Fast Numeric Sign Detection Using Adaptive Thresholding and Geometry of Optimized Fingers

Mela G. Abdul-Haleem[1,*], Loay E. George[2]
[1]Department of Computer Science, College of Science, University of Baghdad, Baghdad, Iraq
[2] College of Science, University of Baghdad, Baghdad, Iraq
Emails: a.mela@sc.uobaghdad.edu.iq; loay.george@sc.uobaghdad.edu.iq

Abstract

A strong sign language recognition system can break down the barriers that separate hearing and speaking members of society from speechless members. A novel fast recognition system with low computational cost for digital American Sign Language (ASL) is introduced in this research. Different image processing techniques are used to optimize and extract the shape of the hand fingers in each sign. The feature extraction stage includes a determination of the optimal threshold based on statistical bases and then recognizing the gap area in the zero sign and calculating the heights of each finger in the other digits. The classification stage depends on the gap area in the zero signs and the number of opened fingers in the other signs as well as the sequence in which the opened fingers appear for those that have the same number of opened fingers. The conducted test results showed the system's high capability to classify all the digits; where both the precision and F-score percentages of the proposed model reached the desired optimal value (100%).

Keywords: ASL; Global thresholding; Chain coding; Edge detection; Elbow point extraction; Gaps\blots removal

1. Introduction

Each society has a slice of deaf or hearing-disability people who have suffered from hard communication with the other hearing-ability people living in their society. Most people from this slice tend to use sign language (SL) for communication and expressing what they want to say [1]. SL can be defined as an interaction style for hearing-impaired people through a set of hand gestures, facial expressions or movements corresponding to digits, letters and words. There are about three hundred used SLs around the world and the most famous SLs are ASL in which only one hand is used to represent a sign, Australian SL, Arabic SL, and Spanish SL [2, 3]. Some SLs require the use of both hands to give signs such as Turkish SL [4] and Indian SL [5, 6]. Although the using of a SL facilitates the lifestyle of this slice of society, they still suffer from the difficulty of communicating with hearing people since the last kind of people are not interested in learning the SLs as a result of the deficiency of standardized form of writing in these languages, the large differences between the sign and spoken languages, and low support for SLs by technologies of communication [7]. The trend towards finding a technological solution to facilitate the process of communication between deaf and hearing people has increased significantly. This solution aims to automatically distinguish SL and convert it into sound or text.

For sign language recognition (SLR), the previous studies focused on using artificial intelligence techniques such as Convolutional Neural Networks (CNN) and Connectionist Temporal Classification (CTC). The highest utilized method is CNN either alone for example the systems in [8-11] or merging it with another technique for example CNN with Bidirectional Long Short Term Memory Layers (BLSTM) [12, 13], CNN with Support Vector Machine (SVM) [14], and CNN with random forest classifier [15]. The other utilized method is CTC in combination with other methods such as HB RNN [16]. The drawbacks of these methods are the high degree of complexity in computations, the requirement of prior training, a need for large quantities of image data for system training correctly, and the probability of appearing convergence trouble during training that needs frequent layer adjustment

DOI: https://doi.org/10.54216/FPA.190201
Received: December 10, 2024 Revised: February 05, 2025 Accepted: March 03, 2025

and hyper parameters learning [17, 18]. Therefore, creating models depending on such methods takes a lot of effort.

The alternative technology that many researchers depended it for SLR is digital image processing. Digital image processing technology is a technique of using a computer to process information in images including image compression [19], enhancement [20], segmentation [21, 22], recognition [23], etc. [24]. The development of image processing techniques and pattern recognition applications increases at a high rate. So, these fields represent a good solution to build a system that satisfies the communication aim between deaf and hearing people. In [25], the authors presented an American SLR system based on Canny Edge and Histogram of Oriented Gradient (HOG) at the local feature extraction stage and k-nearest neighbor (K-NN) for classification. The accuracy of the presented system was 94.2%, while the 86% recognition rate was obtained in the same research when the authors used a bag of features in addition to SVM with k-means for clustering. The same techniques (HOG, and K-NN) have been adopted in [26] with system recognition at 94.1% and in [27] where the average recognition rate for the used Kaggle and Massy University datasets were 99% and 97% respectively. In [28], the authors utilized the YCbCr model for skin segmentation, HOG for feature extraction, and SVM as a classifier for an American SLR. The gained accuracy was 89.54%. An American SLR model for sixteen alphabets was introduced as a mobile application in [29]. The model utilized both Canny Edge and seed-filling methods for hand segmentation as well as speed up robust features (SURF) algorithm at the feature extraction stage and K-means clustering to obtain 97.1% recognition rate. The model showed false classified signs with a high degree of similarity. In [30], the authors used the Scale Invariant Feature Transform (SIFT) method for Indonesian digital SLR and obtained 82% recognition rate for a dataset of 100 images.

This research produces a fast and novel system for recognizing the digits of ASL based on image processing techniques and the principle of pattern recognition. The system includes pre-processing, feature extraction and classification processes as explained in section (3). The description of the used dataset is given in section (2). Some evaluation results for the efficiency of the system performance are explained in section (4). Section (5) gives some conclusions and predictions for the system development in the future.

2. Dataset

The used dataset was taken from the "Sign Language Digits Dataset" [31] and it also exists at "https://www.kaggle.com/ardamavi/sign-language-digits-dataset", where the sign images are of small size (100x100 pixels) in RGB color space, and there are ten classes (0-9 digits) in the dataset. Forty samples were taken for each class and the samples are characterized by different brightness with the presence of rotation and displacement of the hand's location in the samples' images. The total number of samples in our dataset is four hundred samples.

