Finger Detection for Sign Language Recognition

Ravikiran J, Kavi Mahesh, Suhas Mahishi, Dheeraj R, Sudheender S, Nitin V Pujari

***Abstract*— Computer recognition of sign language is an important research problem for enabling communication with hearing impaired people. This paper introduces an efficient and fast algorithm for identification of the number of fingers opened in a gesture representing an alphabet of the American Sign Language. Finger Detection is accomplished based on the concept of Boundary Tracing and Finger Tip Detection. The system does not require the hand to be perfectly aligned to the camera or use any special markers or input gloves on the hand.**

***Index Terms*—Boundary Tracing, computer access for disabled, finger detection, image processing, sign language recognition.**

I. INTRODUCTION

The long-term goal of our research is to enable communication between visually impaired (i.e., blind) people on the one hand and hearing and speech impaired (i.e, deaf and dumb) people on the other. Since the former cannot see and the latter use sign language, there is currently no means of communication between such people who are unfortunately in significantly large numbers in a country such as India.

Our project aims to bridge this gap by introducing an inexpensive computer in the communication path so that the sign language can be automatically captured, recognized and translated to speech for the benefit of blind people. In the other direction, speech must be analyzed and converted to either sign or textual display on the screen for the benefit of the hearing impaired. An important research problem in such a solution is the automatic recognition of sign language through image processing.

This paper presents a special-purpose image processing algorithm that we have developed to recognize signs from the American Sign Language with high accuracy. A significant contribution of this result is that it does not require the person making the signs to use any artificial gloves or markers.

In recent years, there has been a tremendous amount of research on hand gesture recognition. Some of the earlier gesture recognition systems attempted to identify gestures using glove-based devices that would measure the position and joint angles of the hand [5]. However, these devices are very cumbersome and usually have many cables connected to a computer. This has brought forth the motivation of using non-intrusive, vision-based approaches for recognizing gestures.

.

This work has focused primarily on identifying the number of fingers opened in a gesture representing an alphabet of the American Sign Language. Knowing the number of fingers open, it is possible to identify reliably the gesture as an alphabet belonging to a class of gestures which have fingers open (see Fig. 6 below).

We have been working with the added constraints of minimal calibration of the system between different users. Many previous gesture based systems have the common element of markers on the hand [1, 2], data gloves or colored gloves worn by the user [3] to allow the gesture and pose to be derived.

This system achieves the objective of detecting the number of open fingers using the concept of boundary tracing combined with finger tip detection. It handles breaks, if any, during boundary tracing by rejoining the trace at an appropriate position.

II. RELATED WORK

There have been many previous works which extracted certain features of the hand for finger detection.
Some common features extracted include hand silhouettes [6], [7], contours [8], key points distributed along hand (fingertips, joints) [9], [10], [11], [12], [18], [21], and distance-transformed images [13].

There have also been works where finger detection has been accomplished via color segmentation and contour extraction [12, 14]. But this technique requires fine-tuning every time the system switches to a new user as the color complexion varies from person to person.

In view of the limitations posed by the schemes discussed above there is a need to devise an efficient and robust technique for finger detection. The next section discusses our solution to this problem.

III. PROPOSED METHODOLOGY

In this paper we present a robust and efficient technique for finger detection. Our method has three main phases of processing viz., Edge Detection, Clipping and Boundary Tracing.

Manuscript received December 8, 2008. This work was supported in part by a research grant under the Research Promotion Scheme from the All India Council for Technical Education.

Authors are with the Department of Computer Science, PES Institute of Technology, Ring Road, BSK 3rd Stage, Bangalore, 560085 India. Kavi Mahesh and Nitin V Pujari are professors in the department and the other authors are undergraduate students (+919845290073, ravikiran.j.127@gmail.com, DrKaviMahesh@GMail.com).

The first phase employs Canny edge operator and produces an edge detected image which reduces the number of pixels to be processed at runtime. The next phase clips the undesirable portion of the edge detected image for further processing. The final phase traces the boundary of the image and in the process detects finger tips which aid in finger detection.

