Real Time Sign Language Processing System

Dibyabiva Seth(✉), Anindita Ghosh, Ariruna Dasgupta, and Asoke Nath

Department of Computer Science, St. Xavier's College (Autonomous), Kolkata, India
meetdseth@gmail.com, anindital27@gmail.com, dasguptaariruna@gmail.com, asokejoyl@gmail.com

Abstract. A communication gap has always existed between Sign Languages and other Natural Languages. This paper aims to build a real-time autonomous system that can help bridge this communication gap. The present system captures the gestures using a webcam and recognizes the gesture being shown, mapping it to the corresponding English Letter, Numeric Digit and Special Characters. The authors have proposed American Sign Language with some minor modifications. The present form of ASL can be used to recognize all alphabets (A–Z), all numerals (0–9), Backspace, Blank Space. Some Special Characters have been included as well. The present system is built to recognize the finger-spelling component of the American Sign Language (ASL), and can be extended to recognize other sign languages as well.

Keywords: Sign language · American sign language · Finger spelling · Perceptrons

1 Introduction

Sign language is used across the world to help bridge the communication gap for the hearing or speech impaired. The American Sign Language (ASL) is a fully developed natural language and is one of the world's many sign languages. It is neither a simplified language, nor a derivative of the English Language [1]. The largest influence on the development of sign language in America was Thomas Hopkins Gallaudet, a congregational minister. Together with Laurent Clerc, a graduate of the school for the deaf in Paris, Gallaudet transformed the old French Sign language into a sign language that American students would understand better. This system of sign language had a system of grammar and signs to represent every word and is known as the Old Signed English. Later they modified it to a language which was free of all grammar and shortened the sentences down to key phrases. This language later became known as the ASL. The fingerspelling component of ASL [2] has been shown in Fig. 1.

The aim of this paper is to design a practically useful real time processing system for the ASL based on the following broad objectives. The system should:

- be able to recognize the finger spelling gestures of the ASL,
- be able to show the corresponding numerical digits or letters of the English alphabet and some Special characters on the output device,

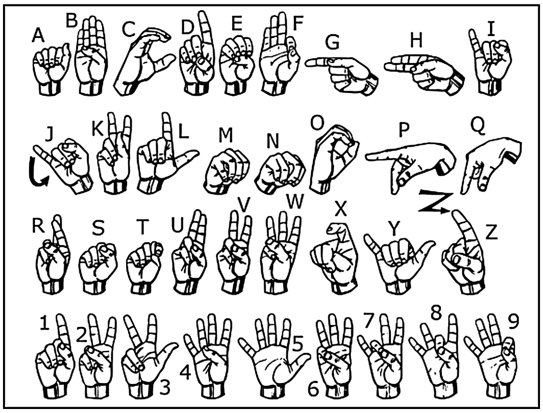


Fig. 1. The ASL fingerspelling gestures

- be able to show the corresponding sign language from a text file,
- be simple and easy to use,
- make use of already existing hardware components which are readily available,
- not use expensive hardware components,
- run on existing software environments.

1.1 Literature Review

Nachamai [3] has used the SIFT algorithm to recognize the English alphabets. Nachamai's method is space, size, illumination and rotation invariant. Pugeault and Bowden [4], on the other hand, have made use of a Microsoft Kinect device along with OpenNI+NITE framework for the acquisition, detection and tracking of the hand movements.

Vogler and Metaxas [5] have designed a framework to recognize isolated and continuous ASL sentences from three dimensional data. They have used Hidden Markov Models (HMMs) to recognize the hand movements. They have tested the framework on a vocabulary consisting of 53 signs.

In this paper, perceptrons have been used to identify the gestures of the ASL. A perceptron is a machine learning algorithm used primarily as a binary classifier. The system has been implemented to identify the 26 gestures of the letters and 9 gestures of the numeric digits of the English alphabet, along with the gestures for space and backspace. Perceptrons converge very fast, making them preferable for use in real-time.

The authors of [6] have implemented a Hand Gesture Recognition Library in MATLAB. They have used red LEDS mounted on a glove to track the hand movements. In this project, the red LEDs have been mounted near the webcam to better illuminate the hand, freeing the hand from the glove.

2 The Real Time Sign Language Processing System

The problem can be mainly divided into three modules, as follows:

- **Image Processing Module** – This module deals with the extraction of the hand gesture of the sign (foreground object) from the rest of the image (background). The extracted image is then scaled appropriately to predefined dimensions and sent to the Gesture Recognition Module.
- **Training Module** – This module forms the core of the Gesture Recognition Module. The purpose of this module is to train the program with a set of images of all the hand gestures used in a specific sign language. It needs to be executed only when one or more gestures need to be changed.
- **Gesture Recognition Module** – This module deals with the recognition of the hand gesture extracted by the Image Processing Module. It is pre-trained by the Training Module. If a sign is identified, it is mapped to the corresponding letter and displayed.
- **Gesture Mapping Module** – This module deals with conversion of a string of characters taken from a file or given by the user to sign language and speech.