3. System Layout

In this paper, a numeric sign recognition system based on adapted thresholding and the geometrical shape of the fingers is proposed. The used image processing techniques of the system including preprocessing, feature extraction and classification processes are given in figure 1.

3.1 Preprocessing Stage

This stage implies the following steps:

A. Calculating the Mean and Standard deviation

To detect hand, it is important to select the optimal channel from RGB channels to work with. For best choosing, the mean (M) and standard deviation (σ) of each channel are calculated using Equations (1 and 2) [32].

$$M = \frac{1}{w*h}\sum_{y=0}^{h-1}\sum_{x=0}^{w-1} ch(x,y) \quad (1)$$

$$\sigma = \sqrt{\sum_{y=0}^{h-1}\sum_{x=0}^{w-1}\frac{(Ch(x,y)-Mc)^2}{w*h}} \quad (2)$$

Where the channel's value is ch and $w * h$ is the image's size.

DOI: https://doi.org/10.54216/FPA.190201
Received: December 10, 2024 Revised: February 05, 2025 Accepted: March 03, 2025

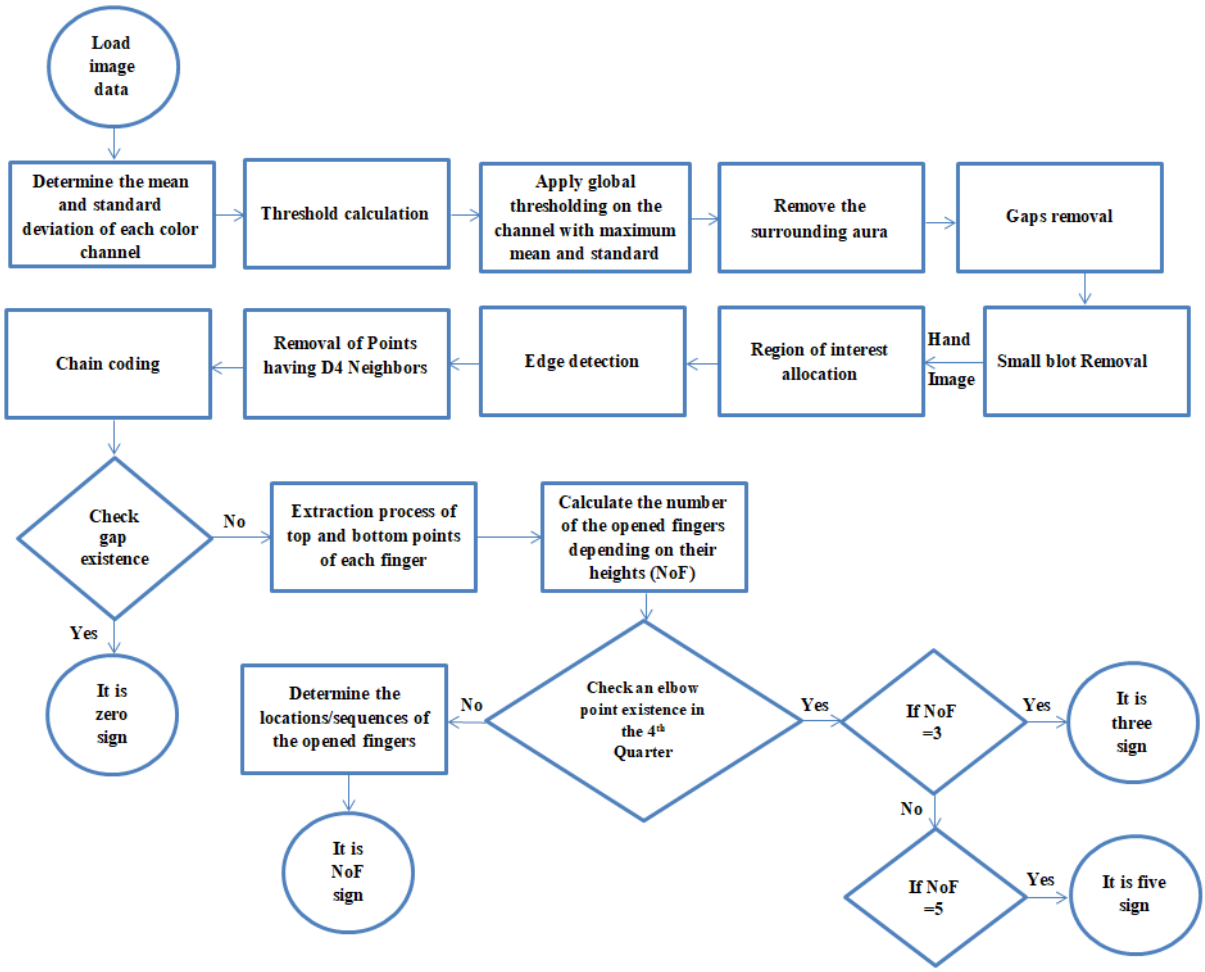

Figure 1. A flowchart of the system

B. Applying Global Threshold

Thresholding is a well-known image processing technique that is used to facilitate image analysis, image segmentation, feature extraction, or recognition of a region of interest in images while ignoring uninterested regions [33, 34]. To calculate the adaptive threshold value (Th), Equation (3) is used. Then, each value of the blue channel is compared with the threshold value to generate the binary image using Equation (4). Figure 2 shows the generated images by applying Eq.3 and Eq.4 on each channel. The selection of the blue channel gives the best result.

$$Th = Mc - Sc * \propto \quad (3)$$

$$Bc(i,j) = \begin{cases} 1 & if\ ch(i,j) > Th \\ 0 & if\ ch(i,j) < Th \end{cases} \quad (4)$$

C. Remove the Surrounding Aura / Shadow

It is obvious from figure 2, the existence of a black aura surrounding the hand region. This black region does not contain any benefit value and must be removed. Loop on the image starting from the origin pixel (0, 0) to the last pixel in the most right corner. Convert each black pixel to a white pixel. Exit the loop when reaching a white pixel. Repeat the process by starting the loop from the last pixel in the most right corner. The effect of this process is shown in figure 3.

