SIGN LANGUAGE RECOGNITION USING THINNING ALGORITHM

S. N. Omkar[1] and M. Monisha[2]
[1]*Department of Aerospace Engineering, Indian Institute of Science, Bangalore, Karnataka, India*
E-mail: omkar@aero.iisc.ernet.in
[2]*Department of Information Technology, National Institute of Technology Karnataka, Surathkal, India*
E-mail: monisha3107@gmail.com

Abstract
In the recent years many approaches have been made that uses computer vision algorithms to interpret sign language. This endeavour is yet another approach to accomplish interpretation of human hand gestures. The first step of this work is background subtraction which achieved by the Euclidean distance threshold method. Thinning algorithm is then applied to obtain a thinned image of the human hand for further analysis. The different feature points which include terminating points and curved edges are extracted for the recognition of the different signs. The input for the project is taken from video data of a human hand gesturing all the signs of the American Sign Language.

Keywords:
Hand Gesture Recognition, Sign Language, Pre-Processing, Thinning Algorithm, Feature Points Extraction

1. INTRODUCTION

Vision based automatic hand gesture recognition has many applications such as human computer interaction, sign language interpretation, remote control and affective computing. There are many challenges with respect to accuracy and usefulness of the hand gesture recognition systems. Implementation of image-based gesture recognition using different methodologies pose different concerns as the result may vary from camera to camera. Various approaches have been used for tackling this issue including the use of artificial neural network models [1], [2]. In this research work a new method is used to interpret the signs using thinning algorithm.

Thinning is a morphological operation that is used to remove selected foreground pixels from binary images. It is particularly used for obtaining a skeletal image by reducing all lines to single pixel thickness. The earlier methods required glove-based devices that would measure the position and joint angles of the hand. These devices were complicated and difficult to handle. Thus it led to introducing new non-intrusive, vision-based methods for interpretation of hand gestures. Certain other systems used markers and hand tracking is carried out based on colour and appearance [3]. Pre-defined databases and visual memory systems are used for pattern matching based recognition of gestures [4]. Boundary tracing is another approach for finger movement detection [5]. In this work the end points of the fingers in hand gestures are used to determine the terminating points. The feature points which include terminating points and curved edges are computed from the thinned image using a simple algorithm. Hither to thinning algorithm has been used for pattern recognition and character recognition [6]. Here thinning algorithm is used for modelling the human hand to obtain an image that makes the further processes easier.

The paper is organised as follows. Section 1 contains the applications and introduction. The proposed method is discussed in section 2. Section 3 presents the method used for background subtraction. Pre-processing techniques and morphological operations performed are discussed in section 4. The thinning algorithm is described in section 5. Section 6 describes the extraction of terminating points and edges. Section 7 explains sign interpretation from the images using different distance calculations and efficiency of the method used. The last section concludes this work.

The schematic representation of this work is shown in Fig.1.

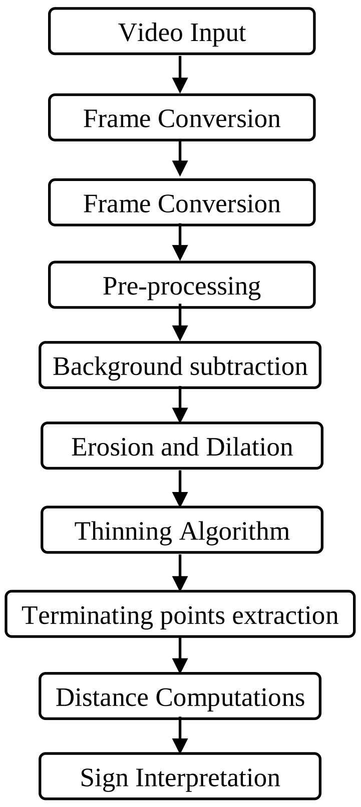


Fig.1. Schematic Representation of the proposed method

2. PROPOSED METHOD

The orientation of the human hand and the position of the different fingers play a very important role in identifying each sign as the difference between two signs may be only in the orientation while the positioning of the fingers remains same. The tips of the fingers are the terminating points which are extracted. A maximum of five end points are obtained for every sign. In cases where there are no fingers opened, the curved edges are computed instead. The points thus extracted are used to calculate the distances between them. By comparing the different distances the required interpretation for the gesture is obtained. This is achieved by having a good knowledge of the signs and their representations. This proposed algorithm is implemented using MATLAB 7.6.0 (R2008a) on an Intel dual core processor, 2 GB ram and Windows Vista SP2 system.

