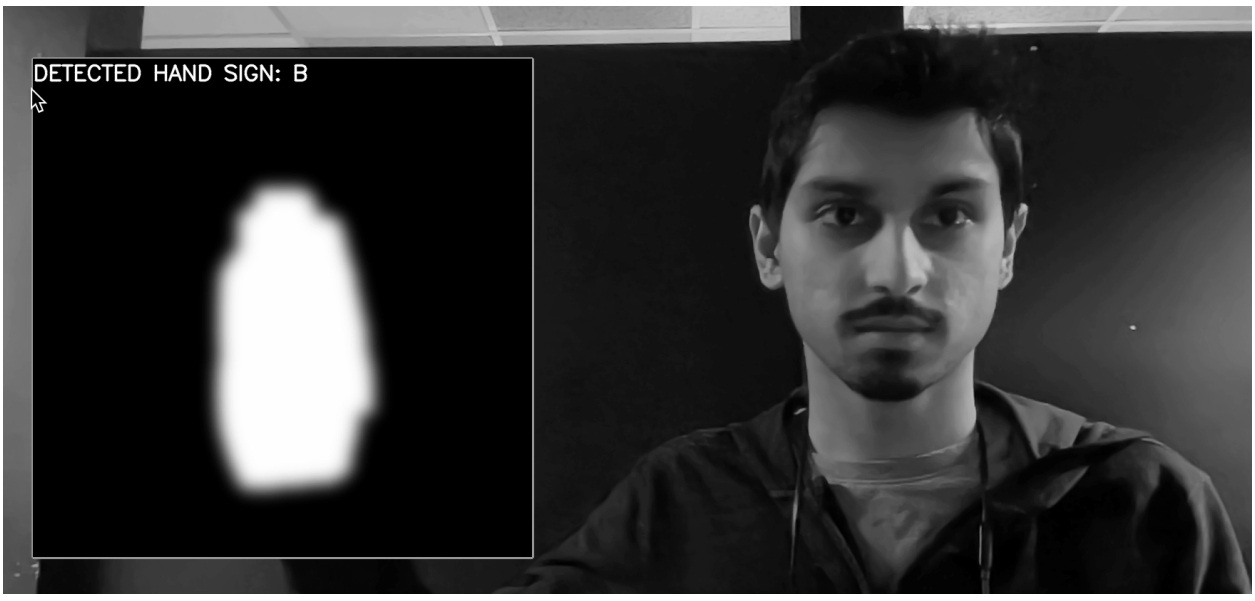
In this experiment, the user is signing B, but the template matching algorithm matched U instead of B.

We decided to try matching contours rather than relying solely on matching templates, but this proved to be even less accurate than matching templates, as shown below. Our contour matching algorithm would generate a contour for all templates, as well as frame being analyzed, and would compare them. We attempted this test around 5 times, and the contour matcher failed to predict the hand sign all 5 times.



In this experiment, the contour matching algorithm incorrectly predicted B, while the actual hand sign was C.

Our final experiment involved using circularity measurements of the user's hand to rule out certain templates from being considered by the template matcher. This allowed us to resolve the common confusion that our template matcher encountered when attempting to differentiate between B and U, or between U and V. Since the templates used Risheet's hands, we decided it would also be usefull to swap users mid test, to see if the templates of Risheet's hands would match to mine, which they did perfectly. Below are images showing the experiment being conducted by both team members.