Two sets of data need to be maintained.

- **Image Dataset** – This is the set of all the pre-clicked images of the gestures of a sign language. The original ASL signs [7] have been shown in Fig. 1, whereas the modified finger-spelling signs have been shown in Fig. 2. These images are used by the Training Module. The images are organized into folders, indexed as per the rule given below. There are multiple images of a single gesture. This is done to accommodate variations in the orientation of the gestures. The index (starting from 1) for a particular letter is named its "gesture index". For the English alphabets, the indexes are 1 through 26 accordingly, for space and backspace, the indexes are 27 and 28 respectively, for the digits 0 through 9, the indexes are 29 through 38 respectively and for the special characters the indexes 39 to 47 have been assigned. For instance, a path like "3/5.bmp" points to the 5th image of the gesture for letter 'C'; the index is calculated as: index of 'C' = (ASCII value of 'C') – (ASCII value of 'A') + 1 = 3.
- **Perceptrons** – The trained perceptrons are stored in simple text files, indexed following the same indexing procedure as above.

2.1 Algorithms

- **Image Processing Module** – The input is taken via the webcam. To make the extraction process easier, the webcam is surrounded by four red LEDs. The color ranges are then appropriately adjusted so that only the foreground object is detected.

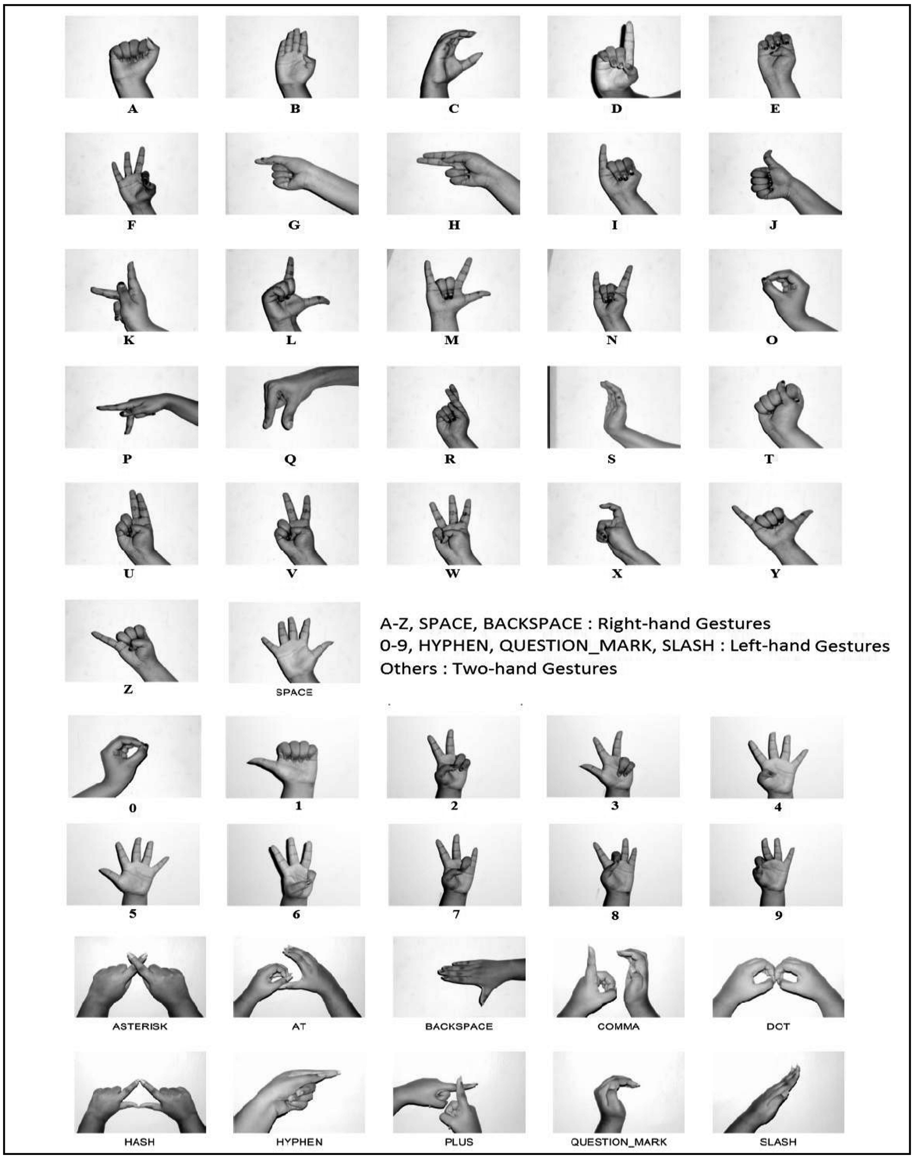

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
SPACE
A-Z, SPACE, BACKSPACE : Right-hand Gestures
0-9, HYPHEN, QUESTION_MARK, SLASH : Left-hand Gestures
Others : Two-hand Gestures
0
1
2
3
4
5
6
7
8
9
ASTERISK
AT
BACKSPACE
COMMA
DOT
HASH
HYPHEN
PLUS
QUESTION_MARK
SLASH