DOI: https://doi.org/10.54216/FPA.190201
Received: December 10, 2024 Revised: February 05, 2025 Accepted: March 03, 2025

Figure 2. Applying a global thresholding; (a) The original image, (b) Applying the global thresholding process on the red channel, (c) Applying the global thresholding process on the green channel, and (d) Applying the global thresholding process on the blue channel

D. Gaps\Black Blots Removal

The operation starts by finding the desired point and then filling the object contours starting from that point. So, the white or black gaps removal algorithm consists of two parts. The first part includes finding a starting desired value and then checking the connected pixels in four directions until completing the gap, while the second part includes the condition that prevents the filling of the white zero's gap or the removal of all black gaps in the image. This condition imposes the criteria on the area of the gap that will be removed by restricting it to only a small ratio of the total image area. For small black blots removal, the area of these blots must not exceed 0.01, while for the white gaps, the changes in the x-axis (Δx)/ the changes in the y-axis (Δy) must be greater than 0.7 and less than 1.57. This condition shows good results in distinguishing between some samples of nine signs that contain gaps and the zero samples since it helps to remove the white gaps of some nine signs because they do not exceed the area in the condition because of hand movement where three fingers must be opened. On the other hand, the gaps in the zero signs do not satisfy the condition. The other way that can be used is checking the existence of opened fingers in nine signs. Figure 3 shows the effect of applying gaps and blots removal on the image. Figure 3(b) shows a sample of nine sign containing a gap.

Figure 3. Aura and gaps/blots removal; (a) The resulting 'Zero' sign image after applying the aura removal stage, the black blots, and the white gaps removal stages respectively; (b) The resulting 'Nine' sign image after applying the aura removal stage, the black blots, and the white gaps removal stages respectively

E. Region of Interest Allocation

A region of interest (ROI) has several shapes like a polyline, a polygon and a rectangular shape [35]. A rectangular-shaped ROI is extracted from the image, which contains the hand area to exclude the white region around the hand area at the beginning, and since the feature extraction stage depends on the geometry of the fingers, the lower third part of the hand image is ignored.

DOI: https://doi.org/10.54216/FPA.190201
Received: December 10, 2024 Revised: February 05, 2025 Accepted: March 03, 2025

F. Edge Detection

An adaptive edge detection method is introduced in Algorithm 1. To simplify the algorithm a structured (3x3) window is opened on each hand point E(x, y) as explained in figure 4.

E7 (x-1, y-1)	**E6 (x-1, y)**	**E5 (x-1, y+1)**
E0 (x, y-1)	**E (x, y)**	**E4 (x, y+1)**
E1 (x+1, y-1)	**E2 (x+1, y)**	**E3 (x+1, y+1)**

Figure 4. A 3x3 structured window

Algorithm (1): Detect the edge points

Input: E(): ROI Image (hand points=1, background points=0)
Output: Edge() Image of Edge Detected hand.

Step 1: Loop on the image E()
Check if* E point = 1 (hand point) *then
Get the structured values of the explained window in figure 4.
Check if* (E6=1 and E2=0) or (E6=0 and E2=1) *then* E is an edge point. *End if
End if
End loop
Step 2: Loop on the image E()
Check if* E point = 1 (hand point) *then
Check if* (E0=1 and E4=0) or (E0=0 and E4=1) *then* E is an edge point. *End if
End if
End loop
Step 3: Put the Edge points in a new image.
Return Edge()

G. Removal of Points Having D4 Neighbors

Since the geometrical features depend on the extraction of the top and bottom points of the fingers, the existence of points that have D4 neighbors on the hand edges will lead to extract spurious top and bottom points. The edge points and the points of D4 neighbors are shaded in figure 5. If the edge point and its D4 neighbors have a value equal to the hand points, the edge point at the center of the structure must be removed. The two states at the top of figure 5 appear in a lot of the selected dataset. Figure 6 shows examples of edge points explained in black points and the points of D4 neighbors explained in red points. The colors in figure 6 are used for explanation only.

Ed (x-1, y-1)	**Ed (x-1, y)**	**Ed (x-1, y+1)**
Ed (x, y-1)	**Ed (x, y)**	**Ed (x, y+1)**
Ed (x+1, y-1)	**Ed (x+1, y)**	**Ed (x+1, y+1)**

Ed (x-1, y-1)	**Ed (x-1, y)**	**Ed (x-1, y+1)**
Ed (x, y-1)	**Ed (x, y)**	**Ed (x, y+1)**
Ed (x+1, y-1)	**Ed (x+1, y)**	**Ed (x+1, y+1)**

Ed (x-1, y-1)	**Ed (x-1, y)**	**Ed (x-1, y+1)**
Ed (x, y-1)	**Ed (x, y)**	**Ed (x, y+1)**
Ed (x+1, y-1)	**Ed (x+1, y)**	**Ed (x+1, y+1)**

Ed (x-1, y-1)	**Ed (x-1, y)**	**Ed (x-1, y+1)**
Ed (x, y-1)	**Ed (x, y)**	**Ed (x, y+1)**
Ed (x+1, y-1)	**Ed (x+1, y)**	**Ed (x+1, y+1)**

Figure 5. The edge point at the center of the structure and its D4 neighbors

DOI: https://doi.org/10.54216/FPA.190201
Received: December 10, 2024 Revised: February 05, 2025 Accepted: March 03, 2025

