

Comparison of Massively Parallel Hand-print Segmenters

R. Allen Wilkinson and Michael D. Garris,
National Institute of Standards and Technology,
Gaithersburg, MD 20899

Abstract

NIST has developed a massively parallel hand-print character recognition system that allows components to be interchanged. Using this system, three different character segmentation algorithms have been developed and studied. They are blob coloring, histogramming, and a hybrid of the two. The blob coloring method uses connected components to isolate characters. The histogramming method locates linear spaces, which may be slanted, to segment characters. The hybrid method is an augmented histogramming method that incorporates statistically adaptive rules to decide when a histogrammed item is too large and applies blob coloring to further segment the difficult item. The hardware configuration is a serial host computer with a 1024 processor Single Instruction Multiple Data (SIMD) machine attached to it. The data used in this comparison is "NIST Special Database 1" which contains 2100 forms from different writers, where each form contains 130 digit characters distributed across 28 fields. This gives a potential 273,000 characters to be segmented. Running the massively parallel system across the 2100 forms, blob coloring required 2.1 seconds per form with an accuracy of 97.5%, histogramming required 14.4 seconds with an accuracy of 95.3%, and the hybrid method required 13.2 seconds with an accuracy of 95.4%. The results of this comparison show that the blob coloring method on a SIMD architecture is superior.

1 Introduction

Improved recognition algorithms and the increased performance of computers have touched off an imaging revolution, offering a new method of information archiving, retrieving, and processing for many existing and new applications. This paper compares three different character segmenters within a modular recognition system. The recognition system has been designed to read hand-printed information from structured forms, that is, documents on which entry fields are consistently located, so that upon correct identification of the form type, relative field locations can be predicted.

Traditionally, documents of this nature have been stored as paper archives. Now they are beginning to be digitally scanned and archived onto magnetic or optical media as binary or

8-bit images. These documents are collected at centralized locations within a short period of time, such as during a decennial census, and then processed at a later date. Therefore the recognition system developed at NIST has been designed to process archived images independent of when digitization occurs. A massively parallel machine has been used in order to obtain high throughput and to study the feasibility of using new computer architectures for this type of application.

Character recognition, the classification of well formed and cleanly segmented characters, has been studied in great detail in the past[1][2][3][4][5]. Studies conducted on a model recognition system at NIST[6] suggest that the recognition component is only one of many important components needed to successfully convert laboratory research on optical character recognition into an effective application technology. Areas in need of development include the development of intelligent and efficient segmentation schemes, which is the focus of this paper. Section 2 defines the hardware architecture supporting the system. Section 3 describes the functional components of the system. Section 4 describes three different hand-printed character segmentation approaches. Section 5 presents the results of the study and section 6 contains conclusions drawn from the results.

2 Hardware Architecture

This model recognition system is implemented across two integrated computers¹. The data storage and central processing control is supported by a Sun 4/470 UNIX server. The Sun has 32 Megabytes of main memory, approximately 10 gigabytes of magnetic disk, and two CD-ROM drives. Connected to the Sun 4/470 is an Active Memory Technology 510C Distributed Array Processor (DAP). The parallel machine is a Single Instruction Multiple Data (SIMD) architecture and consists of two separate 32 by 32 grids of tightly coupled processors. One grid contains 1-bit processing elements while the other contains 8-bit processing elements. Data mappings of both vector mode and matrix mode are well-suited to the DAP, making it useful for both neural networks and traditional image processing. The parallel machine is responsible for conducting all low-level computationally intensive system tasks.

The distributed functionality and interaction between the two computers is important. The recognition system has been designed as a laboratory model for algorithmic development and system performance analysis. The system design reflects this by emphasizing modularity over in-line performance optimization. A production version of this system, in which algorithms are fixed and interactions between functional components are known, may be more effective if implemented on other hardware platforms or in Very Large Scale Integration (VLSI). The recognition system in this paper places much more emphasis on the software development of algorithms and functional control than on hardware. A programmable massively parallel machine offers development flexibility along with computational power.