Fig. 2. Modified ASL

```
Algorithm Image_Processing_Module
    Step 1. Capture an image via webcam.
    Step 2. Scan through all the pixels of the image. If a pixel lies within the
            pre-defined pixel value range, make it white. Else make that pixel
            black.
    Step 3. Check the area of all the white connected regions in the image.
    Step 4. Keep the white region with the largest area, and remove the other
            white regions in the image.
    Step 5. If the area of the largest white region is less than a predefined
            AREA_THRESHOLD, then remove that white region also and stop.
            Else, keep that white region and it is the extracted gesture shown by
            the user. Send this extracted gesture as an input to the Gesture
            Recognition Module.
    End
```

- **Training Module** – This module uses the set of pre-clicked images of the different gestures to train the program. The output of this module is the set of perceptrons. There is a separate perceptron for each letter of the English Alphabet (one vs rest classification model).

```
Algorithm Training_Module:
    For each perceptron, do (recall that perceptrons are indexed according to the
  letters, starting from 1)
        a. For each gesture, do ($n$ is the number of pixels in an image,
           same for all images here)
            i. Convert the monochrome (0-1) image of the gesture to
               a vector
                    $A = [a_1, a_2, \dots, a_n]^T$,
               where the superscript $T$ stands for transpose.
            ii. Denote the perceptron trained so far by the vector
                    $W = [w_1, w_2, \dots, w_n]^T$.
               [$W$ is initialized to $\mathbf{0}_{n\times 1}$ in our case.]
            iii. Compute
                    $obtained\_result = A^T W = \sum_{i=1}^{n} a_i w_i$
            iv. If perceptron index = gesture index, then
               $actual\_result = 1$.
               Else $actual\_result = 0$.
            v. Compute $N = \|A\|_1$. Note that elements of $A$ are either
               0 or 1, so $\|A\|_1$ simply gives the number of 1's (white
               pixels) in $A$.
            vi. Update $W$ according to the relation
                $W = W + A * ((actual\_result - obtained\_result)/N)$.
        b. If perceptron satisfies the predefined MATCH_CRITERIA,
           then continue.
           Else repeat from (a).
    End
```

- **Gesture Recognition Module** – This module tests the extracted gesture and tests it with all the perceptrons. If a match is found, then the corresponding letter is displayed.

```
Algorithm Gesture_Recognition_Module
  Step 1. Repeat until all perceptrons have been tested
        a. Convert the monochrome (0-1) image of the gesture to a vector
               $A = [a_1, a_2, \dots, a_n]^T$.
        b. Denote the current perceptron by the vector
               $W = [w_1, w_2, \dots, w_n]^T$.
        c. Compute
               $obtained\_result = A^T W = \sum_{i=1}^{n} a_i w_i$
        d. If obtained_result > MATCH_THRESHOLD, then a match has
           been found and it is mapped to the corresponding letter and the
           search is stopped.
           Else continue.
  Step 2. If no match has been found, then the gesture is not a valid gesture
          and no output is provided.
End
```

3 Results and Discussions

A typical result of the image processing module is given in Fig. 3. It can be seen that the foreground object (hand gesture) has been successfully extracted from the background.

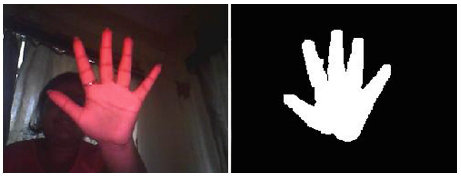


Fig. 3. Foreground Object Extraction

To obtain the system accuracy, each character has been tested 50 times and the accuracy is calculated by the rule.

$$accuracy = \frac{Total\ no.\ of\ correct\ outputs}{Total\ no.\ of\ tries}$$

Figure 4 shows the accuracy of the algorithm where each gesture is shown 50 times consecutively and the number of times the correct output obtained is recorded. The calculated overall accuracy is approximately 91.2%.