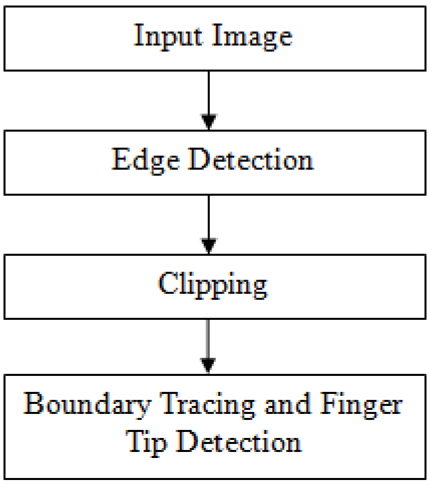


Figure 1: Finger Detection Flowchart

A. *Canny Edge Detection*

Edge detection is a phenomenon of identifying points in a digital image at which the image brightness changes sharply or, more formally, has discontinuities.

The Canny algorithm uses an optimal edge detector based on a set of criteria which include finding the most edges by minimizing the error rate, marking edges as closely as possible to the actual edges to maximize localization, and marking edges only once when a single edge exists for minimal response [4]. According to Canny, the optimal filter that meets all three criteria above can be efficiently approximated using the first derivative of a Gaussian function.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad (1)$$

$$\frac{\partial G(x,y)}{\partial x} \alpha\, x e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad \frac{\partial G(x,y)}{\partial y} \alpha\, y e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad (2)$$

The first stage involves smoothing the image by convolving with a Gaussian filter. This is followed by finding the gradient of the image by feeding the smoothed image through a convolution operation with the derivative of the Gaussian in both the vertical and horizontal directions.

$$G = \sqrt{Gx^2 + Gy^2} \qquad (3)$$

$$\theta = \arctan\left[\frac{Gy}{Gx}\right] \qquad (4)$$

The non-maximal suppression stage finds the local maxima in the direction of the gradient, and suppresses all others, minimizing false edges. The local maxima is found by comparing the pixel with its neighbors along the direction of the gradient. This helps to maintain the single pixel thin edges before the final thresholding stage.

Instead of using a single static threshold value for the entire image, the Canny algorithm introduced hysteresis thresholding, which has some adaptivity to the local content of the image. There are two threshold levels, t_h, high and t_l, low where $t_h > t_l$. Pixel values above the t_h value are immediately classified as edges. By tracing the edge contour, neighboring pixels with gradient magnitude values less than t_h can still be marked as edges as long as they are above t_l. This process alleviates problems associated with edge discontinuities by identifying strong edges, and preserving the relevant weak edges, in addition to maintaining some level of noise suppression. While the results are desirable, the hysteresis stage slows the overall algorithm down considerably.

The performance of the Canny algorithm depends heavily on the adjustable parameters, σ, which is the standard deviation for the Gaussian filter, and the threshold values, t_h and t_l. σ also controls the size of the Gaussian filter.

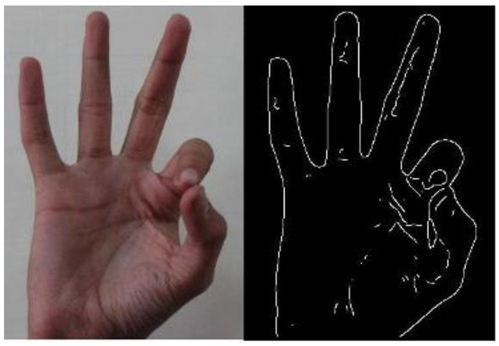


Figure 2: Image of a Hand Gesture Before and After Edge Detection.