¹The Sun 4/470 and DAP 510C or equivalent commercial equipment are identified in order to adequately specify or describe the subject matter of this work. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the equipment identified is necessarily the best available for the purpose.

System Data Flow

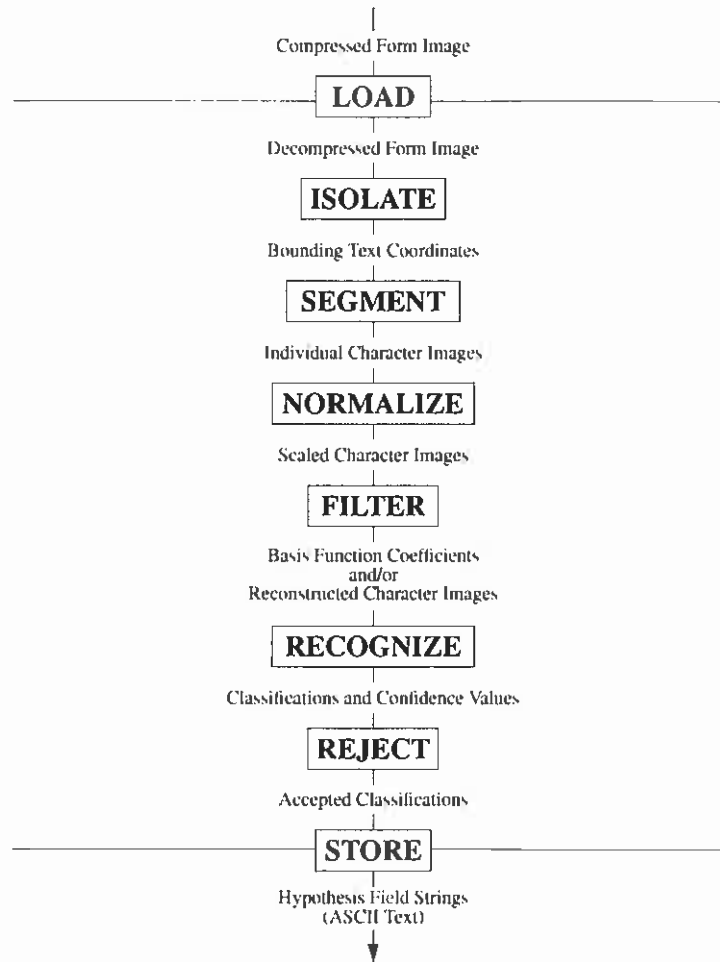


Figure 1: The functional components of the recognition system.

3 Functional Components

Figure 1 illustrates the functional components used in the NIST recognition system model[6]. These modules can be divided into three categories: system loading and storing, image processing, and recognition. The figure charts the transformation of data from initial input image through ASCII text output. An image of a document is loaded into the system, the hand-printed data is located and preprocessed, a recognition algorithm classifies isolated hand-printed characters, and the recognition results are retrieved from the system and stored as text.

The loading and storing modules are used to transfer data from the serial machine to the parallel machine. The loading module is responsible for decompressing the image and transferring it to the parallel machine[7]. The results are returned from the parallel machine by the storing module. All other modules are implemented on the parallel machine.

The isolate, segment, and normalize modules are responsible for the image processing. The images of forms need to be broken down into fields before characters can be segmented. This is done by the isolate module. The segment module, which is discussed in detail in section 4, turns a field image into individual character images, one character per image. These character images are then normalized into 32 by 32 pixel images by the normalize module.

The filter, recognize, and reject modules are responsible for turning character images into ASCII characters. The filter module produces features for the recognize module using a variety of filters[6]. The filters used for this paper are the shear and KL transformations[8]. The recognize module uses the features generated to decide the classification of the image and a confidence of the decision. A multi-layer perceptron, MLP[9], was used in this paper with 48 input neurodes, 10 hidden neurodes, and 10 output neurodes. It is then the responsibility of the reject module to accept the recognize module's classification or to reject and label the image as unknown.