In the first stage the video sequence is obtained using a highly sensitive SONY Cyber Shot Digital Video Camera having 8.0 million effective pixels and a 1/2.5-in.CCD image sensor which produces a NTSC and PAL output. The video is taken with a frame rate of 25 fps. This video sequence is then converted into frames which are stored as images in the computer memory. The frames thus acquired are pre-processed to remove discrepancies arising on account of factors like lighting and environment which affects the quality of the video. Firstly the contrasts of the frames are improved by using the *imadjust* command. Next the small amount of salt and pepper noise is removed by applying the median filter function. Then background subtraction is done which yields in the isolation of the foreground object which in this case is the human hand. Morphological operations like dilation and erosion are performed next with the help of a structuring element in order to further process the images. At this stage the thinning algorithm is applied to produce the thinned image of the hand. Finally, from extracted terminating points and curved edges, distances are computed to recognise the specific sign.

The results of pre-processing are shown in Fig.2.

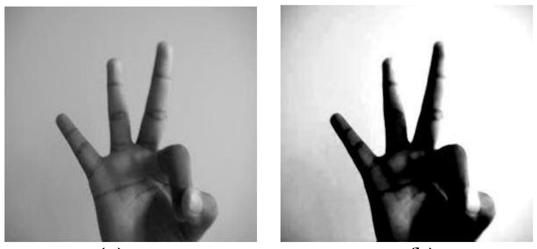

(a) (b)

Fig.2. Pre-processing of images
(a) Input Frame, (b) Output Frame after pre-processing

The background subtraction algorithm is illustrated in the following section.

3. EUCLEDIAN DISTANCE THRESHOLD METHOD

The method used for background subtraction is Euclidian distance threshold method where in, based on the difference between the computed Euclidean distance and the pre-set threshold value, the pixels are set in the foreground image. This algorithm is explained using MATLAB commands.

The following are the various steps involved in the algorithm:

Step 1: Set the first image as the background image.

Step 2: The background image is then subtracted from the next frame read.

Step 3: Separate the RGB components for each pixel in the background and foreground images for easy computation.

bg_r = imbg(:,:,1); bg_g = imbg(:,:,2); bg_b = imbg(:,:,3);
fr_r = imfr(:,:,1); fr_g = imfr(:,:,2); fr_b = imfr(:,:,3);

Step 4: The formula for calculating the Euclidean distance is:

$$E = \sqrt{(bg\_r - fr\_r)^2 + (bg\_g - fr\_g)^2 + (bg\_b - fr\_b)^2} \quad (1)$$

Step 5: Set a threshold value = T.

Step 6: If the value of E is greater than T, then the pixel intensity (p) value is set as 255 otherwise zero.

$$p = \begin{cases} 255, & E > T \\ 0, & otherwise \end{cases}$$

This resulted in an image in which the foreground object is isolated from the background.

Step 7: This process is continued for all the frames of the video sequence.

Fig.3 shows the results of background subtraction algorithm on an input frame.

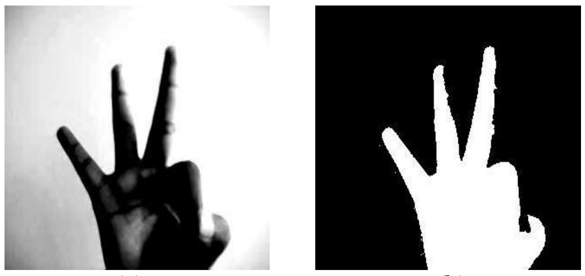

(a) (b)

Fig.3. Results of background subtraction
(a) Input frame, (b) Silhouette of human hand

The following section explains the morphological operations used in the proposed method.

4. MORPHOLOGICAL OPERATIONS

Two morphological operations - dilation and erosion - are then carried out for further cleaning of the silhouette images. Small extraneous pixels not part of the human hand are removed by erosion and the connectedness is maintained by dilation which adds pixels to the object boundaries.