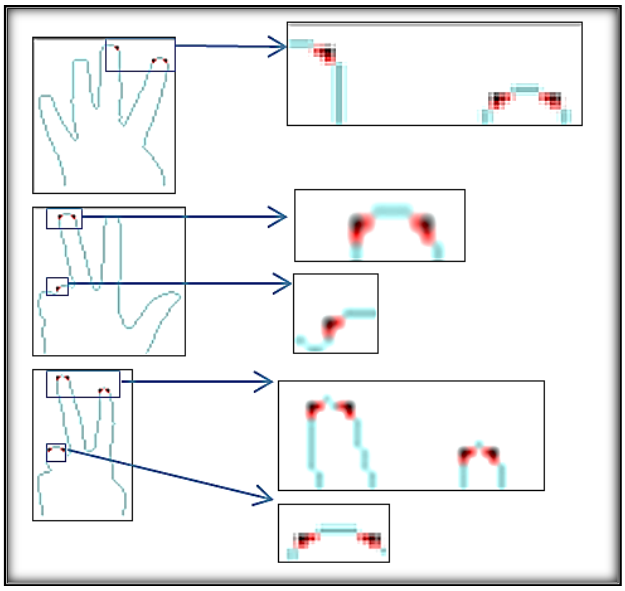

Figure 6. Examples of edge points with their D4 neighbors

H. Hand Curve Allocation and Chain Coding

The existence of a gap in the zero sign represents a good characteristic to classify this class. The output of edge detection for the zero sign is a hand curve and a hand middle gap, while the output for other signs is a curve of hand edges only (see figure 7). First, a start point of the curve at the lower left corner is specified. The determination of the start point is done by starting a loop on a tested sample from the lower left corner and checking the first existence of an edge point where it does not exceed 0.47 of the image's width. Then, the eight-neighbor chain coding method is used to determine the positions of the curve points and remove the curve from the zero sign image. If there are other edge points in the image, the existence of gap will be checked using the eight-neighboring boundary filling algorithm to recognize the Zero class (see figure 8). Otherwise, the determined curve points are preserved in another image for recognition of the other classes.

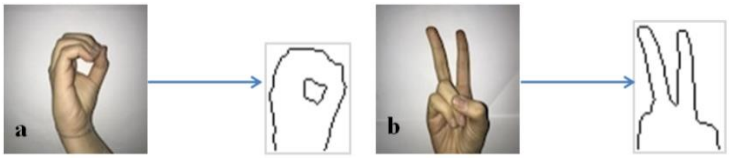

Figure 7. The results of the edge detection process on two samples; (a) A 'zero' sign sample; (b) A 'two' sign sample

Figure 8. The result of applying the chain coding process on a 'Zero' sign image

3.2 Feature Extraction and Classification Stages

A. Gap Area of a Zero Sign

The eight-neighboring boundary filling method is used to determine the area of the gap. For all tested zero signs, this area lay between (0.009 and 0.02) of the total area of the tested image.

B. Calculate the Number of Opened Fingers

DOI: https://doi.org/10.54216/FPA.190201
Received: December 10, 2024 Revised: February 05, 2025 Accepted: March 03, 2025

The other classes are classified according to the geometry of the fingers, and the number of opened fingers (NoF) in each sign. To calculate the number of opened fingers, the height of each finger is calculated by taking the distance between the maximum top point and bottom point of each finger. The maximum height of all five fingers' heights is extracted to distinguish between the opened fingers and the closed ones. The opened fingers have heights that exceed a specific ratio of the maximum height and then the number of these fingers is obtained. It must be obvious that 3, 6, 7, 8, and 9 digits have three opened fingers. Hence, the sequence in which the opened fingers appear plays a critical role for distinguishing between them. For example, in digit 'six' the second, third and fourth fingers are opened fingers and have heights that exceed the determined ratio, while the first and last fingers are closed fingers and have heights smaller than the ratio. For digit 'Nine', the first, second, and third fingers are opened fingers, have heights that exceed the determined ratio, while the fourth and fifth fingers are closed fingers, and have heights smaller than the ratio and so on.

The process of identifying the top and bottom points is a challenge because of the existence of a straight line at the top of a finger (i.e., many top points for the same finger), the appearance of a spur region at the middle of fingers which may lead to the appearance of spurious top points and the sort in which the top and bottom points appear where the heights of the fingers are calculated. As well as for the (Three or Five) sign, there is a large concave in the curve, between the fourth and fifth fingers in the images of the used data set where many points may wrongly be considered as top or bottom points. All these problems are solved in the proposed system. The identification process of the top and bottom points is explained in Algorithm 2. Algorithm 3 explains the process of extraction NoF feature. Figure 9 shows the output of applying Algorithm 2 on samples in the utilized dataset where the top points were colored in black and the bottom points were colored in red.

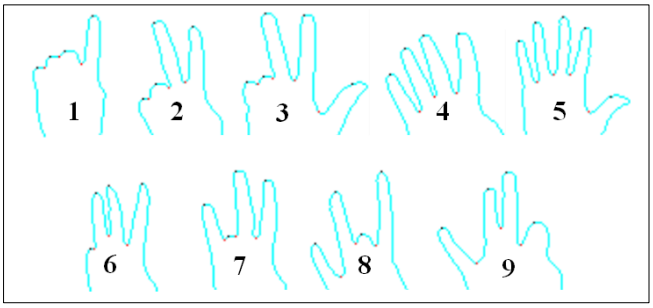

Figure 9. The top and bottom points

Algorithm (2): Top and bottom points detection

Input: E1(): The image contains the Curve points as black points
Xq(): an array of x-coordinates of the curve points
Yq(): an array of y-coordinates of the curve points
K: total number of curve points
Output: Top(): an array of records (x-position, y-position),
Bottom(): an array of records (x-position, y-position)

Extract all potential top points. Each top point has a height lower than the two surrounding curve points
Loop on the image E1() starting from the original point (0,0)
Loop on curve points
Check if* y-coordinate of the next curve point ≥ y- coordinate of the current curve point *then
Check if* y-coordinate of the previous curve point > y-coordinate of the current curve point *then
Top()← x and y-coordinates
Increment counter M // M is the number of top points
End if
End if

DOI: https://doi.org/10.54216/FPA.190201
Received: December 10, 2024 Revised: February 05, 2025 Accepted: March 03, 2025

End loop

End loop

Eliminate the repetition of top points at the top of the same finger by taking the average of the x-coordinates of sequential top points that have the same height and the difference between the x-coordinates is 1.

liminate spurious top points.