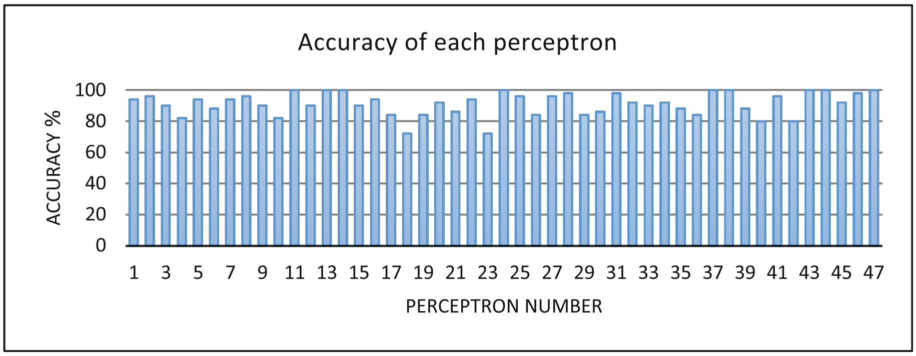
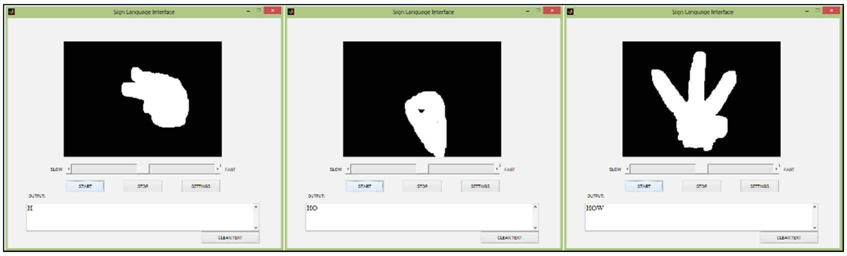


Fig. 4. Bar graph showing number of correct outputs.

The real-time working of the "Sign Language Interface" and the "Text to Sign Converter" have been shown in Figs. 5 and 6 respectively. The word "How" was tested and the results were recorded.



H

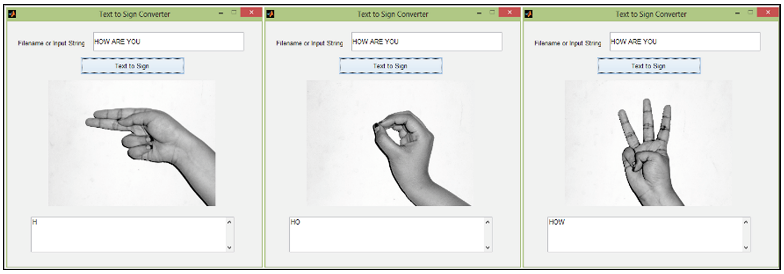
HO

HOW

Fig. 5. Sign Language Interface showing the word 'HOW'.

4 Limitations

- This system is limited to the finger spelling method, which is not the most practical method for communication for deaf people. It would generally take too much time to finger spell a few sentences. So mainly this should be used to express proper nouns or short sentences conveying the meaning properly.



Filename or Input String: HOW ARE YOU

H

Filename or Input String: HOW ARE YOU

HO

Filename or Input String: HOW ARE YOU

HOW

Fig. 6. Text to Sign Converter showing the word 'HOW'.

- Similar hand orientations are hard to distinguish. For instance, M and N have similar gestures, which make it difficult to distinguish between them. To overcome this difficulty, some modifications were made to these gestures.
- There are certain gestures which involve hand movements, as shown for J and Z. This system deals with image processing and hence there is no provision for video capturing to show these hand movements. Hence, such gestures have been modified accordingly.

References

1. History of ASL, http://iml.jou.ufl.edu/projects/fall05/rosen/history.html
2. Finger Spelling Alphabet, http://www.fingerspellingalphabet.com
3. Nachamai, M.: Alphabet recognition of american sign language: a hand gesture recognition approach using sift algorithm. Int. J. Artif. Intell. Appl. (IJAIA) **4**(1), 105–115 (2013)
4. Pugeault, N., Bowden, R.: Spelling it out: real-time ASL fingerspelling recognition. In: Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on 2011 Nov 6, pp. 1114–1119. IEEE (2011)
5. Vogler, C., Metaxas, D.: ASL recognition based on a coupling between HMMs and 3D motion analysis. In: Sixth International Conference on Computer Vision, pp. 363–369. IEEE, January 1998
6. Paul, J.F., Seth, D., Paul, C., Dastidar, J.G.: Hand gesture recognition library. Int. J. Sci. Appl. Inf. Technol. **3**(2), 44–50 (2014)
7. ASL University: Fingerspelling. http://www.lifeprint.com/asl101/fingerspelling
8. MathWorks: text-to-speech. http://in.mathworks.com/matlabcentral/fileexchange/18091-text-to-speech