The input for a morphological operation is a binary image where the pixel values are either 0 or 1 (black or white). So the output from the background subtraction is converted into a binary image using the *im2bw* command. In a morphological operation, each pixel in the input image is compared with its neighbours to find the output image. The size and shape of the neighbourhood must be decided first. This is done with the help of a structuring element obtained by the *strel* function. For this purpose a 3x3 square element is used as the structuring element given by *se=strel('square',3).* Erosion and dilation are performed using the MATLAB commands *imerode* and *imdilate* respectively. These functions take two arguments; one the input image and the other the structuring element.

The mathematical expressions for the above processes are:

a) Dilation of A by B expressed as,

$$A \oplus B = \cup A_x, \qquad x \in B \quad (2)$$

where, A is the input image and B is the structuring element. This means that for every pixel x ϵ B, A is translated by the corresponding pixel coordinates. Then the union of all those translations is taken.

And

b) Erosion of A by B expressed as,

$$A \ominus B = \{w : B \subseteq A\} \qquad (3)$$

where, A is the input image and B is the structuring element. In other words the erosion of A by B consists of all points w = (x, y) for which B_w is in A.

Thus a perfect silhouette of the human hand representing the sign is obtained at the end of this stage. Fig.4 shows the effects of dilation and erosion on a silhouette image.

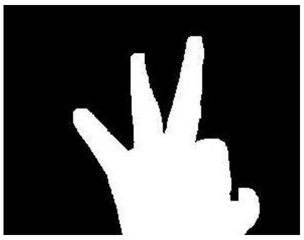


Fig.4. Output after dilation and erosion.

Thinning algorithm is described in the next section.

5. THINNING ALGORITHM

Thinning algorithm preserves the topology and shape of the original image i.e. the significant feature essential for object recognition or classification by eliminating most of the original foreground pixels. This is done by iteratively deleting pixels inside the shape to shrink it without shortening it or breaking it apart [7].

Thinning operation takes at a time two inputs, one the input image, which maybe either binary or greyscale and the other the structuring element. In this algorithm the origin of the structuring element is translated to every possible pixel position within the image. At each position the element is compared to the underlying image. If the structuring element and the image exactly match the image pixel underneath the origin of the structuring element is set to zero (background). If they do not match it is left unchanged. The number of pixels added or deleted is determined by the structuring element.

A flat linear structuring element given by the command *strel('line', 6, 90)* where 6 denotes the length and 90 denotes the angle (in degrees) of the line is considered for the thinning operation. The length is approximately the distance between the centres of the structuring element members at opposite ends of the line. Fig.5 shows the structuring elements used for morphological thinning. At each iteration, the image is first thinned by the left hand structuring element and then by the right hand one and then with the remaining six *90°* rotations of the two elements.

The thinning operation is achieved by the hit-and-miss transform [8]. The thinning of an image A by a structuring element B is given by,

$$\text{thin}(A, B) = A - \text{hit and miss}(A - B) \qquad (4)$$

0	0	0
0	1	0
1	1	1

0	0	0
1	1	0
0	1	0

Fig.5. Structuring elements used in morphological thinning

This process is repeated until the resultant thinned image remains same. Generally, the origin of the structuring element is at the centre.

The thinning algorithm comprises of:

Step 1: Partitioning the video frame into two distinct subfields in a checkerboard pattern.

Step 2: Delete the pixel p from the first subfield if and only if the conditions (5), (6), and (7) are satisfied.

$$X_H(p) = 1 \qquad (5)$$

$$X_H(p) = \sum_{i-1}^{4} b_i$$

where,

$$b_i = \begin{cases} 1 & \text{if } X_{2i-1} = 0 \text{ and } (X_{2i} = 1 \text{ or } X_{2i+1} = 1) \\ 0 & \text{otherwise} \end{cases}$$

X_1, X_2,...X_8 are the values of the eight neighbours of *p*, starting with the east neighbour and are numbered in counter-clockwise order.