Step 3.1: Calculating the distance between each two subsequent top points using:

is an //index of top()

if the dis is not exceeding 7, replace the two top points with the mid of the coordinates of the two top points. Otherwise, keep the first top point at the same coordinates.

Enter the last top point in Top().

Delete the repetition of the top points in the Top array (delete the duplication of the top points that have the same coordinates)

If the new top point (mid-point) is not on the hand curve, then search for the nearest curve point to it in eight directions and replace the nearest curve point with the mid-point.

xtract all potential bottom points.

The bottom points lie at the bottom of each finger. So, start from each top point and follow the curve points next to the top point until reaching the maximum height of the sequential curve points.

***Loop* on the top points**

***Loop* on the curve points**

Find the top point at the curve

***Loop While* y- coordinate of the next curve point ≥ y- coordinate of the current curve point**

End loop

Bottom()← x and y-coordinates

Increment counter

End loop

End loop

Eliminate the repetition of bottom points at the bottom of the same finger by taking the average of the x-coordinates of sequential bottom points that have the same height and the difference between the x-coordinates is 1.

Delete the repetition of the bottom points in the Bottom array (delete the duplication of the bottom points that have the same coordinates)

ort both the top and bottom points in such a way that the x-coordinates appear in ascending order in the Top and Bottom arrays.

Step 6: Extract an elbow point in the fourth quarter of the image.

Get the last top point coordinates from the Top array 'T2 is a top point of the fifth finger

Find the top point T1 of the fourth finger by searching in the Top() for the top point with x-coordinate > Image width /2 and has the minimum distance from the last top point.

loop on the curve points between the two detected top points and calculate the gradient of the curve slope using $S = \frac{\Delta x}{\Delta y} = \frac{x2-x1}{y2-y1}$, where x1 is the x-coordinate of the current curve point, x2 is the x-coordinate of a curve point that stays five points away from x1 and y1 and y2 are the y-coordinates of the two curve points. All the S values between the two top points range between [-1, 1] except one region that exceeds the value (1.5).

DOI: https://doi.org/10.54216/FPA.190201
Received: December 10, 2024 Revised: February 05, 2025 Accepted: March 03, 2025

Check if $S \geq 1.5$ then calculate the average of x-coordinates (Avx) and the average of y-coordinates (Avy) for that part of the curve (the elbow point).

If the elbow point (Avx, Avy) is not on the hand curve, then search for the nearest curve point to it in eight directions and replace the nearest curve point with the elbow point.

If the elbow point in the fourth quarter of the image where its x-coordinate > width/2 and its y-coordinate > height/2, delete all the top and bottom points between T1 and T2.

Add the coordinates of the elbow point to the Bottom() and set a Boolean flag (F) to true.

Return Top() and Bottom().

Algorithm (3): The NoF Extraction

Input: Top(): an array of top points

Bottom(): an array of bottom points

Output: NoF: a number of opened fingers

Calculate the distance between each top and bottom point using dis equation and put the result in *h()*. 'h contains the heights of the five fingers.

Loop on the Top()

Check if* the top point is not the last point *then

$dis \leftarrow \sqrt{(Top(i).x - Bottom(i).x)^2 + (Top(i).y - Bottom(i).y)^2}$ **//i is an index of Top() and Bottom().**

Else

Check if* the number of bottom points < the number of top points *then

Calculate dis between the last top point and the last bottom point //connect the last top point with the previous bottom point

End if

End if

End loop

Step 2: Find the maximum value in h().

Calculate the ratio Ra using the following equation:

$Ra = \propto * max$ **// $\propto$ is a constant and equals to 0.49 for the testing dataset**

Extract the Number of opened fingers.

Initial counter NoF←0

***Loop* on h()**

***Check if* h(i) > Ra *then Increment* NoF**

End loop

Return NoF.

C. Classification Process

Algorithm (4) shows the main steps for digit classification.

Algorithm (4): The classification process

DOI: https://doi.org/10.54216/FPA.190201
Received: December 10, 2024 Revised: February 05, 2025 Accepted: March 03, 2025

Input: Cc(): the output of the chain coding process
Output: Cl: detected digit
Step 1: Check if the tested sample represents a zero sign.
Area← 8-directions boundary filling method
Ra1←0.009* image area
Ra2← 0.02* image area
Check if* Ra1 ≤ Area ≤ Ra2 *then
cl←0
Else
Step 2: Detect the top and bottom points.
Perform Algorithm (2)
Step 3: Calculate NoF
Perform Algorithm (3)
Check if* F=True *then
Check if* NoF=3 *and h(0) < Ra and h(1) < Ra then ' h(0)and h(1) are the heights of the first and the second fingers.
Cl←3
Else Check if* NoF=5 *then* Cl←5 *End if
End if
Else 'F equals False
Step 4: Check the locations of the opened fingers

***Check if* NoF=1 *then* Cl←1**

***Else Check if* NoF=2 *then* Cl←2**

***Else Check if* NoF=4 *then* Cl←4**

***Else Check if* NoF=3 and h(0)< Ra and h(1)>Ra and h(2)>Ra and he(3)>Ra *then* Cl←6**

***Else Check if* NoF=3 and h(0)> Ra and h(1)<Ra and h(2)>Ra and he(3)>Ra *then* Cl←7**

***Else Check if* NoF=3 and h(0)> Ra and h(1)>Ra and h(2)<Ra and he(3)>Ra *then* Cl←8**

***Else Check if* NoF=3 and h(0)> Ra and h(1)>Ra and h(2)>Ra and he(3)<Ra *then* Cl←9**

End if

End if

Step 5: Return Cl.