The bigger the value for σ, the larger the size of the Gaussian filter becomes. This implies more blurring, necessary for noisy images, as well as detecting larger edges. As expected, however, the larger the scale of the Gaussian, the less accurate is the localization of the edge. Smaller values of σ imply a smaller Gaussian filter which limits the amount of blurring, maintaining finer edges in the image. The user can tailor the algorithm by adjusting these parameters to adapt to different environments with different noise levels. The threshold values and the standard deviation for the Gaussian filter are specified as 4.5, 4.7 and 1.9 for the above used input source and background environment.

B. *Clipping*

Clipping is used to cut a section through the data (image) currently being rendered. The image contents that pertain to the area of interest are retained after clipping.

The edge detected image contains portions which are unnecessary for further analysis. Hence we eliminate them by adopting two techniques discussed below.

The first technique examines pixels from the bottommost y level and at each level checks if there are three or more consecutive white pixels. If the above condition is satisfied we mark this y-level as “y1”.

The second technique exploits the fact that most of the edge detected images of hand gestures have the wrist portion which has a constant difference between the either ends on the same y-level. When it approaches the palm and region of the hand above it, this difference increases drastically. We make use of this fact and find the y-level where this event occurs and mark this y-level as “y2”.

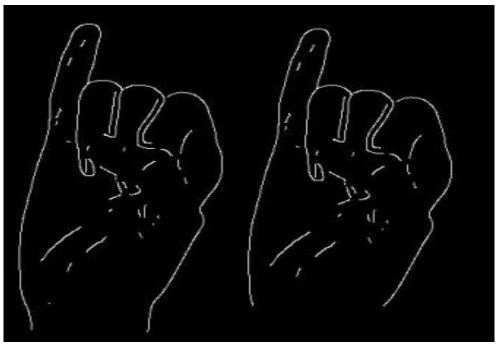


Figure 3: Edge-Detected Image of a Hand Gesture Before and After Clipping.

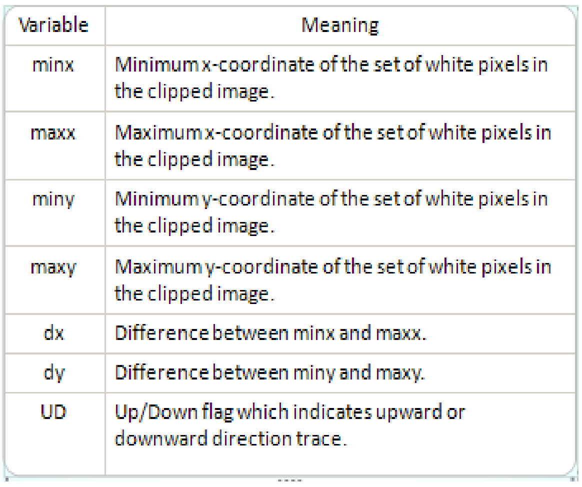
Now we choose the maximum of (y1, y2) as the clipping y-level. All the pixels below this y-level are now cleared by overwriting them with a black pixel.

C. *Boundary Tracing*

This phase of the algorithm is the heart of processing. The edge detected image which is clipped serves as the input to this phase. The output is the traced image where the trace-points are highlighted in blue and the points where the finger tip is detected are highlighted in red.

This phase consists of the following steps: identifying the optimal y-level, identifying the initial trace direction, tracing with appropriate switch of direction, rejoining the trace on encountering breaks, finger tip detection.

In the explanation of the above steps, the following are assumed:



Variable	Meaning
minx	Minimum x-coordinate of the set of white pixels in the clipped image.
maxx	Maximum x-coordinate of the set of white pixels in the clipped image.
miny	Minimum y-coordinate of the set of white pixels in the clipped image.
maxy	Maximum y-coordinate of the set of white pixels in the clipped image.
dx	Difference between minx and maxx.
dy	Difference between miny and maxy.
UD	Up/Down flag which indicates upward or downward direction trace.

1) *Identifying the Optimal y-level*

This step involves identifying the y-level to start the trace of the image. By experimenting with different y-levels for various image samples, we fixed the optimal y-level as 30-35% of *dy* from the top of the edge detected clipped image. Hence the starting pixel for trace is the first white pixel found as a result of scanning from *minx* to *maxx* on the optimal y-level.