At this point an important observation can be made. In figure 1 there is a consistent and dramatic reduction in the volume of data flowing through the system from input to output. The original binary document images have been digitized at 300 pixels per inch. At input, the system is given 8,000,000 pixels of binary image data, which after successful isolation and segmentation are reduced to about 133,120 pixels. After feature extraction in the filter module, the segmented character images are reduced to 33,280 bits of coefficient values, and after recognition the classifications are stored as 1,040 bits of ASCII text. This represents an 8,000 to 1 data compression from input to output. Each successive module contributes to the reduction of problem complexity and/or required data bandwidth.

4 Segmenters

The segment module represents a critical module in the recognition system's flow, requiring high algorithm complexity to process a large amount of global image data. This realization motivated the investigation of various character segmentation techniques. Using the model

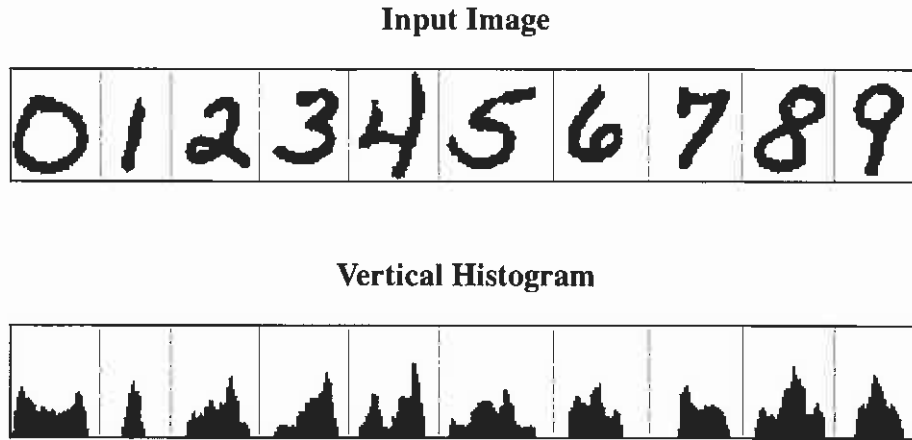


Figure 2: Histogram of an image with segmentation lines

described in section 3, different algorithmic approaches were compiled into the system and compared. NIST has developed an automated scoring technique which has been used to assess and compare the performance of each system configuration. This scoring system compares the reference data with the hypothesised recognition system output using a dynamic string alignment scheme. By reconciling system output with reference information, the scoring package is able to automatically compute performance statistics at the character and field levels.

4.1 Histogram Method

The histogram method segments an image of a line of text into images of characters by finding the vertical voids in the image to be processed. This is accomplished by using thresholded spatial binary histograms [10]. Figure 2 shows the vertical spatial histogram of an input image with the dotted lines signifying the points at which segmentation will occur. The minima of the histogram determine segmentation locations as shown in the figure.

There are two distinct limitations with this simplistic method. First, a character containing a vertical void will be separated and assumed to be two characters. An example of a character that has a vertical void is a double quote as shown in figure 3 (A). Characters which overlap but do not touch, as in figure 3 (B), will not be separated. Also, connected characters, as in figure 3 (C), will not be separated because a vertical void can not be found between them. These phenomena frequently occur in hand-print, making the simple histogram method unsatisfactory.

Hand-printed text can be segmented by applying an enhanced histogram method which uses domain knowledge about hand-printing to assist in segmentation. This knowledge includes the fact that printing is most often written with a slant, and that no one uses a consistent slant [10]. Also statistically based rules can be produced that are specific to each field being segmented by gathering statistics about the data height and location of vertical voids.