4. Experimental Results

To evaluate the performance of the designed model, the precision, recall, and F1 scores have been computed, which show the optimal system's performance. To obtain these quantitative measures, we initially compute the confusion matrix that is shown in figure 10. As explained in Section 2, the number of samples for each class is forty samples, representing 10% of the total number of the examined samples. These ratios are highlighted in blue in figure 10 and misclassified samples are represented in red. So, the results showed the system's high ability to recognize all the samples in the examined data set.

True Negative value (T_N) represents the total number of negative examples that the system classifies them accurately and True Positive value (T_P) represents the total number of positive examples that the system classifies them accurately. While False Positive value (F_P) represents the total number of actual negative examples that the system classifies them as positive and False Negative value (F_N) is the total number of actual positive examples that the system classifies them as negative [36]. From (T_P, T_N, F_P and F_N) values, we can obtain precision, recall, and F1 scores using (5-7) equations [37]. Table 1 shows the values of these measures for the testing dataset. The misclassification rate is the number of incorrect predictions divided by the total number of predictions.

$$\text{Precision (p)} = \text{T\_P}/(\text{T\_P} + \text{F\_P}) \quad (5)$$

$$\text{Recall (R)} = \text{T\_P}/\text{T\_P} + \text{F\_N} \quad (6)$$

$$\text{F1 score} = 2 * [(\text{P} * \text{R})/(\text{P} + \text{R})] \quad (7)$$

DOI: https://doi.org/10.54216/FPA.190201
Received: December 10, 2024 Revised: February 05, 2025 Accepted: March 03, 2025

	Training Set										
Target / Output	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9	
Class 0	40 100%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	40/400 10% 0%
Class 1	0 0%	40 100%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	40/400 10% 0%
Class 2	0 0%	0 0%	40 100%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	40/400 10% 0%
Class 3	0 0%	0 0%	0 0%	40 100%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	40/400 10% 0%
Class 4	0 0%	0 0%	0 0%	0 0%	40 100%	0 0%	0 0%	0 0%	0 0%	0 0%	40/400 10% 0%
Class 5	0 0%	0 0%	0 0%	0 0%	0 0%	40 100%	0 0%	0 0%	0 0%	0 0%	40/400 10% 0%
Class 6	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	40 100%	0 0%	0 0%	0 0%	40/400 10% 0%
Class 7	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	40 100%	0 0%	0 0%	40/400 10% 0%
Class 8	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	40 100%	0 0%	40/400 10% 0%
Class 9	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	40 100%	40/400 10% 0%
	40/400 10% 0%	40/400 10% 0%	40/400 10% 0%	40/400 10% 0%	40/400 10% 0%	40/400 10% 0%	40/400 10% 0%	40/400 10% 0%	40/400 10% 0%	40/400 10% 0%	400/400 100% 0%

Figure 10. Confusion matrix of 10 single-hand ASL digits

Table 1: (P, R, and F1) score for the testing data set

Class	F-N	F-P	T-N	T-P	Precision	Recall/ Sensitivity	F-score
0	**0**	**0**	**360**	**40**	**1**	**1**	**1**
1	**0**	**0**	**360**	**40**	**1**	**1**	**1**
2	**0**	**0**	**360**	**40**	**1**	**1**	**1**
3	**0**	**0**	**360**	**40**	**1**	**1**	**1**
4	**0**	**0**	**360**	**40**	**1**	**1**	**1**
5	**0**	**0**	**360**	**40**	**1**	**1**	**1**
6	**0**	**0**	**360**	**40**	**1**	**1**	**1**
7	**0**	**0**	**360**	**40**	**1**	**1**	**1**
8	**0**	**0**	**360**	**40**	**1**	**1**	**1**
9	**0**	**0**	**360**	**40**	**1**	**1**	**1**
Accuracy	**1**						
Misclassification Rate	**0**						
Average of F1 Scores	**1**						

DOI: https://doi.org/10.54216/FPA.190201
Received: December 10, 2024 Revised: February 05, 2025 Accepted: March 03, 2025

Another test of the system's performance is the time factor. The test showed the high execution speed of the system for classification of the digits as explained in table 2.

Table 2: Average computational time for each executed process and the total time

Process	Time in (milliseconds)
Image Loading	**0.0298**
Global Thresholding	**0.0084**
Remove the Surrounding Aura / Shadow	**0.0006**
Blots Removing	**0.0049**
Gaps Removing	**0.0128**
ROI Allocation	**0.0013**
Edge Detection & Removal of Points Having D4 Neighbors	**0.0026**
Chain Coding	**0.0017**
Feature Extraction & classification	**0.0031**
Total Time	**0.0652**

5. Conclusion

A novel recognition system for a numerical ASL was adopted in this research. Many image-processing techniques were used to detect the boundary of the hand's fingers. For features extraction, the gap area in the Zero digit was calculated to distinguish this digit where it should lie between (0.009 and 0.02) of the total area of the tested image and for classifying other digits, the number of the opened fingers was obtained depending on the heights of the fingers. Although all 3, 6, 7, 8, and 9 digits have the same number of opened fingers, the system recognized them perfectly relying on the sequence in which the opened fingers appear. The tests showed the speed and high performance of the system to distinguish between the digits where the obtained precision and F-score percentages reached the desired optimal value (100%) and the total execution time was 0.0652 milliseconds. In future work, the ASL alphabet will be taken into consideration as well as applying the system to other types of SLs or datasets.