2) *Identifying the initial trace direction*

From the initial trace point, we proceed towards miny without changing the current x-coordinate, until there is no pixel to proceed. Then we examine the neighboring white pixels and set the “left” or “right” flags appropriately.

3) *Tracing with appropriate switch of direction*

After identifying the trace direction, the system proceeds by tracing pixels. For every five pixels traced, we write the fifth pixel with blue color. Also whenever we find no pixels in the current direction, we check if there is a pixel in the other direction. If present, we toggle the direction of trace, else it is a break.

4) *Rejoining the trace on encountering breaks*

A break can be encountered while tracing upwards or downwards a finger. They are handled separately based on the flag “UD” which indicates whether we are travelling up(+1) or down(-1).

Varying the x-coordinate from current value to *maxx*, we scan from the current y-level towards the upper or lower boundary of the image based on the value of “UD” for a white pixel. If found, we then start the trace again from this pixel re-initializing the count to 0.

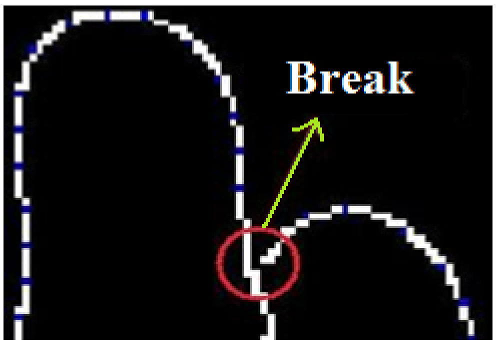


Figure 4: Rejoining the Trace after Encountering a Break.

5) *Finger Tip Detection*

Whenever the flag "UD" switches from +1 to -1, it indicates the change in trace direction from up to down. This event signifies the detection of a finger tip and hence we write this pixel with red. After finding a finger tip there are two techniques to find the starting point of the next finger.

In the first technique we trace downwards from the finger tip position to the optimal y-level, then from that position we increment the x-coordinate until we find a white pixel, this serves as the starting point for processing the next finger.

The second technique is employed when the fingers are adjoined. In this technique we check if a white pixel exists towards the right/left of the current downward trace pixel, if found, this serves as the starting point of processing next finger.

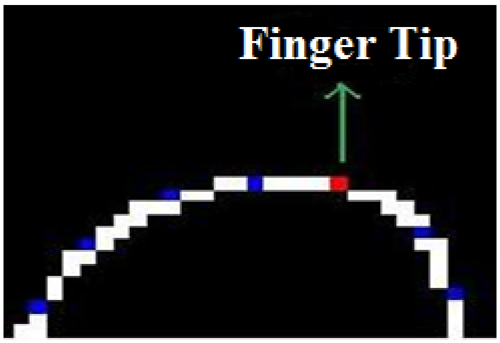


Figure 5: Finger Tip Detection.

The algorithm iterates through the above process until the x-coordinate has not reached *maxx*. The count of the finger tips at this stage is the count of the number of fingers.

IV. Implementation and Results

In this section we describe the accuracy of our algorithm. The application has been implemented in Java 1.6 (Update 7) using the ImageIO libraries. The application has been tested on a Pentium IV running at 3.00 GHz. The images have been captured using a 6 Mega Pixel Canon PowerShot S3 IS.
The captured images are of resolution 640x480. For the performance evaluation of the finger detection, the system has been tested multiple times on samples authored by a set of 5 different users.

Figure 6 shows a subset of American Sign Language gestures which have fingers open. Figure 7 shows the performance evaluation results. These results are plotted on a graph, where the y-axis represents number of tests and the x-axis represents the gestures of the American Sign Language corresponding to alphabets. The columns are paired for each gesture: the first column is the number of times the fingers are correctly identified in the gesture; the second column is the total number of times that the test on the gesture has been carried out. As it can be seen in Fig. 7, the finger recognition works accurately for 95% of the cases.