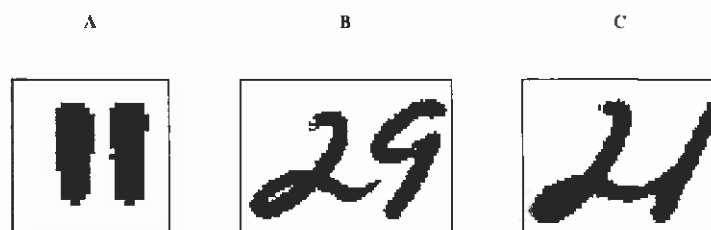


Figure 3: Examples of (A) vertical void, (B) overlapping and (C) connected characters

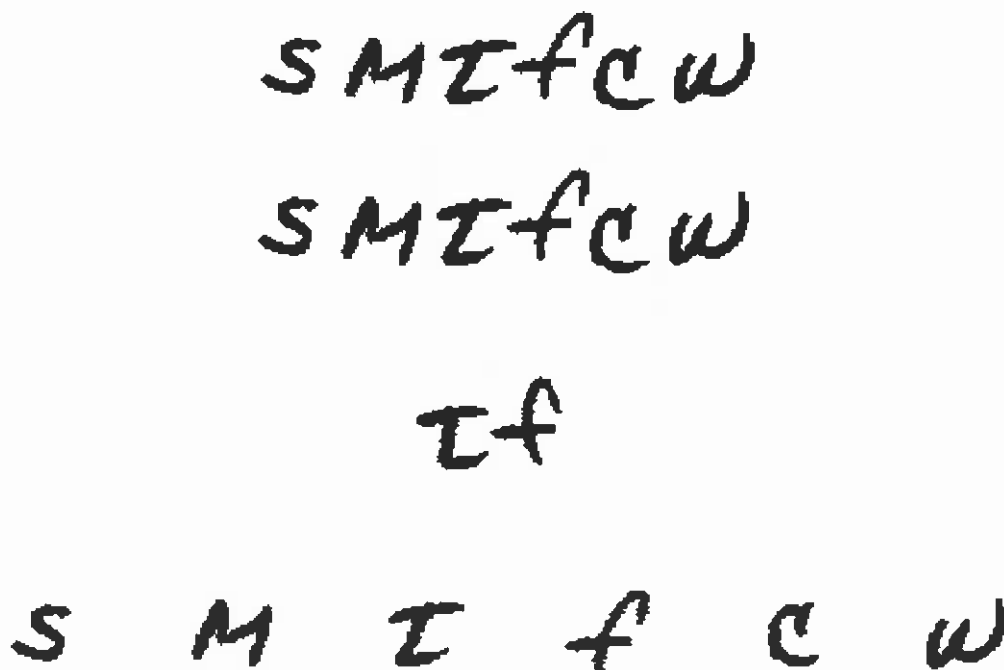


Figure 4: Example of cuts found using Hand-Print Line Segmenter

These rules are used to decide when a segmented image is potentially too large or too small to be a character image. When an image is too large it is assumed to be an image of more than one character. This image needs to be analyzed to find the best slope at which a straight line can segment the image, if such a slope exists. If no slope is found then it is assumed that the image contains a single character. Otherwise, the slope is used to shear the image which is then passed to the segmenter recursively. Shearing an image is shifting rows of pixels in order to slant the image without rotating it. A shear is used because rotating followed by counter-rotating can cause the image to be distorted, whereas shearing is perfectly reversible and therefore causes no distortion. Shearing is also less computationally expensive. This process will continue until no new slope can be found or the segmenter has reached a maximum level of recursion (10 levels for this study). The recursive call limit has not appeared to be a problem; to date the maximum level occurring in practice is three.

The segmentation of the hand-printed string “smtfcw” by the enhanced histogram method is

shown in figure 4. The first line is the input image. The second line shows the segmentation cuts found before applying the statistically adaptive rules, shearing and recursion. Applying the adaptive rules and shearing the image of the “tf” is shown in line three. A recursive call to the segmenter is used to segment the “tf” image. The segmented characters are shown in line four.