Funding: "This research is funded by its authors."

Conflicts of Interest: "None."

References

[1] N. F. Attia, M. T. F. S. Ahmed, and M. A. M. Alshewimy, "Efficient deep learning models based on tension techniques for sign language recognition," *Intelligent Systems with Applications*, vol. 20, p. 200284, 2023. doi: 10.1016/j.iswa.2023.200284.

[2] S. Wilcox and J. Peyton, "American Sign Language as a foreign language," *CAL. Dig.*, pp. 159-160, 1999.

[3] A. Sultan, W. Makram, M. Kayed, and A. A. Ali, "Sign language identification and recognition: A comparative study," *Open Computer Science*, vol. 12, no. 1, pp. 191-210, 2022. doi: 10.1515/comp-2022-0240.

[4] Z. Katılmış and C. Karakuzu, "Double handed dynamic Turkish Sign Language recognition using Leap Motion with meta learning approach," *Expert Systems with Applications*, vol. 228, p. 120453, 2023. doi: 10.1016/j.eswa.2023.120453.

[5] A. Sharma, N. Sharma, Y. Saxena, et al., "Benchmarking deep neural network approaches for Indian Sign Language recognition," *Neural Computing & Applications*, vol. 33, no. 12, pp. 6685-6696, 2021. doi: 10.1007/s00521-020-05448-8.

DOI: https://doi.org/10.54216/FPA.190201
Received: December 10, 2024 Revised: February 05, 2025 Accepted: March 03, 2025

[6] A. Wadhawan and P. Kumar, "Deep learning-based sign language recognition system for static signs," *Neural Computing & Applications*, vol. 32, no. 12, pp. 7957-7968, 2020. doi: 10.1007/s00521-019-04691-y.

[7] D. Bragg, O. Koller, M. Bellard, L. Berke, P. Boudreault, A. Braffort, et al., "Sign language recognition, generation, and translation: An interdisciplinary perspective," in *21st International ACM SIGACCESS Conference on Computers and Accessibility*, pp. 16-31, 2019. doi: 10.1145/3308561.3353774.

[8] K. Wangchuk, P. Riyamongkol, and R. Waranusast, "Real-time Bhutanese Sign Language digits recognition system using Convolutional Neural Network," *ICT Express*, vol. 7, no. 2, pp. 215-220, 2021. doi: 10.1016/j.icte.2020.08.002.

[9] H. I. Mohammed and J. Waleed, "Hand gesture recognition using a convolutional neural network for Arabic sign language," *AIP Conference Proceedings*, vol. 2475, no. 1, p. 070012, 2023. doi: 10.1063/5.0104256.

[10] S. Sharma and S. Singh, "Vision-based hand gesture recognition using deep learning for the interpretation of sign language," *Expert Systems with Applications*, vol. 182, p. 115657, 2021. doi: 10.1016/j.eswa.2021.115657.

[11] A. Orbay and L. Akarun, "Neural sign language translation by learning tokenization," in *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*, pp. 222-228, 2020. doi: 10.1109/FG47880.2020.00002.

[12] W. Abdul, M. Alsulaiman, S. U. Amin, M. Faisal, G. Muhammad, E. R. Albogamy, M. A. Bencherif, and H. Ghaleb, "Intelligent real-time Arabic sign language classification using attention-based inception and BiLSTM," *Computers & Electrical Engineering*, vol. 95, p. 107395, 2021. doi: 10.1016/j.compeleceng.2021.107395.

[13] Q. Zhang, D. Wang, R. Zhao, and Y. Yu, "MyoSign: Enabling end-to-end sign language recognition with wearables," in *Proceedings of the 24th International Conference on Intelligent User Interfaces*, pp. 650-660, 2019. doi: 10.1145/3301275.3302296.

[14] M. S. Alom, M. J. Hasan, and M. F. Wahid, "Digit recognition in sign language based on convolutional neural network and support vector machine," in *2019 International Conference on Sustainable Technologies for Industry 4.0 (STI)*, pp. 1-5, 2019. doi: 10.1109/STI47673.2019.9067999.

[15] B. W. Salim and S. R. M. Zeebaree, "Kurdish Sign Language recognition based on transfer learning," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 11, no. 6s, pp. 232-245, 2023. [Online]. Available: https://ijisae.org/index.php/IJISAE/article/view/2843.

[16] B. Fang, J. Co, and M. Zhang, "DeepASL: Enabling ubiquitous and non-intrusive word and sentence-level sign language translation," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, Delft, The Netherlands, Nov. 6–8, 2017, p. 13. doi: 10.1145/3131672.3131693.

[17] H. M. Hammed, O. M. H. Almiahi, and O. Shauchuk, "Detection of anthropogenic objects based on the spatial characteristics of their contour in aerial image," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 23, no. 1, pp. 206-215, 2021. doi: 10.11591/ijeecs.v23.i1.pp206-215.

[18] A. Hussain, S. U. Amin, M. Fayaz, and S. Seo, "An efficient and robust hand gesture recognition system of sign language employing finetuned Inception-V3 and EfficientNet-B0 network," *Computer Systems Science and Engineering*, vol. 46, no. 3, pp. 3509-3525, 2023. doi: 10.32604/csse.2023.037258.

[19] M. A. Hameed, S. B. Al-Khoja, and R. R. Ismail, "Small binary codebook design for image compression depending on rotating blocks," *Iraqi Journal of Science*, vol. 62, no. 10, pp. 3719-3723, 2021. doi: 10.24996/ijs.2021.62.10.30.