$$2 \le \min\{n_1(p), n_2(p)\} \le 3 \qquad (6)$$

where,

$$n_1(p) = \sum_{k=1}^{4} X_{2k-1} \vee X_{2k}$$

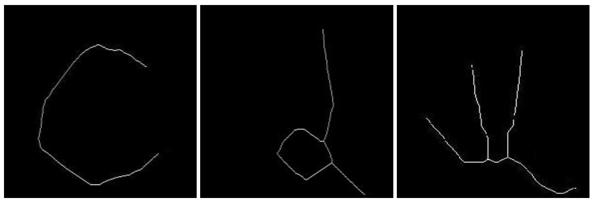
$$n_2(p) = \sum_{k=1}^{4} X_{2k} \vee X_{2k+1}$$

$$(X_2 \vee X_3 \vee \overline{X_8}) \vee \overline{X}_1 = 0 \qquad (7)$$

Step 3: Deleting the pixel p from the second subfield if and only if the conditions (5), (6), and (8) are satisfied.

$$(X_6 \vee X_7 \vee \overline{X_4}) \vee \overline{X}_5 = 0 \qquad (8)$$

The step 1 and step 2 together form a single iteration of the thinning algorithm. Here, an infinite number of iterations (n = ∞) is specified to get the thinned image. Fig.6 shows the resultant thinned images for a few hand signs.



(a) (b) (c)

Fig.6. Results of thinning for different hand gestures (a) Letter C (b) Letter D (c) Number 9

The following section describes the algorithm used for extracting the terminating points and curved edges.

6. FEATURE POINTS EXTRACTION

The terminating points are the tips of the fingers in the thinned image of the human hand. These are extracted using the elimination rules shown below in Fig.7. Here a terminating point is defined as a pixel with a single, eight-connected neighbour [9].

0	0	0
0	1	0
0	0	1

0	0	0
0	1	0
0	1	0

0	0	0
0	1	0
1	0	0

0	0	0
1	1	0
0	0	0

1	0	0
0	1	0
0	0	0

0	1	0
0	1	0
0	0	0

0	0	1
0	1	0
0	0	0

0	0	0
0	1	1
0	0	0

Fig.7. Elimination rules for end points

The following steps are involved in the algorithm used for extracting the terminating points.

Step 1: For every pixel in the image, check for the surrounding eight pixels.

Step 2: For a pixel whose value p = 1, if only one of the surrounding pixels has the value p = 1, then save that pixel as a terminating point.

Step 3: Otherwise ignore it and repeat steps one and two for the next pixel.

Step 4: This is continued till all the pixels are checked.

Elimination rules for extracting curved edges are shown in Fig.8.

0	0	1
0	1	0
0	0	1

0	1	0
0	1	0
0	1	0

0	0	1
0	1	0
0	1	0

0	0	1
1	1	0
0	0	0

1	0	0
0	1	1
0	0	0

1	0	0
0	1	0
0	1	0

1	0	0
0	1	0
0	0	1

Fig.8. Elimination rules for curved edges

The following steps are involved in the algorithm used for extracting the curved edges.

Step 1: For every pixel in the image, check for the surrounding eight pixels.

Step 2: For a pixel whose value p = 1, if two of the surrounding pixels have the value p = 1 and it matches any one of the diagram shown above, then save that pixel as a curved edge.

Step 3: Otherwise ignore it and repeat steps one and two for the next pixel.

Step 4: This is continued till all the pixels are checked.

After the feature points are extracted the distance and efficiency computations for the images are discussed.

7. DISTANCES COMPUTATIONS AND EFFICIENCY

Once the curved points and end points are obtained, distances between them are computed. By comparing the distances and specifying the limits for each sign, the particular sign is recognised. The efficiency of the proposed method is calculated for five different video sets. Each of them contains all the signs for numbers (1 to10) and alphabets (A to Z) of the American Sign Language.

A correct result (CR) is obtained when a sign is recognized correctly and an incorrect result is obtained (IR) when the sign is recognised incorrectly or is unidentifiable.

The efficiency is computed using the formula,

$$\text{Efficiency} = CR / (IR + CR) \times 100 \qquad (9)$$

Efficiency calculations are done separately and shown in the following Table.1 and 2 for numbers and alphabets respectively.