4.1.1 Limitations

There are several possible limitations to functionality that can occur with the histogram method. Characters with vertical voids are not concatenated. Some of the angles at which a segmentation cut must be performed are hard to find. New ways of determining the angle need to be investigated. Characters can only be separated by a straight line. It is possible for characters not to have a separating straight line. These overlapping characters are not separable. Connected characters such as in figure 3 (C) are not separated. No mechanisms have yet been implemented to handle this condition. This segmenter operates only on a line image of text as opposed to a page of text, but can be used as part of a page segmentation scheme.

4.2 Blob Coloring Method

The blob coloring method uses the traditional image processing technique of connected components[11]. Connectedness can be defined with respect to the neighbors of a point or pixel (i, j) . The scheme used in this study is the 4-connected components. The four nondiagonal adjacent neighbors are the 4-connected neighbors of a point, while 8-connected neighbors are the 8 points which surround a point. Using one of these schemes, two points, p_1 and p_2 , are said to be connected if there exists a sequence of points, beginning with p_1 and ending with p_2 , such that consecutive points are connected using the same scheme. These connection schemes lend themselves well to being programmed in a parallel manner. On the parallel grid architecture a combination of shifts and compares can quickly label the complete image.

Using blob coloring will cause over-segmentation. For example, an image of a dotted letter “i” will become two separate blobs, the body of the i and the dot above it. These two images really represent one letter and should be only one image. The same would be true for any characters that have breaks in a stroke. Correcting for over-segmentation has been added to the traditional blob coloring so that characters that are comprised of disjoint pieces, such as “5”s with disjoint tops, will be considered one image. The pasting of two adjacent blobs is accomplished by using statistically adaptive rules. One rule assumes that the bottom of two adjacent characters will lie along the same relative baseline. At the same time another rule assumes that the right side of the left character and the left side of the right character will not overlap by more than half the average character’s width. If the bottoms are not sufficiently close and the overlap is not less than half the average character width, then the two blobs are pasted together as if they had never been separated.

4.2.1 Limitations

The blob coloring method suffers from the limitation that it can not separate connected characters such as shown in figure 3 (C) and cases of over-segmentation must be considered.

4.3 Hybrid Method

The hybrid method works just like the enhanced histogram method except that before it accepts an image which exceeds statistical rules it tries to use blob coloring. This occurs in two different situations. The first situation is when histogramming can find only one item in the image and the image data width is greater than a threshold. For our experiments this threshold is 128 pixels, the maximum size of our segmented images. This can occur during the first histogram analysis or just after trying to determine a new angle at which to slant the segmenter's view of the image. This situation can occur when two characters overlap but are not separable by a straight line. Figure 3 (B) shows an example of this. The second situation is when the histogram segmenter has reached the maximum number of levels of recursion, which is 10 in our implementation. This means the histogram method has changed the direction 10 times and continued to segment the image but has not reached the point of returning from the recursion. Under these conditions blob coloring is called to finish the segmenting of the image.

4.4 Limitations

The limitations for the hybrid method are the same as for the histogram method with only a few exceptions. The hybrid segmenter can separate intertwined, such as overlapping but not connected characters, where the histogram segmenter can not.

5 Results

In designing this study it was anticipated that blob coloring would be more accurate and more computationally expensive than the histogram and hybrid methods. The histogram method was anticipated to be the fastest because of the computational intensity of blob coloring on a serial machine, while the hybrid method would likely fall in between. The segmentation speed is reported as the percentage of total system time per form image. These results were obtained by running the system on 2100 full page images from "NIST Special Database 1". With respect to accuracy it was thought that the blob coloring method would be the most accurate followed by the hybrid method then the histogram method. The accuracy is measured by the complete system's output accuracy. The output accuracy is not the segmentation accuracy but the accuracy of the recognition and segmentation modules combined. Since the only module that changes is the segmentation module, only the relative differences are important. The total of possible characters to be segmented was 272.870, evenly distributed among the 10 digit classes.