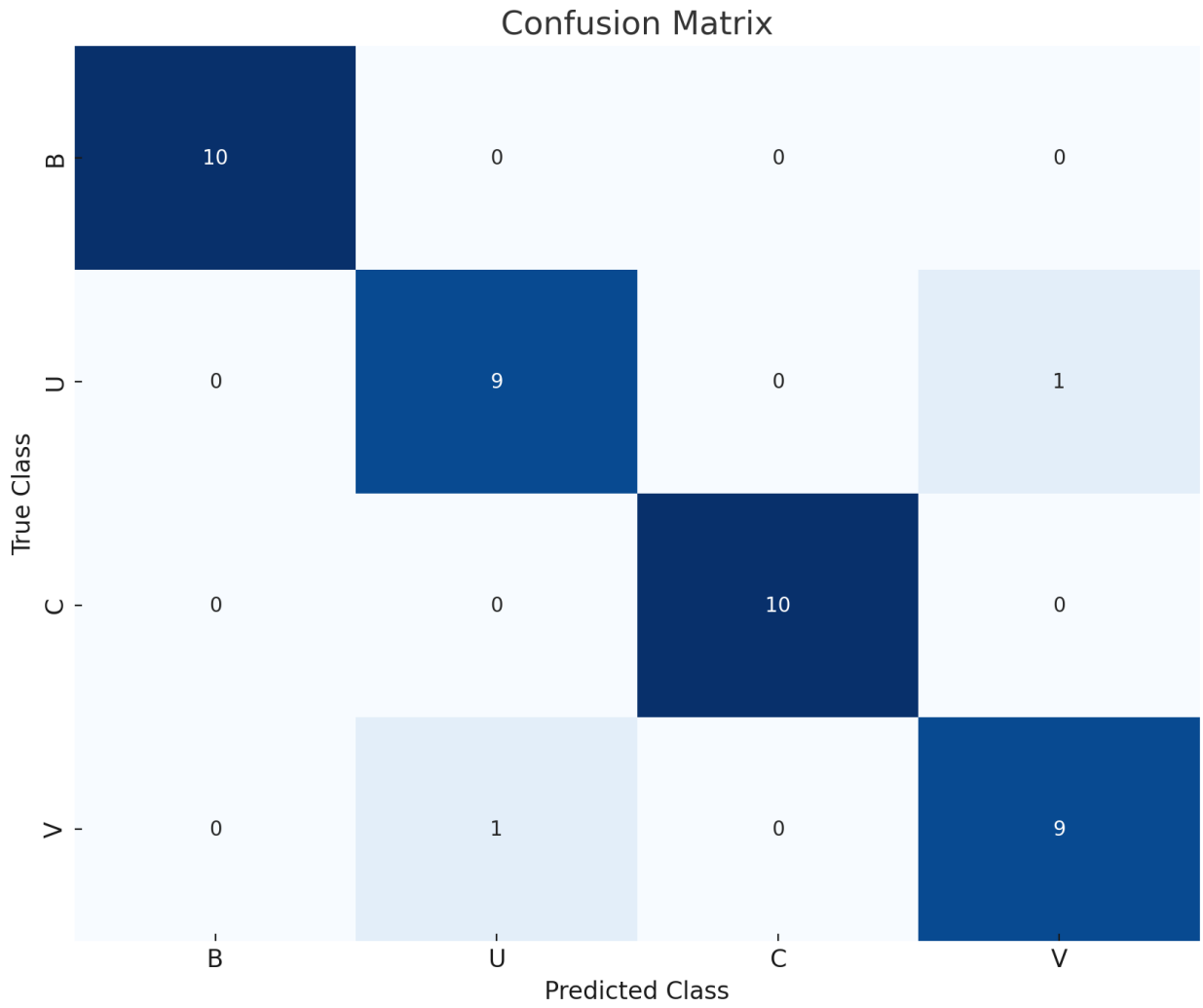


Risheet started the experiment since the templates were taken of his hands.

I switched with him later on to see if the templates could match on my hands as well, which worked.

The results of the final experiment are discussed in the next section.

Final Results:



The final experiment yielded extremely positive results. With a near perfect success rate for almost all classes, with the notable exception being V and U, which were mis-predicted once.

A link to the test video can be found here:

https://drive.google.com/file/d/1aDDEQleFXGTCJWQW5xRx9ljNWpiz-wQS/view?usp=sharing

Discussion & Conclusion:

Strengths	Weaknesses
Works across multiple users regardless which user generated the templates	Moving too far away from the camera or rotating too much results in incorrect predictions
Can detect hand signals with very high accuracy	Not wearing sleeves would greatly impact the accuracy of the circularity measurements, and therefore the accuracy of the final predictions.
	Background must be a dark, uniform background for the program to work.

Our results do show that our program is generally successful, so long as the experiments are conducted in controlled environment with specific factors (such as a dark background and covered sleeves), we consider this to be a satisfactory result, given the simplicity of the skin detection algorithm that was used for the project.

This project could be improved by improving its core components. Updating the skin detection algorithm to be more advanced could help with skin detection in a wide variety of environments and lighting conditions. More templates can be loaded into the program to test its scalability.

During the course of this project, when we encountered the obstacles that made template matching B.U C.V difficult, we were tempted to switch to easier hand signs, that were more distinguishable from one another, however, we stuck to our original goal and managed to get a very good final result. Our message is to never give up or concede your goals, perseverance is the key to success 💙.

Citations:

Circularity calculation: https://stackoverflow.com/questions/74580811/circularity-calculation-with-perimeter-area-of-a-simple-circle

Inspiration video for our code implementation: https://www.youtube.com/watch?v=v-XcmsYIzjA&t=157s

Our skin detection algorithm was taken from lab notes.