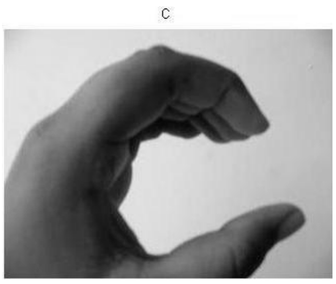
Table.1. Efficiency calculation for numbers

Input Video	CR	IR	CR/(IR+CR)	Efficiency (%)
Video 1	9	1	0.9	90
Video 2	10	0	1.0	100
Video 3	10	0	1.0	100
Video 4	9	1	0.9	90
Video 5	10	0	1.0	100

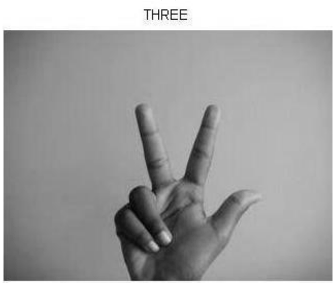
Table.2. Efficiency calculation for alphabets

Input Video	CR	IR	CR/(IR+CR)	Efficiency (%)
Video 1	26	0	1.000	100.00
Video 2	23	3	0.8846	88.46
Video 3	24	2	0.9231	92.31
Video 4	26	0	1.0000	100.00
Video 5	25	1	0.9615	96.15

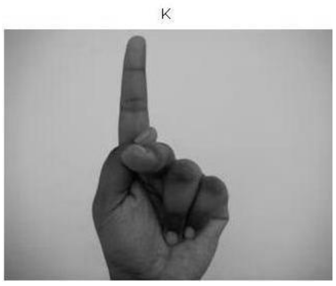
The output for a few signs is shown in Fig.9.



(a)



(b)



(c)

Fig.9. (a) Letter C (b) Number 3 (c) Letter K

Thus the signs of the American Sign Language are recognised using the proposed method with high accuracy.

8. CONCLUSION AND FUTURE WORK

Thinning algorithm is used to create the thinned image of the human hand for easy recognition of the represented gesture. All the signs used to represent alphabets and numbers are recognised using the proposed method. The video data is taken mainly focusing the hand and a maximum of five end points and a few curved edges is considered for computation. The efficiency of the algorithm is computed by applying the algorithm for different video sequences and the method was found to be accurate. In future this work can be further developed for translating the recognized signs into continuous text or speech. Recognizing gestures that involve motion of hands will be a tough challenge.

REFERENCES

[1] Jamie Lynn Boeheim, "*Human Activity Recognition Using Limb Component Extraction*", March 2008.

[2] Tin Hninn and Hninn Maung,"Real-Time Hand Tracking and Gesture Recognition System Using Neural Networks", *World Academy of Science, Engineering and Technology*, Vol. 50, pp. 466-470, 2009.

[3] Justus Piater, Thomas Hoyoux, Wei Du, "Video Analysis for Continuous Sign Language Recognition", *In 4th Workshop on the Representation and Processing of Sign Languages: Corpora and Sign Language Technologies*, 2010

[4] Elena Sánchez-Nielsen, La Laguna, Luis Antón-Canalís and Mario Hernández-Tejera, "Hand Gesture Recognition for Human-Machine Interaction", *Journal of WSCG*, Vol. 12, No. 1-3, 2003.

[5] Ravikiran J, Kavi Mahesh, Suhas Mahishi, Dheeraj R, Sudheender S and Nitin V Pujari, "Finger Detection for Sign Language Recognition", *Proceedings of the International MultiConference of Engineers and Computer Scientists*, Vol. 1, pp. 0-4, 2009.

[6] Lam, L and Suen C.Y, "An Evaluation of Parallel Thinning Algorithms for Character - Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 9, pp. 914-919, 1995.

[7] K. Srinivasan, K. Porkumaran and G. Sainarayanan, "Development of 2D Human Body Modelling using Thinning Algorithm", *ICTACT Journal on Image and Video Processing*, Vol. 1, No. 2, pp. 80-86, 2010.

[8] V. Vijaya Kumar, A. Srikrishna, Sadiq Ali Shaik and S. Trinath, "A New Skeletonization Method Based on Connected Component Approach", *International Journal of Computer Science and Network Security*, Vol. 8, No. 2, pp. 133-137, 2008.

[9] Lei Huang, Genxun Wan and Changping Liu, "An Improved Parallel Thinning Algorithm", *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, pp. 780-783, 2003.