The actual results were different from that which were hypothesized. The blob coloring method proved to be the fastest, followed by the histogram method, then the hybrid method. The accuracy scores were just as had been hypothesized. Blob coloring took 11.40% of the recognition system time with an output accuracy of 97.5% while segmenting 265.427 possible characters. The histogram method had an output accuracy of 95.3% and took 46.95% of the recognition system time and segmented 257.145 possible characters. The hybrid method achieved an output accuracy of 95.4% which was 44.71% of the recognition system time and segmented 265.754 possible characters.

Method	Seconds per form	Accuracy	Total Segmented	Percentage system time
Blob coloring	2.1	97.5%	265.427	11.40%
Histogram	14.4	95.3%	257.145	46.95%
Hybrid	13.2	95.4%	265.754	44.71%

Table 1: Comparison of Segmenter statistics

6 Conclusions

With the use of a massively parallel hand-print recognition system and the ability to interchange components in that system, various segmentation methods can be analyzed for speed and accuracy. Massively parallel machines are changing the way processing is viewed. Some traditionally CPU intensive algorithmic methods lend themselves well to a parallel machine. With respect to segmentation, blob coloring has proven to be 6 times faster than histogramming when done on a parallel machine, while also being 2% more accurate. This shows that of the three methods developed and studied, blob coloring is the superior method for speed and accuracy.

References

- [1] M. D. Garriss, R. A. Wilkinson, and C. L. Wilson. Methods for enhancing neural network handwritten character recognition. In *International Joint Conference on Neural Networks*, volume 1, pages 695-700. IEEE and International Neural Network Society, 7 1991.
- [2] G. E. Hinton and C. K. I. Williams. Adaptive elastic models for character recognition. In J. Moody, S. Hanson, and R. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4, pages 512-519. Morgan Kaufmann, Denver, December 1991.
- [3] L. D. Jackel, H. P. Graf, W. Hubbard, J. S. Denker, D. Henderson, and Isabelle Guyon. An application of neural net chips: Handwritten digit recognition. In *IEEE International Conference on Neural Networks, volume II*, pages 107-115. San Diego, 1988.

- [4] G. Martin and J. Pittman. Recognizing hand-printed letters and digits. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 405–414. Morgan Kaufmann, 1990.
- [5] C. L. Wilson, R. A. Wilkinson, and M. D. Garriss. Self-organizing neural network character recognition on a massively parallel computer. In *Proc. of the IJCNN, volume II*, pages 325–329, June 1990.
- [6] M. D. Garriss, C. L. Wilson, J. L. Blue, G. T. Candela, P. Grother, S. Janet, and R. A. Wilkinson. Massively parallel implementation of character recognition systems. In *Conference on Character Recognition and Digitizer Technologies*, volume 1661, pages 269–280, San Jose California, February 1992. SPIE.
- [7] M. D. Garriss. Design and collection of a handwriting sample image database. *Social Science Computing Journal*, 10, 1992. to be published.
- [8] Anil K. Jain. *Fundamentals of Digital Image Processing*, chapter 5.11, pages 163–174. Prentice Hall Inc., prentice hall international edition, 1989.
- [9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, et al., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*, chapter 8, pages 318–362. MIT Press, Cambridge, MA, 1986.
- [10] R. A. Wilkinson. Segmenting of Text Images with Massively Parallel Machines. In D. P. Casasent, editor, *Intelligent Robots and Computer Vision*, volume 1607, pages 312–323. SPIE, Boston, 1991.
- [11] R. C. Gonzalez and P. Wintz. *Digital Image Processing*. Addison-Wesley Publishing Company, second edition, November 1987.