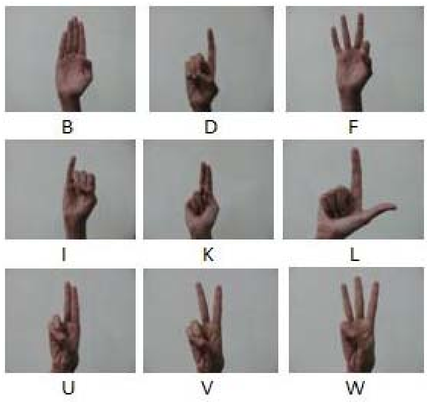


Figure 6: American Sign Language Gestures

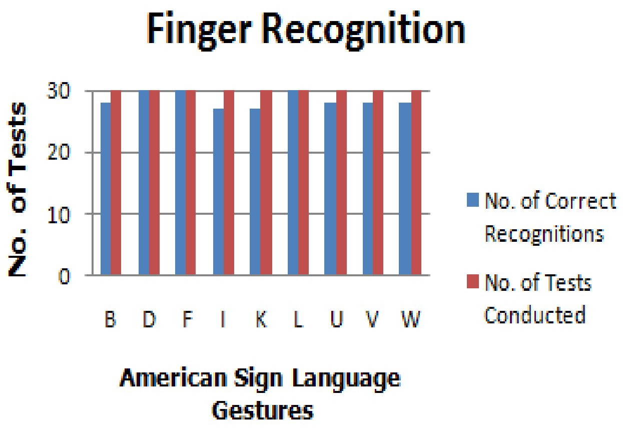


Figure 7: System Performance Evaluation Results.

V. Conclusion and Future Work

A boundary-trace based finger detection technique is presented and cusp detection analysis is done to locate the finger tip. This algorithm designed is a simple, efficient and robust method to locate finger tips and enables us to identify a class of hand gestures belonging to the American Sign Language which have fingers open.

The accuracy obtained in this work is sufficient for the purposes of converting sign language to text and speech since a dictionary can be used to correct any spelling errors resulting from the 5% error in our gesture recognition algorithm.

In future work, sensor based contour analysis can be employed to detect which fingers in particular are open. This will give more flexibility to interpret the gestures. Furthermore, hand detection method using texture and shape information can be used to maximize the accuracy of detection in cluttered background.

More importantly, we need to develop algorithms to cover other signs in the American Sign Language that have all the fingers closed. An even bigger challenge will be to recognize signs that involve motion (i.e, where various parts of the hand move in specific ways).

In our future work, we plan to complete other modules of the overall solution needed to enable communication between blind and deaf people. In particular, we will focus on translating the recognized sequences of signs to continuous text (i.e., words and sentences) and then to render the text in speech that can be heard by blind people.

ACKNOWLEDGMENT

The authors would like to thank the Principal and the management of the PES Institute of Technology for their continued support in carrying out this research and in publishing this paper at the conference.

REFERENCES

[1] V. Buchmann, S. Violich, M. Billinghurst A. Cockburn: FingARtips: Gesture based direct manipulation in Augmented Reality, In Proc. of the 2nd international conf. on Computer graphics and interactive techniques in Australasia and South East Asia (Graphite 2004). 15-18th June Singapore, 2004, ACM Press, New York, New York, pp. 212-221.

[2] Y. Kojima, Y. Yasumuro, H. Sasaki, I. Kanaya, O. Oshiro, T. Kuroda, Y. Manabe and K. Chihara: Hand Manipulation of Virtual Object in Wearable Augmented Reality, In Proc. 7th International Conf. on Virtual Systems and Multimedia (VSMM'01), pp 463-470, October 2001

[3] Y. Iwai, K. Watanabe, Y. Yagi, M. Yachida: Gesture Recognition by Using Colored Gloves, IEEE International Conference on Systems, Man and Cybernetics (SMC'96),Vol. 1, pp. 76-81, Beijing, China, Aug. 1996.

