



Real Time Sign Language Detection using Image Processing

Background:

- 5% of the total world population face the problem of hearing loss, that accounts for around 466 million people, out of which 34 million are children.
- Traditionally, communicating with the deaf was only possible through written text and sign language.
- Both these methods are somewhat tedious, as it's not always practical to write down text and the general population being ignorant about sign language.

Objective:

- The objective of this project was to develop a real time sign language detection system that would break down the language barrier between the common public and the deaf.

Implementation:

- The solution to this problem requires the implementation of image processing concepts. We will use the OpenCV library for this.
- The solution is divided into two parts:
 - 1) Detecting the users hand through the webcam
 - 2) Detecting the gesture that the user has put up to the webcam

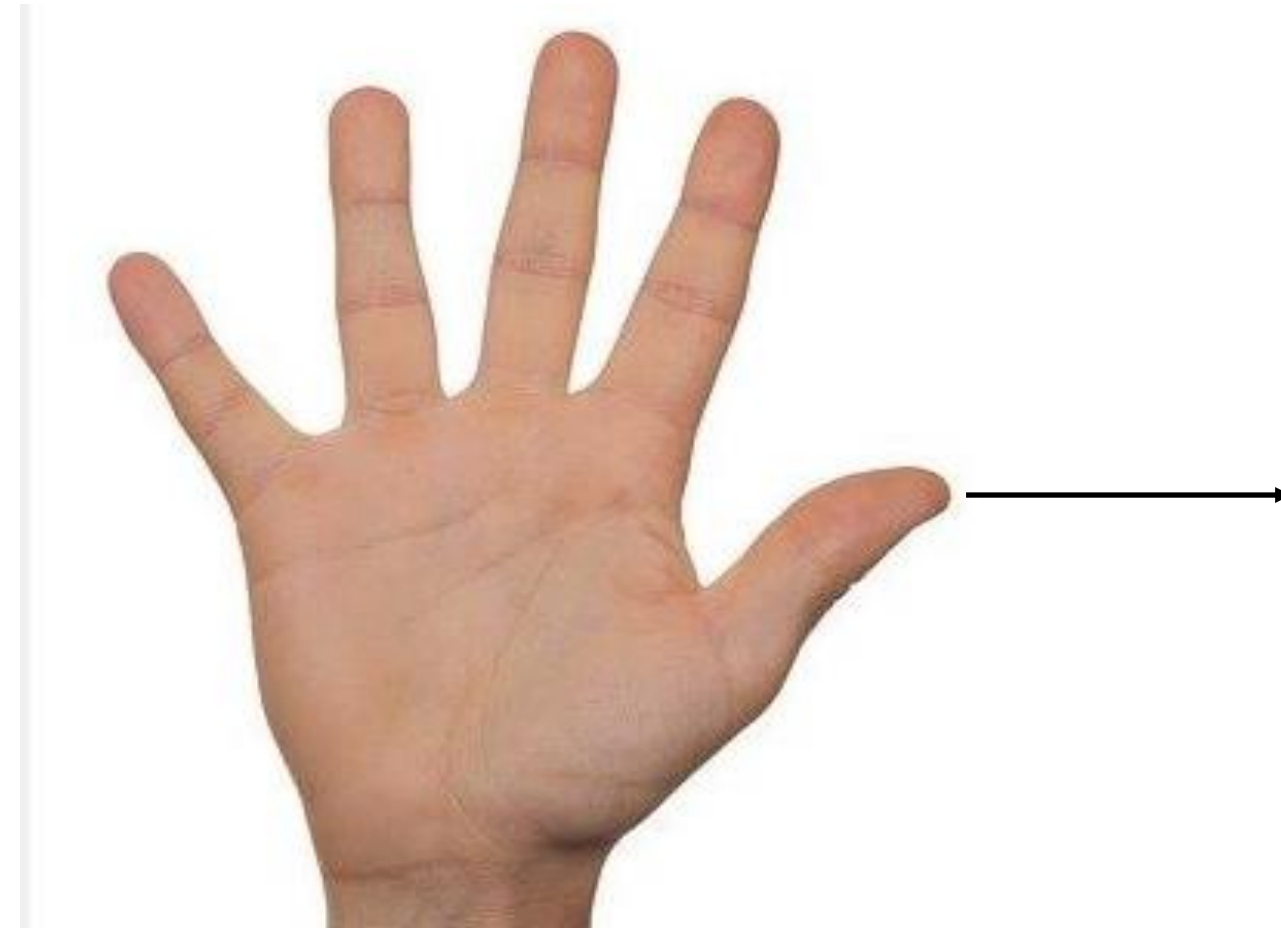
Detecting the users hand through the webcam:

- We first set up a region of interest (ROI) in which we will be detecting the user's hand. The ROI is a green box in which the user put his/her hand in.
- The green box in the image is our ROI, to detect an object in the ROI, we first draw contours around that object using the `findContours()` function
- Using the contours, we draw a convex hull using `convexHull()` function based on the contours and then calculate the convexity defects using `convexityDefects()` based on the contours and the hull.
- If these three steps are successful, then that means an object(in this case a user's hand) is in our ROI.

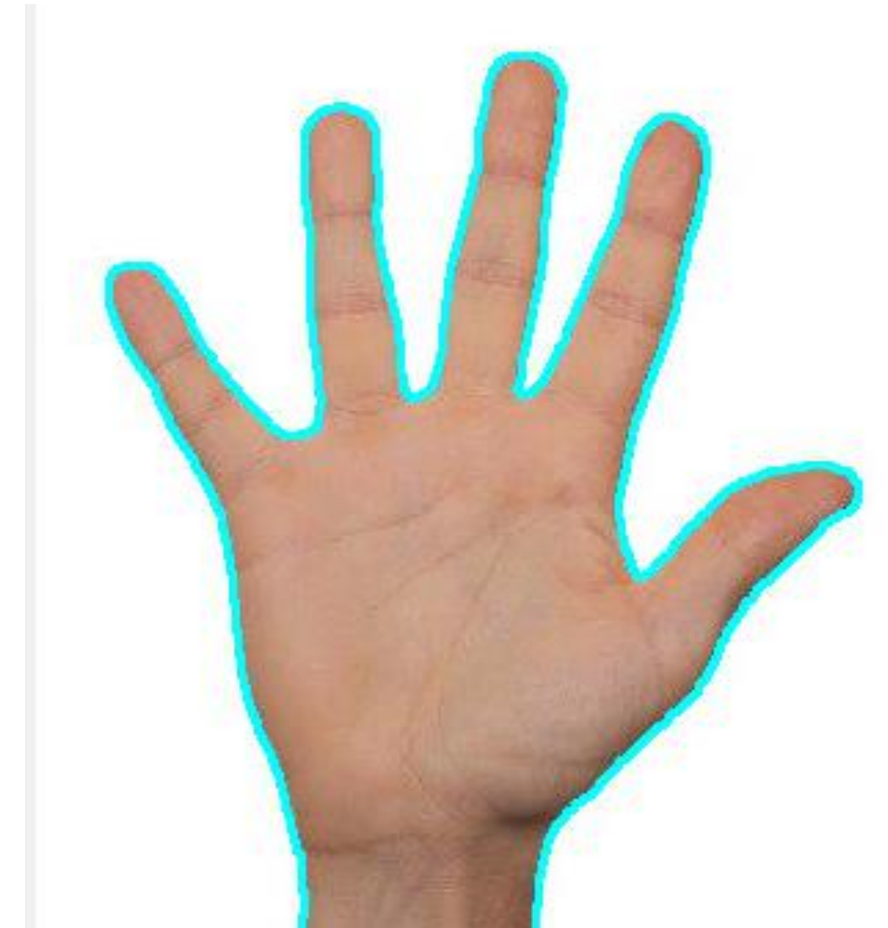
Region Of Interest:



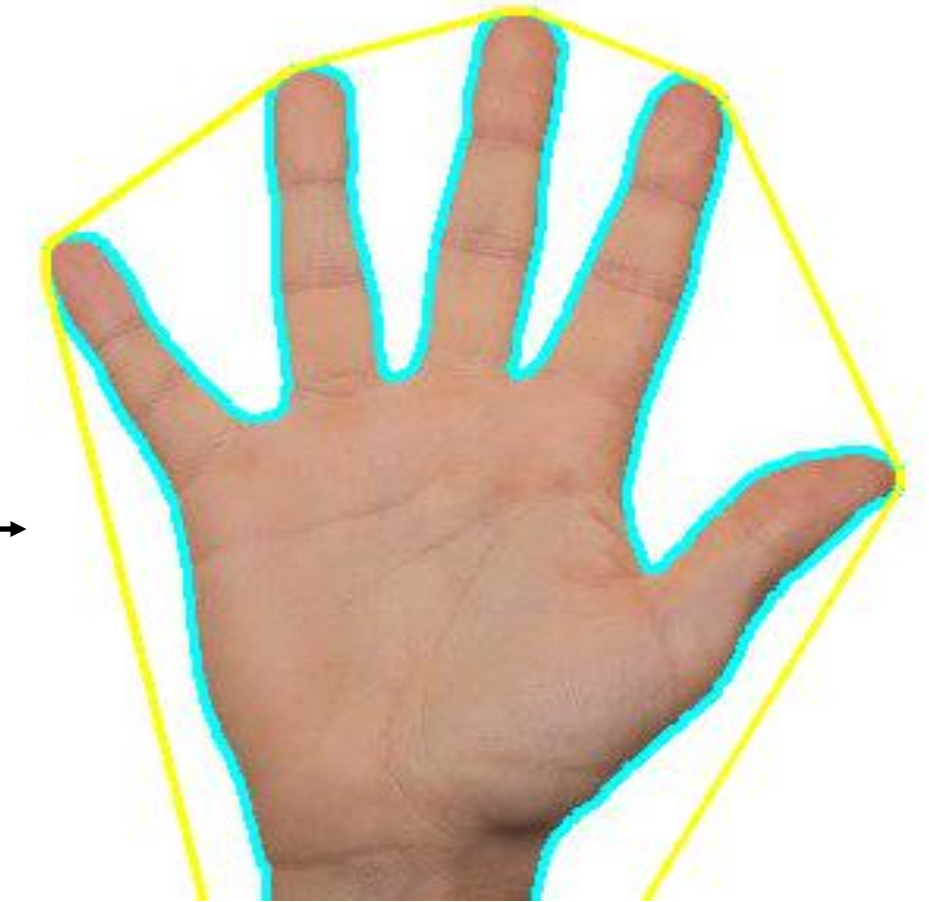
Hand Image:



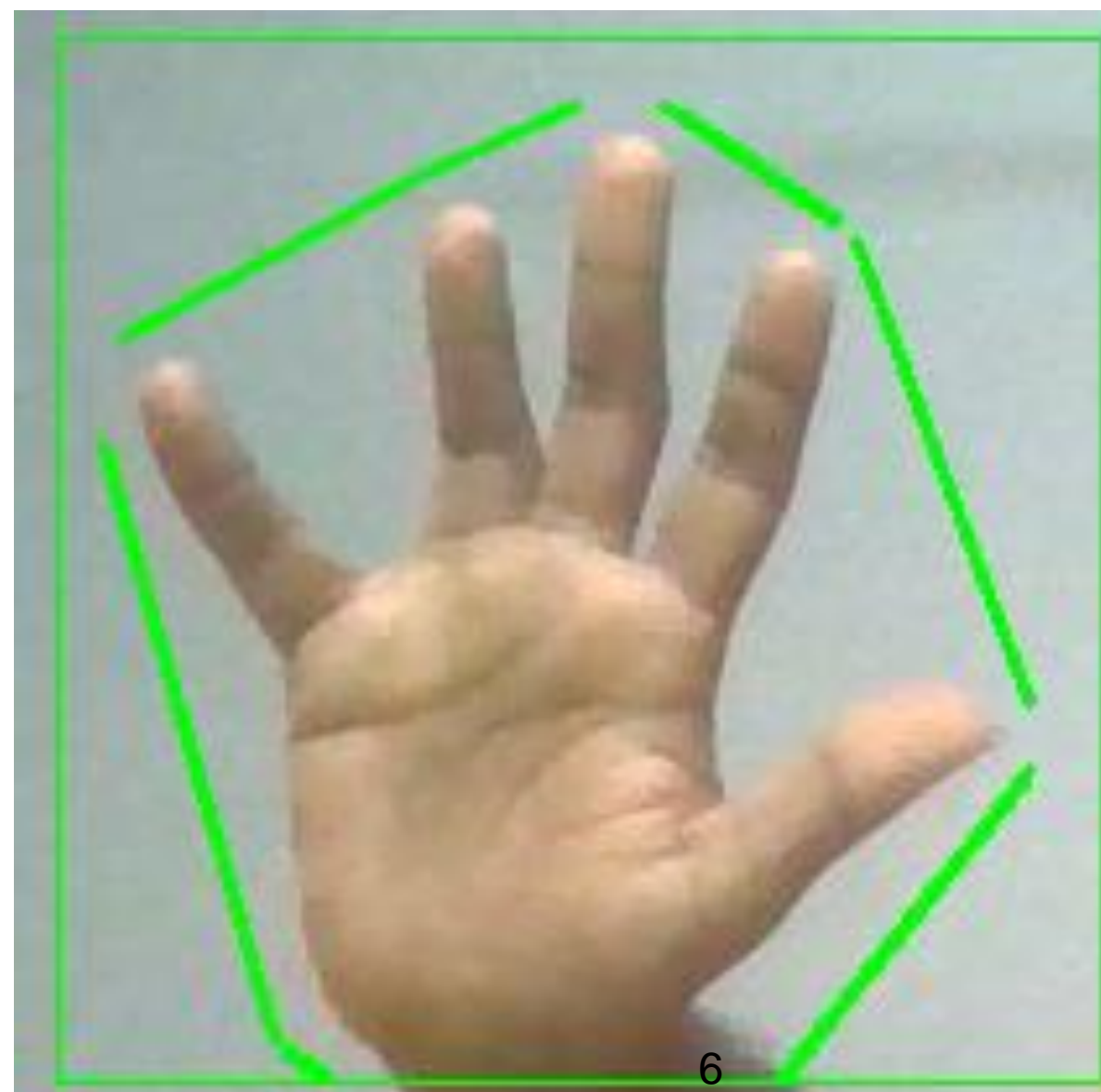
Contours drawn on hand:



Convex Hull:



Convex Hull output in webcam feed:



Detecting the gesture:

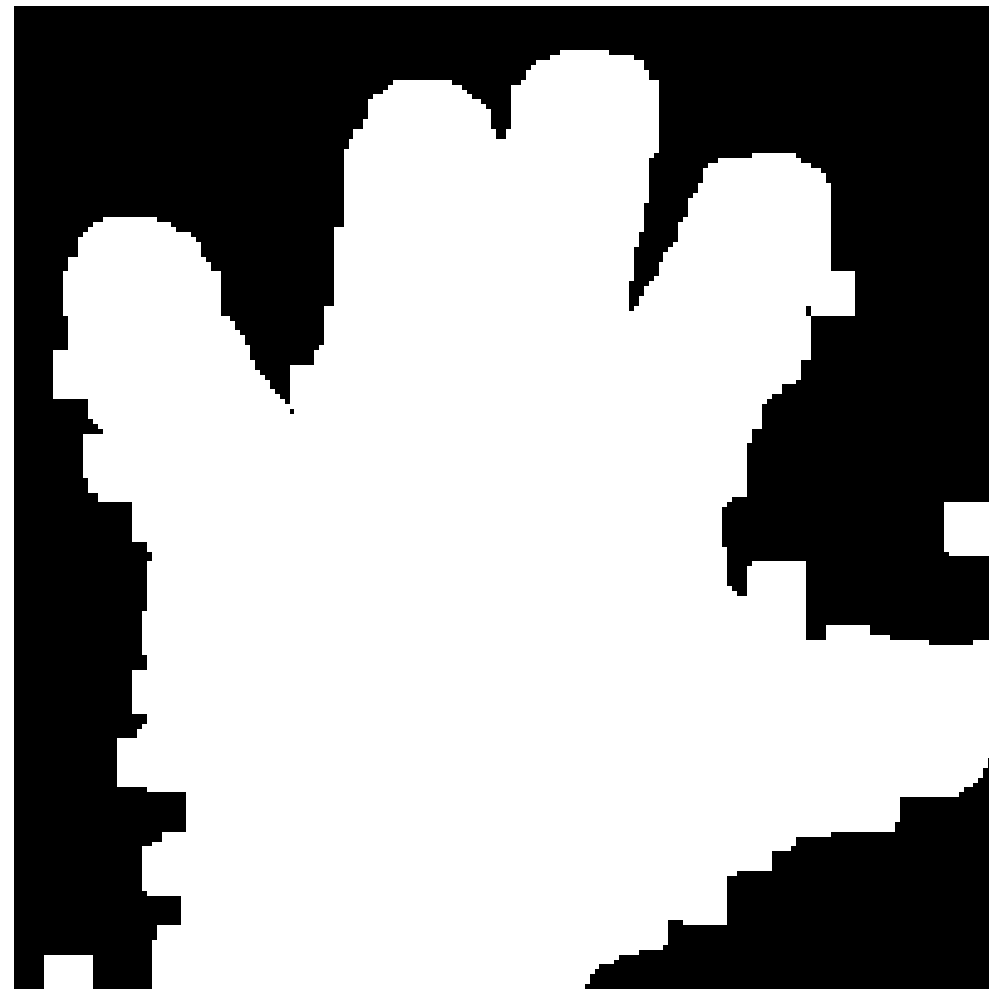
- The program currently works with 4 signs: Hello, Yes, No and I. These images are first converted to grayscale.
- We have the images for each of these signs and we calculate the histogram values of these images using the `cv.calcHist()` function which returns an array of values.
- Plotting these values gives a plot with pixel values (ranging from 0 to 255, not always) in X-axis and corresponding number of pixels in the image on Y-axis.
- The images and the plots for each of them are on the next slide:

Sign for Hello:

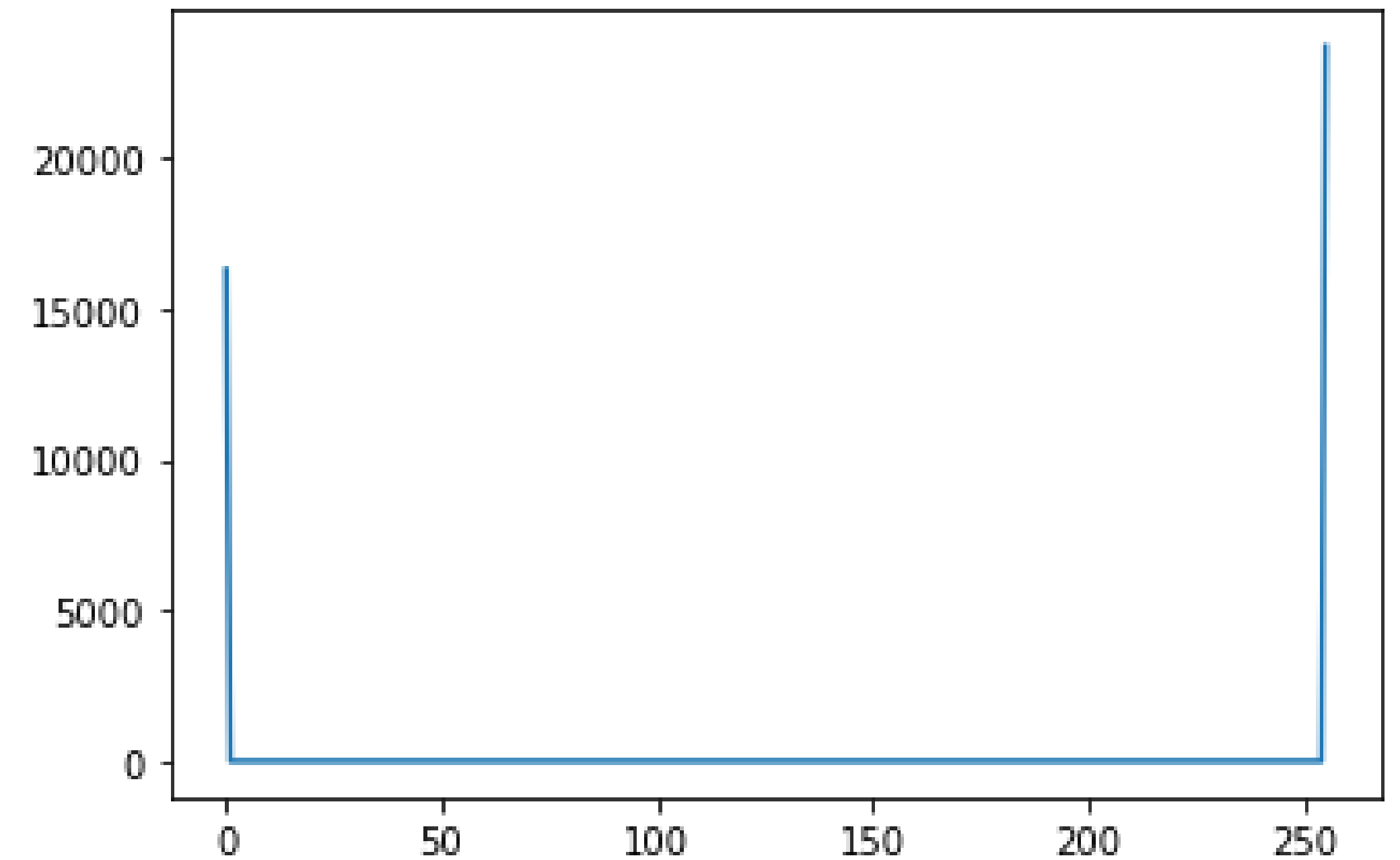
Original Image:



After converting to grayscale



Plot of Histogram values:

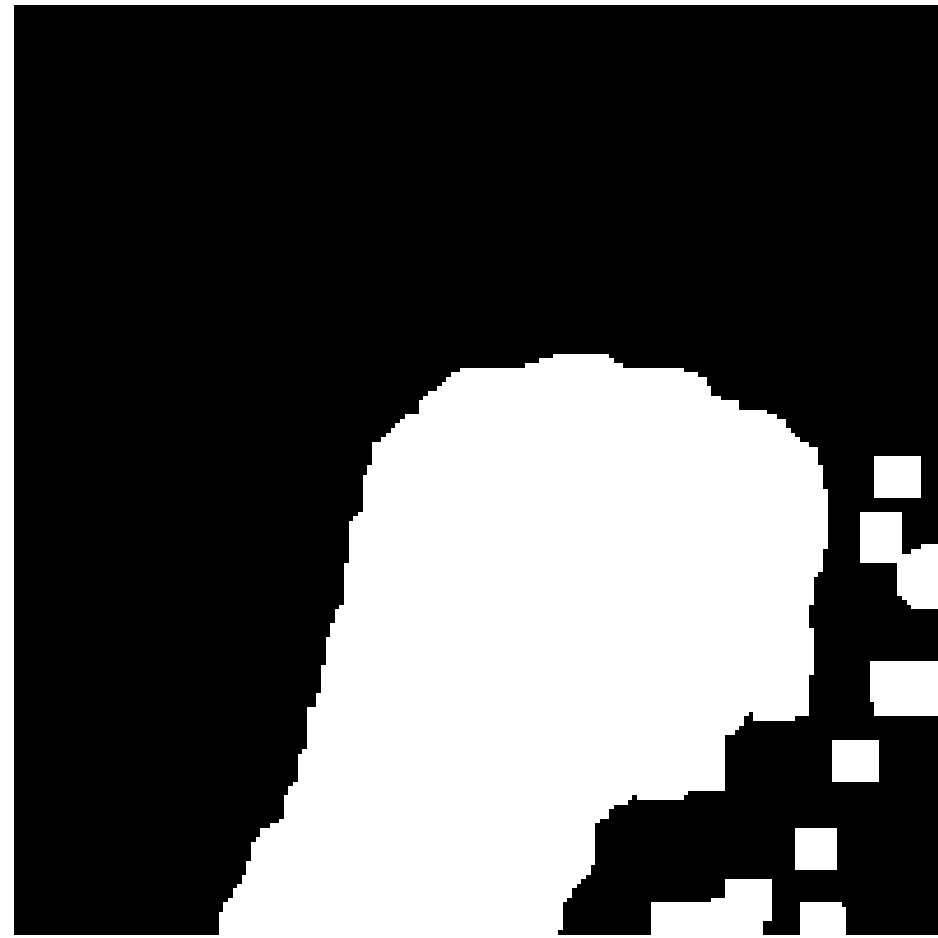


Sign for Yes:

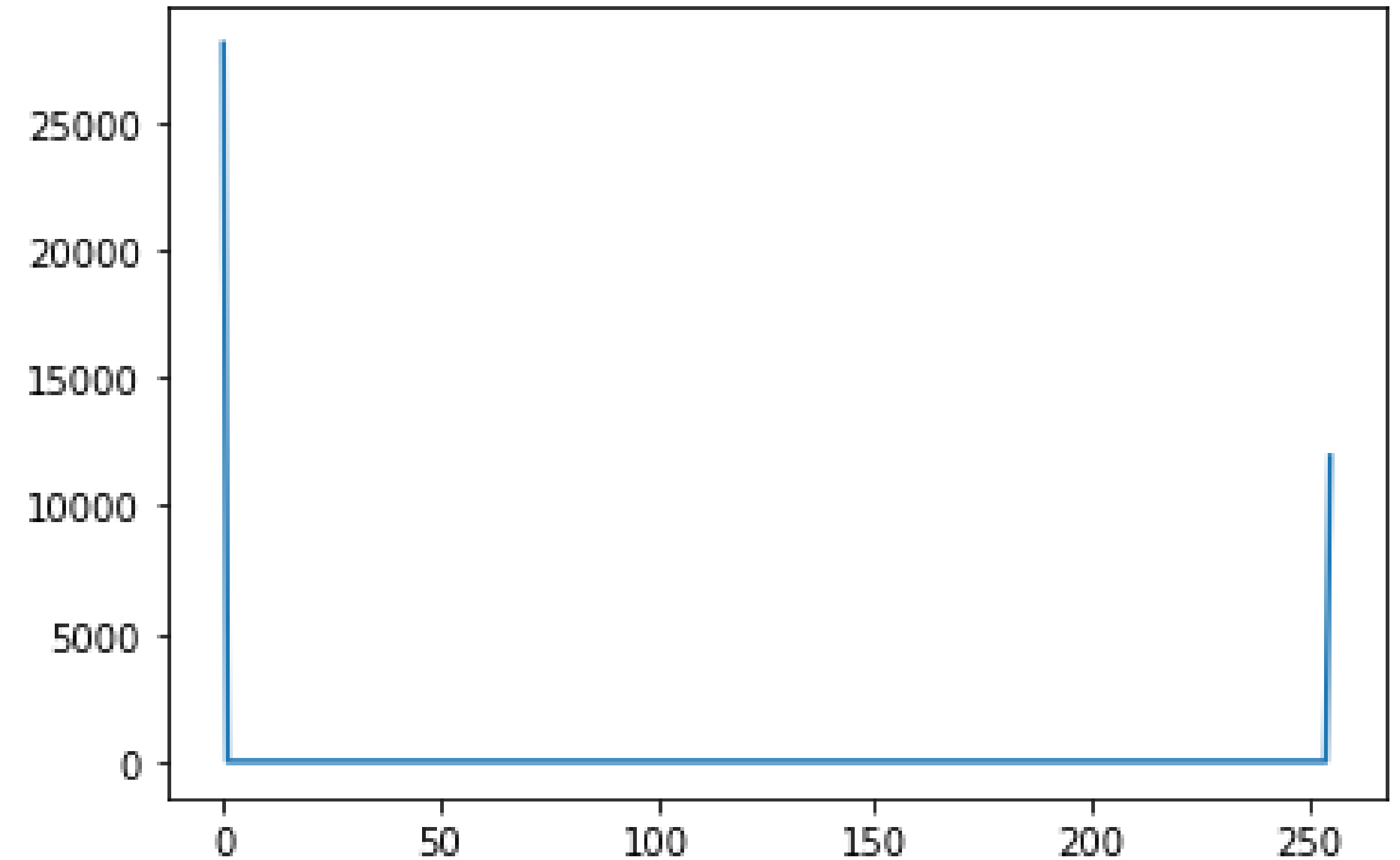
Original Image:



After converting to grayscale

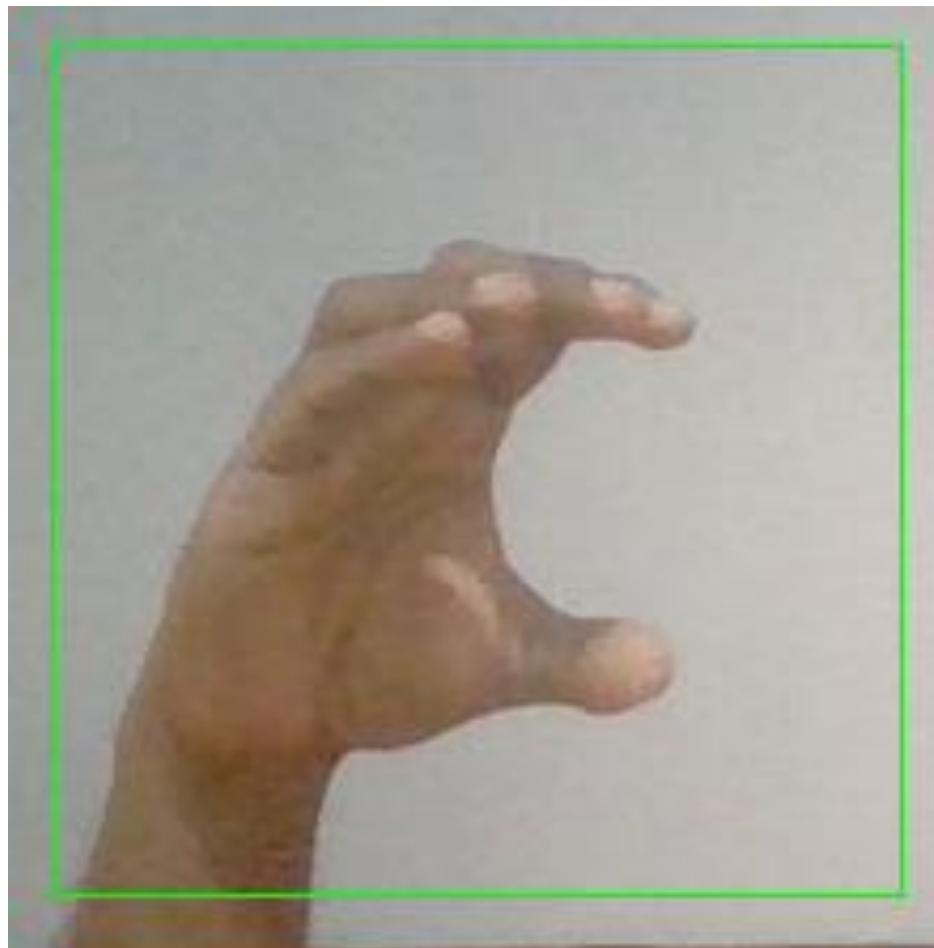


Plot of Histogram values:

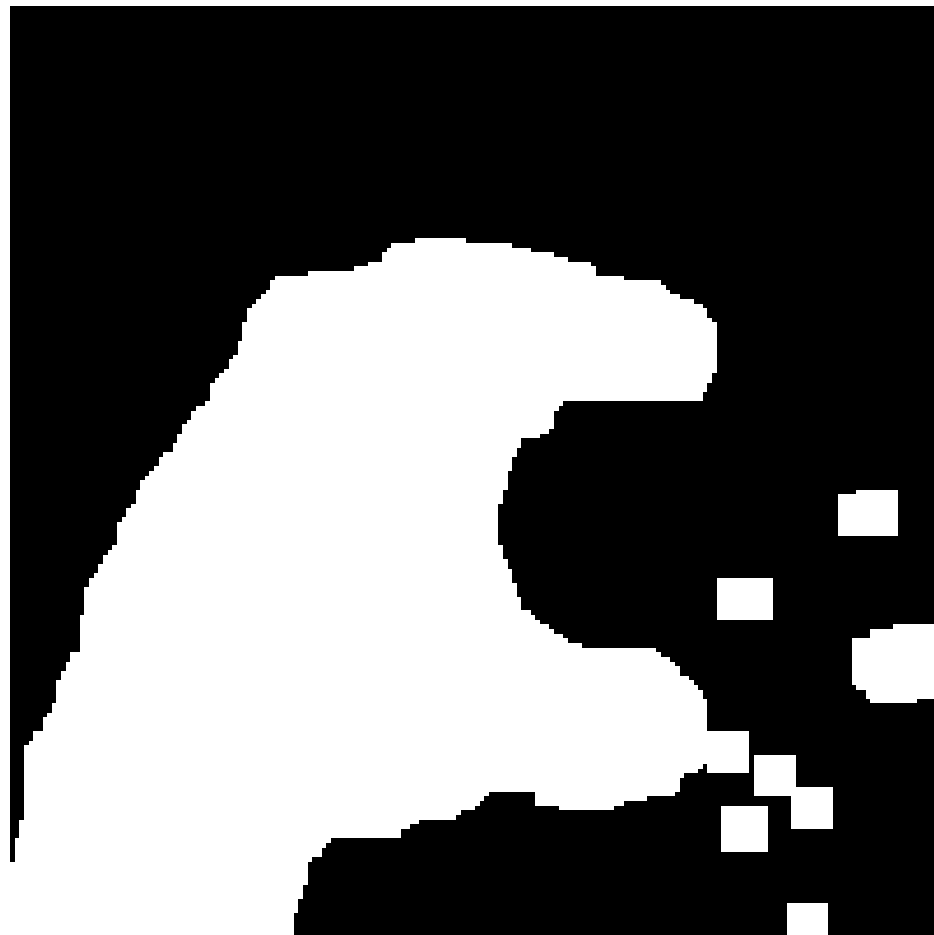


Sign for No:

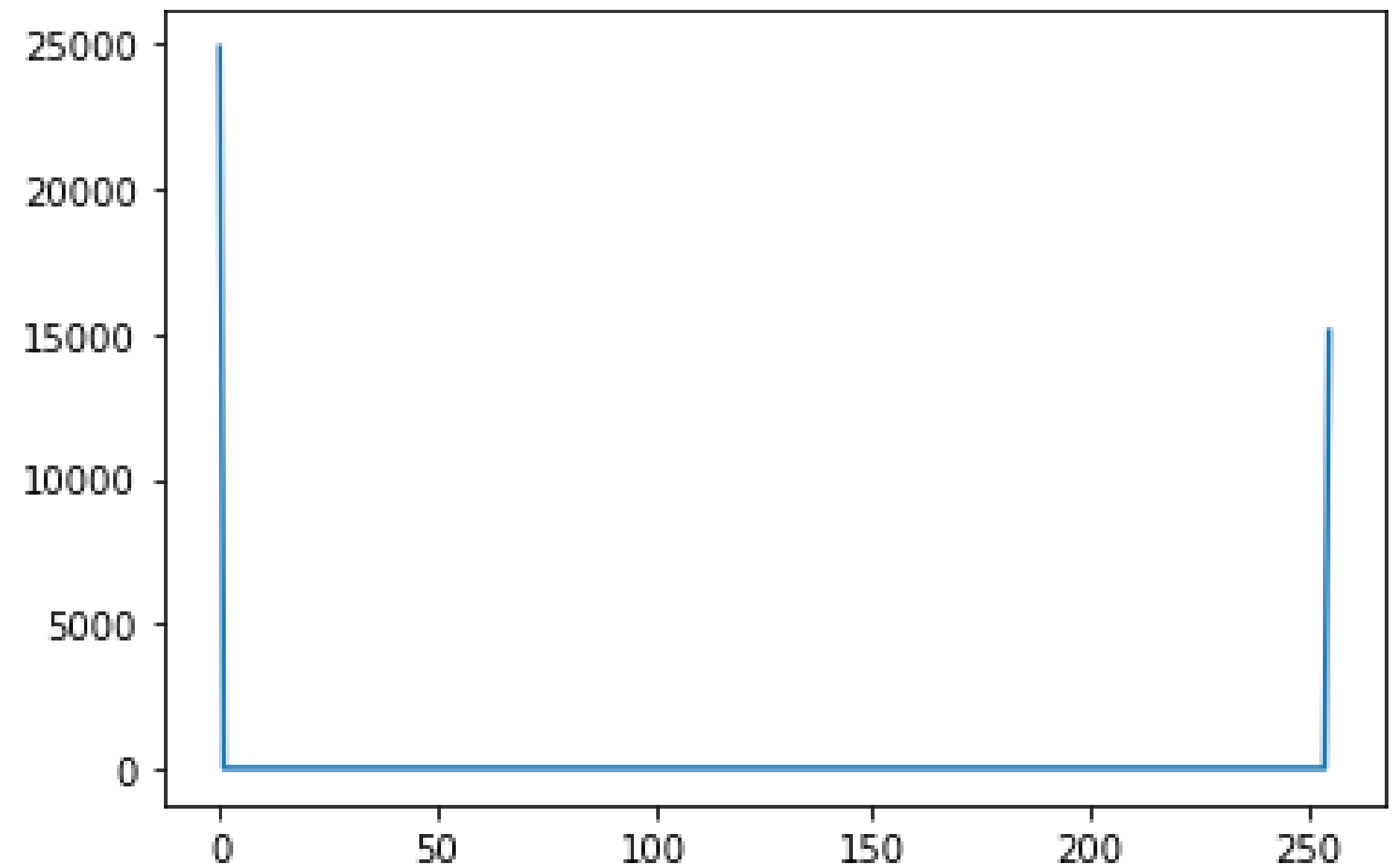
Original Image:



After converting to grayscale



Plot of Histogram values:

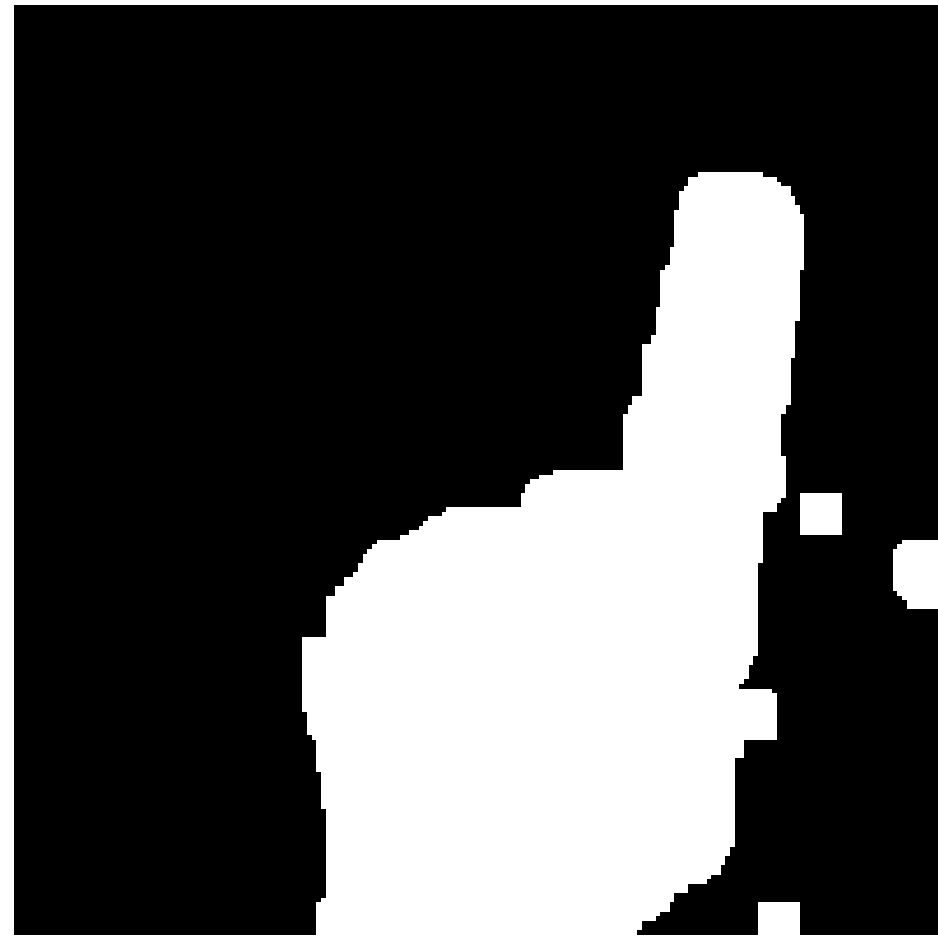


Sign for I:

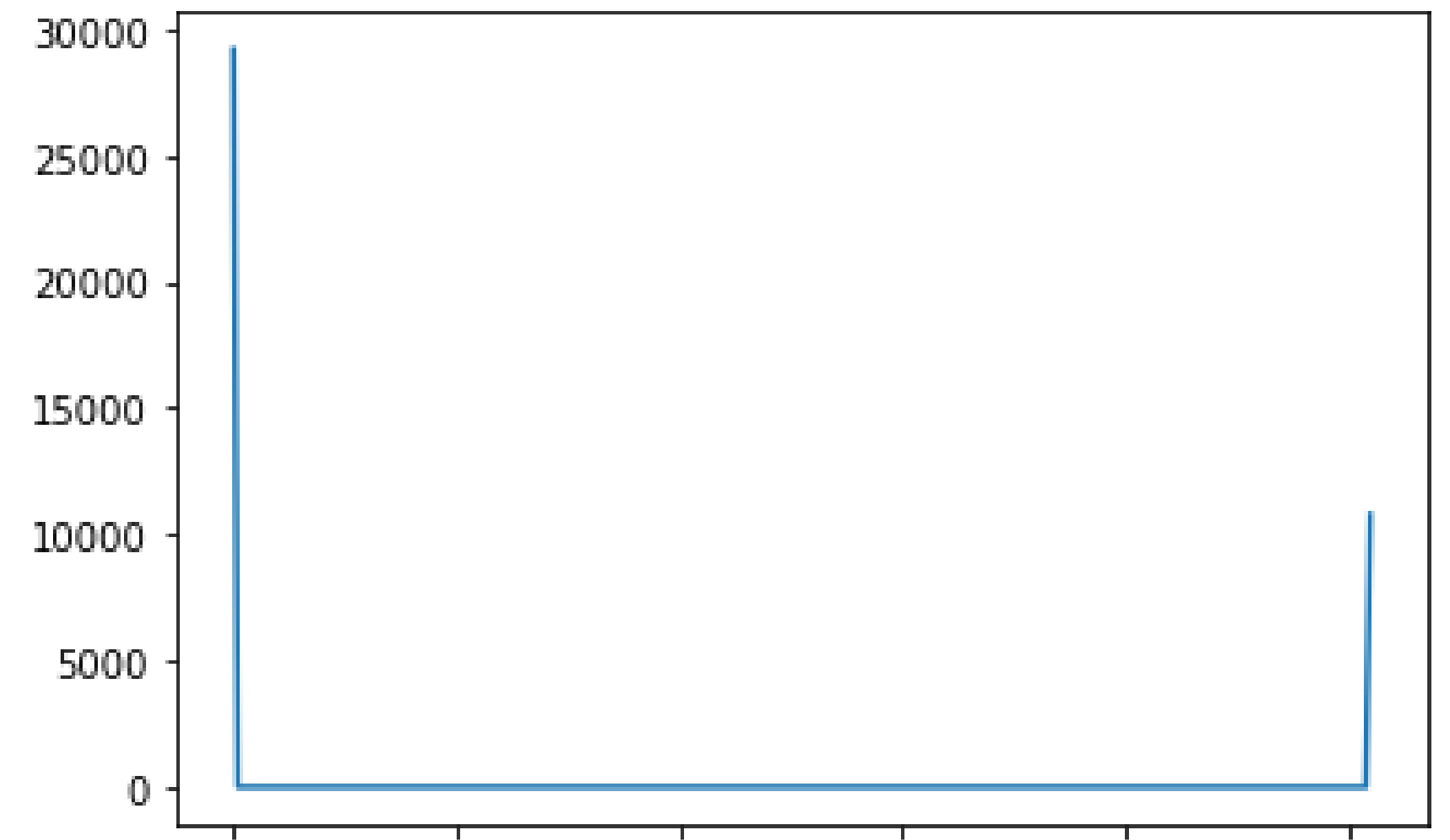
Original Image:



After converting to grayscale



Plot of Histogram values:



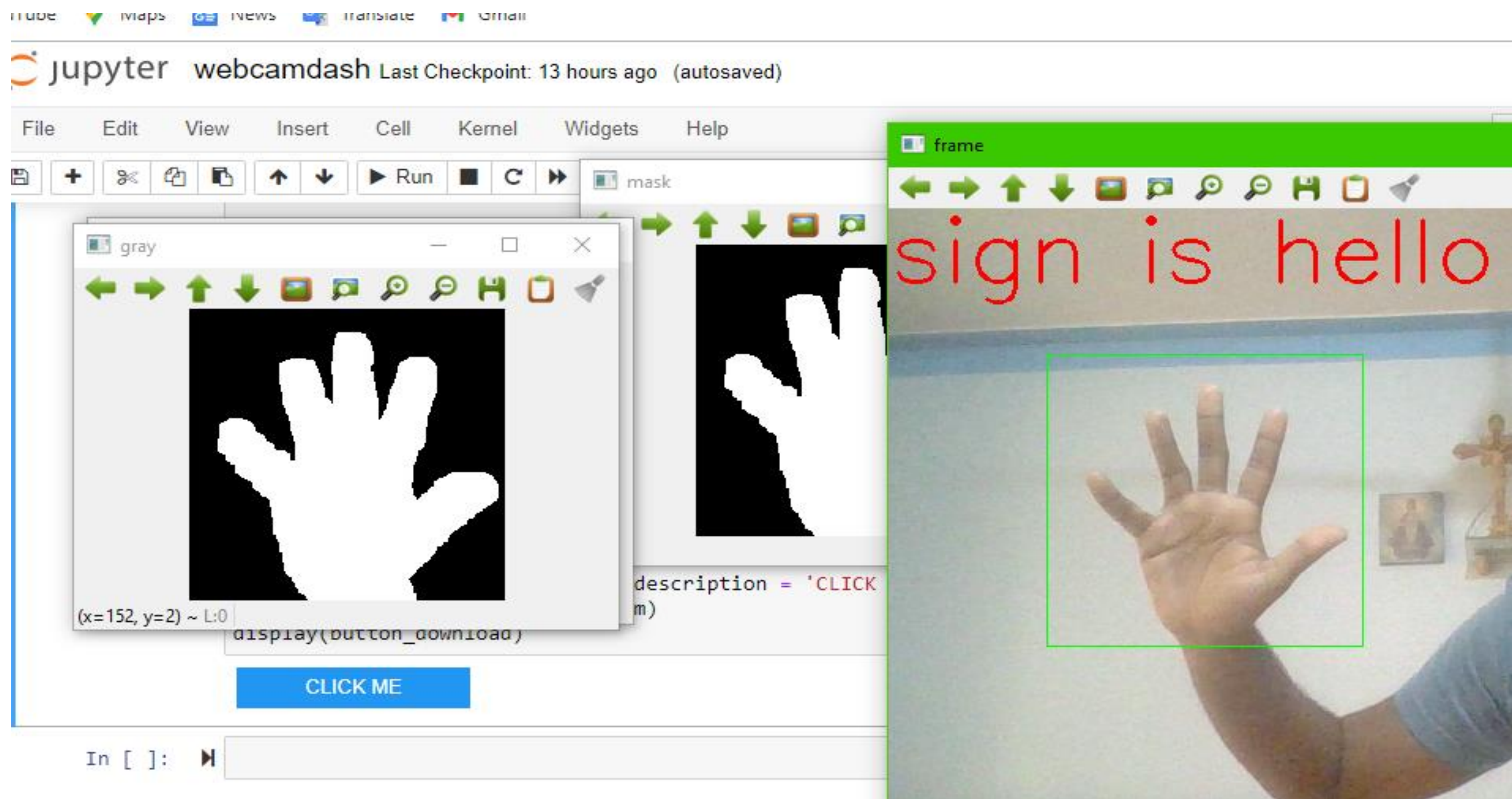
Detecting the gesture:

- After getting the histogram value array for each of the signs, we switch the webcam on and when the user puts up a gesture in the region of interest, the program detects the hand, clicks an image of that gesture, converts to grayscale and calculate the histogram value for that gesture as well (same processing that was done to sign images is done to this image as well)
- To figure out what gesture is put up, we have to see the similarity of that gesture with the signs we have.
- We do this by finding out how close the gesture histogram values is to each of the histogram values of the signs by calculating a value “C”.

Detecting the gesture:

- This value “C” is calculated using the formula:
while $i < \text{len}(\text{histogramofgesture})$ and $i < \text{len}(\text{histogramofsign})$:
 $c += (\text{histogramofgesture}[i] - \text{histogramofsign}[i])^2$
 $i += 1$
 $c = c^{(1/2)}$*
- We then compare the resultant “C” values of the gesture histogram value with those of the individual signs and pick out the lowest value. The lowest “C” value means the highest similarity between the gesture and the sign.
- Using this approach, we can detect sign language in real-time.

Final Output:



Dashboard:

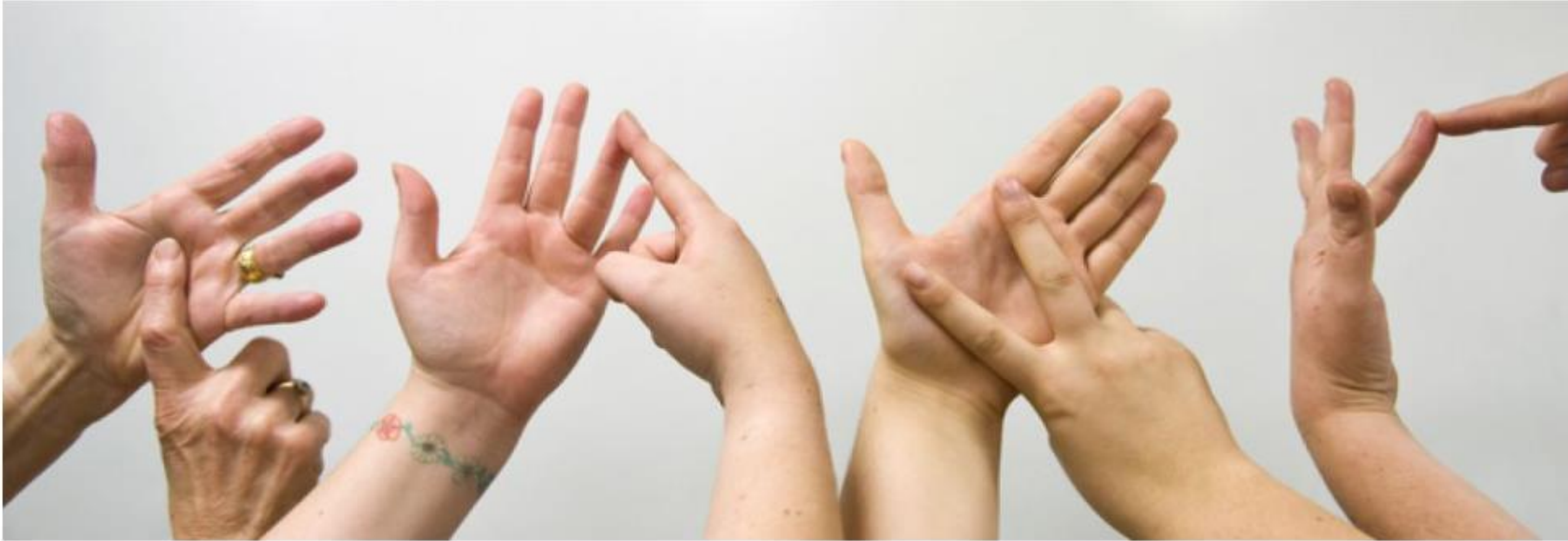
← → ↻ ⓘ localhost:8869/notebooks/PROJECTS/Sign%20language%20interpreter/dataf%20model/webcamdash.ipynb?dashboard 🔍 ☆ 🌐 ⚙️ 👤 ⋮

📱 Apps 📺 YouTube 📍 Maps 📰 News 🗨️ Translate 📧 Gmail

File Edit View Insert Cell Kernel Widgets Help Not Trusted | Python 3

Sign Language Interpreter

The sign language interpreter is an application which functions as a tool that can break down the communication barrier between the hard-of-hearing and the general public.



CLICK ME !!

A photograph of several hands raised in the air, suggesting a group agreement or applause. The hands are of various skin tones and some have visible tattoos. A large, semi-transparent blue rectangle is overlaid on the center of the image, containing the text "The End" in white. The background is a plain, light-colored wall.

The End