[20] M. G. Abdul-Haleem, L. E. George, and A. G. Abdul-Haleem, "A system for rapid localization and intensity evaluation of saponins accumulation in plant tissues," *Journal of Theoretical and Applied Information Technology*, vol. 97, no. 3, pp. 1030-1040, 2019. [Online]. Available: https://www.jatit.org/volumes/Vol97No3/27Vol97No3.pdf.

[21] A. Shihab, H. Salah, and M. Mocanu, "Detection and diagnosis of skin cancer based on K-means cluster and convolutional neural network," in *2021 23rd International Conference on Control Systems and Computer Science (CSCS)*, pp. 143-150, Bucharest, Romania, 2021. doi: 10.1109/CSCS52396.2021.00031.

DOI: https://doi.org/10.54216/FPA.190201
Received: December 10, 2024 Revised: February 05, 2025 Accepted: March 03, 2025

[22] L. Abualigah, N. K. Al-Okbi, E. M. Awwad, et al., "Correction: Boosted Aquila Arithmetic Optimization Algorithm for multi-level thresholding image segmentation," *Evolving Systems*, 2024. doi: 10.1007/s12530-024-09576-7.

[23] M. A. Rajab and L. E. George, "Car logo image extraction and recognition using K-medoids, Daubechies wavelets, and DCT transforms," *Iraqi Journal of Science*, vol. 65, no. 1, pp. 431-442, 2024. doi: 10.24996/ijs.2024.65.1.35.

[24] L. Wang, Q. Meng, H. Wang, et al., "Digital image processing realized by memristor-based technologies," *Discover Nano*, vol. 18, p. 120, 2023. doi: 10.1186/s11671-023-03901-w.

[25] I. Mahmud, T. Tabassum, M. P. Uddin, E. Ali, A. M. Nitu, and M. I. Afjal, "Efficient noise reduction and HOG feature extraction for sign language recognition," in *2018 International Conference on Advancement in Electrical and Electronic Engineering (ICAEEE)*, pp. 1-4, Gazipur, Bangladesh, 2018. doi: 10.1109/ICAEEE.2018.8642983.

[26] A. Sharma, A. Mittal, S. Singh, and V. Awatramani, "Hand gesture recognition using image processing and feature extraction techniques," *Procedia Computer Science*, vol. 173, pp. 181-190, 2020. doi: 10.1016/j.procs.2020.06.022.

[27] I. A. Adeyanju, O. O. Bello, and M. A. Azeez, "Development of an American sign language recognition system using Canny edge and histogram of oriented gradient," *Nigerian Journal of Technological Development*, vol. 19, no. 3, pp. 195-205, 2022. doi: 10.4314/njtd.v19i3.2.

[28] S. Lahoti, S. Kayal, S. Kumbhare, I. Suradkar, and V. Pawar, "Android based American sign language recognition system with skin segmentation and SVM," in *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1-6, Bengaluru, India, 2018. doi: 10.1109/ICCCNT.2018.8493838.

[29] C. M. Jin, Z. Omar, and M. H. Jaward, "A mobile application of American sign language translation via image processing algorithms," in *2016 IEEE Region 10 Symposium (TENSYMP)*, pp. 104-109, Bali, Indonesia, 2016. doi: 10.1109/TENCONSpring.2016.7519386.

[30] I. Mahfudi, M. Sarosa, R. A. Asmara, and M. A. Gustalika, "Indonesian sign language number recognition using SIFT algorithm," in *IOP Conference Series: Materials Science and Engineering*, vol. 336, no. 1, p. 012010, 2018. doi: 10.1088/1757-899X/336/1/012010.

[31] A. Mavi, "A new dataset and proposed convolutional neural network architecture for classification of American sign language digits," *arXiv preprint arXiv:2011.08927*, 2020.

[32] W. K. Mutlag, S. K. Ali, Z. M. Aydam, and B. H. Taher, "Feature extraction methods: A review," *Journal of Physics: Conference Series*, vol. 1591, no. 1, p. 012028, 2020. doi: 10.1088/1742-6596/1591/1/012028.

[33] I. H. Abbas and M. Q. Ismael, "Automated pavement distress detection using image processing techniques," *Engineering, Technology & Applied Science Research*, vol. 11, no. 5, pp. 7702-7708, 2021. doi: 10.48084/etasr.4450.

[34] M. R. Naeemah, "Textural analysis of liver tumor using watershed segmentation based on statistical and geometrical features," *Iraqi Journal of Science*, vol. 60, no. 8, pp. 1877-1887, 2019. doi: 10.24996/ijs.2019.60.8.25.

[35] N. M. Hashem, H. Kh. Abbas, and H. J. Mohamad, "Detection and recognition of car plates in parking lots at Baghdad University," *Iraqi Journal of Science*, vol. 64, no. 2, pp. 1018-1029, 2023. doi: 10.24996/ijs.2023.64.2.43.

[36] A. Kulkarni, D. Chong, and F. A. Batarseh, "Foundations of data imbalance and solutions for a data democracy," in F. A. Batarseh and R. Yang, Eds., *Data Democracy*, Academic Press, pp. 83-106, 2020. doi: 10.1016/B978-0-12-818366-3.00005-8.

[37] R. K. Pathan, M. Biswas, S. Yasmin, et al., "Sign language recognition using the fusion of image and hand landmarks through multi-headed convolutional neural network," *Scientific Reports*, vol. 13, no. 1, p. 16975, 2023. doi: 10.1038/s41598-023-43852-x.

DOI: https://doi.org/10.54216/FPA.190201
Received: December 10, 2024 Revised: February 05, 2025 Accepted: March 03, 2025