[4] Canny, J., “A Computational Approach to Edge Detection”, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, November 1986.

[5] V. I. Pavlovic, R. Sharma, and T. S. Huang, “Visual interpretation of hand gestures for human-computer interaction: A review.”IEEE Trans. on PatternRecognition and Machine Intelligence, vol. 19 issue 7, pp 677-695, July 1997.

[6] N. Shimada et al., “Hand gesture estimation and model refinement using monocular camera—Ambiguity limitation by inequality constraints,” in Proc. 3rd Conf. Face and Gesture Recognition, pp. 268-273, 1998.

[7] M. Sonka, V. Hlavac, and R. Boyle, Image processing, Analysis, and Machine Vision, Brooks/Cole Publishing Company, 1999.

[8] T. Starner and A. Pentland, “Real-time American Sign Language recognition from video using hidden Markov models,” in Proc.of International Symposium on Computer Vision, pp. 265–270, 21-23 Nov. 1995.

[9] J. Lee and T. Kunii, “Model-based analysis of hand posture,” IEEE Comput. Graph. Appl., vol. 15, no. 5, pp. 77-86, Sept. 1995.

[10] H. Fillbrandt, S. Akyol, K. F. Kraiss, “ Extraction of 3D Hand Shape and Posture from Image Sequences for Sign Language Recognition.” IEEE International Workshop on Analysis and Modeling of Faces and Gestures, vol. 17, pp. 181-186, October 2003.

[11] T. Starner, A. Pentland , J. Weaver, “Real-Time American Sign Language Recognition Using Desk and Wearable Computer Based Video,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20 issue 12, pp. 1371-1375, December 1998.

[12] K. Imagawa, S. Lu, and S. Igi, “Color-based hands tracking system for sign language recognition” , in Proc. Third IEEE International Conference on Automatic Face and Gesture Recognition, pp. 462 – 467, April 14-16, 1998.

[13] C.H. Teh and R. T. Chin, “On image analysis by the methods of moments” , IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 10, no. 4, pp. 496 – 513, July 1988.

[14] Sung Kwan Kang, Mi Young Nam, Phill Kyu Rhee, “Color Based Hand and Finger Detection Technology for User Interaction”, International Conference on Convergence and Hybrid Information Technology, Aug 2008.

[15] J.J. Kuch and T.S. Huang, “Vision-based hand modeling and tracking for virtual teleconferencing and telecollaboration,” in Proc. IEEE Int. Conf. Computer Vision, Cambridge, MA, pp. 666-671. June 1995.

[16] J. Davis and M. Shah, “Visual gesture recognition,” Vision, Image, and Signal Processing, vol. 141, pp. 101-106, Apr. 1994.

[17] J. Rehg and T. Kanade, “DigitEyes: vision-based human hand tracking,” School of Computer Science Technical Paper CMU-CS-93-220, Carnegie Mellon Univ., Dec.1993.

[18] Y. Shirai, N. Tanibata, N. Shimada, “Extraction of hand features for recognition of sign language words,” VI'2002,Computer-Controlled Mechanical Systems, Graduate School of Engineering, Osaka University, 2002.

[19] C. Nölker, H. Ritter, “Detection of Fingertips in Human Hand Movement Sequences,” Gesture and Sign Language in Human-Computer Interaction, I. Wachsmuth and M. FroÈhlich, eds., pp. 209-218, 1997.

[20] B. Bauer and H. Hienz, “Relevant features for video-based continuous sign language recognition,” in Proc. of Fourth IEEE International Conference on Automatic Face and Gesture Recognition, pp. 440-445, March 2000.

[21] Y. Hamada, N. Shimada and Y. Shirai, “Hand Shape Estimation under Complex Backgrounds for Sign Language Recognition” , in Proc. of 6th Int. Conf. on Automatic Face and Gesture Recognition, pp. 589-594, May 2004.