Segmenting Text Images With Massively Parallel Machines

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

To be presented at SPIE Intelligent Robotic Systems, Boston, Nov. 11-15, 1991

Abstract

Image segmentation, the partitioning of an image into meaningful parts, is a major concern of any computer vision system. The meaningful parts of a text image are lines of text, words and characters. In this paper, the segmentation of pages of text into lines of text and lines of text into characters on a parallel machine will be examined. Using a parallel machine for text image segmentation allows the use of techniques that are impractical on a serial machine due to the computation time needed. It is possible to use a parallel machine to segment text images of lines using spatial histograms with an accuracy of 97.9% at a speed of 30 milliseconds or less per character. Statistically adaptive rules based on dynamic adaptive sampling are used for line segmentation and also for improved accuracy of character segmentation. The segmentation of lines from a page can also be accomplished using a set of statistically adaptive rules which allow sloped lines of text to be segmented. The use of these statistical rules on a parallel machine increases processing time by no more than 1 millisecond per character. Using statistical rules in combination with knowledge about the printed style increases the segmentation accuracy to 99.2% correct for machineprinted text and 89.6% for handprinted text.

1 Introduction

The goal of this paper is to describe how a binary image of a page of text is segmented into individual character images. Segmentation is the separation of a textual image into meaningful parts, which, at the lowest level, are character images. A character is an irreducible symbol in a writing system. For simplicity in this paper, the background of an image will be white and the foreground characters will be black.

Several properties of the text image can adversely affect segmentation, including salt and pepper noise, touching characters, and broken characters. Salt and pepper noise is the reversal of polarity in a digitized pixel, where black becomes white or white becomes black as shown in part A of figure 1. Touching characters occur when two adjacent characters are digitized so that the binary representation of the characters is connected, part B of figure 1. A single character that is itself cut into parts during segmentation is a broken character.

Broken characters can be caused by over-segmentation or by noise causing discontinuity in a character, part C of figure 1.

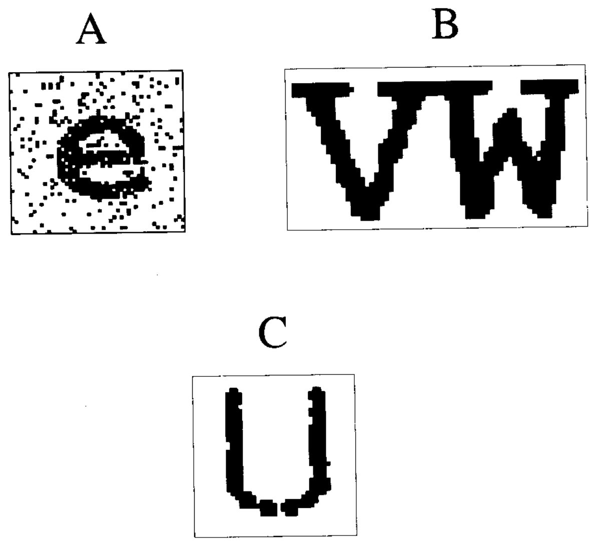


Figure 1: Examples of (A) salt and pepper noise, (B) a connected character and (C) a broken character

2 The Character Recognition System Problem

An important fact to understand is the volume of documents to be digitized. This volume is very large. One credit card company now digitizes 4×10^{12} bytes of document images each year and stores the data on optical media. It is estimated that the total key data entry segment of the United States economy is 20 billion dollars per year.

It is also important to understand the magnitude of the data to be manipulated when processing an image of a page of text. An average page is 21.25 by 27.50 centimeters. When scanned at 12 pixels per millimeter the image is 2550 by 3300 pixels which is 8,415,000 pixels total. Since it is a binary image that will translate into 8,415,000 bits or 1,051,875 8-bit words, that is a 1 megabyte data file.

If the page is to be rotated each pixel must be moved, which means 8,415,000 calculations must be made to find the new pixel location. Each calculation consists of two trigonometric function calls, four multiplies and two additions; that is, at least eight floating point instructions. It will take at least 67.32 megaflops to compute the rotation of a binary page. Rotation is only one of many basic and simple image processing operations including; histograms, translation, image smoothing and sharpening.

At the present time the technology of character recognition is advancing rapidly for several reasons. First, the introduction of document imaging systems has made the incremental cost of introducing character recognition much lower by providing a large supply of scanned images. Second. new methods of character recognition, such as neural networks [1] [2], are increasing the accuracy of the recognition process. Finally, many of the new recognition methods, and image processing methods in general, are being made more practical by parallel processing computers, which have significantly reduced computing costs and increased recognition speed.

There are three parts to the character recognition system: finding the relevant text, segmenting the text, and recognizing the text. Each step when performed accurately will increase the performance of the next step. Many systems have been proposed that would be able to find the relevant text [3] [4]. When finding the relevant text. the less non-relevant graphical or noise data that is retained, the better the segmentation of the text can be done. Correspondingly, recognition of the text is much easier and increases in accuracy as the segmentation increases in accuracy.

3 Other Work in Segmentation

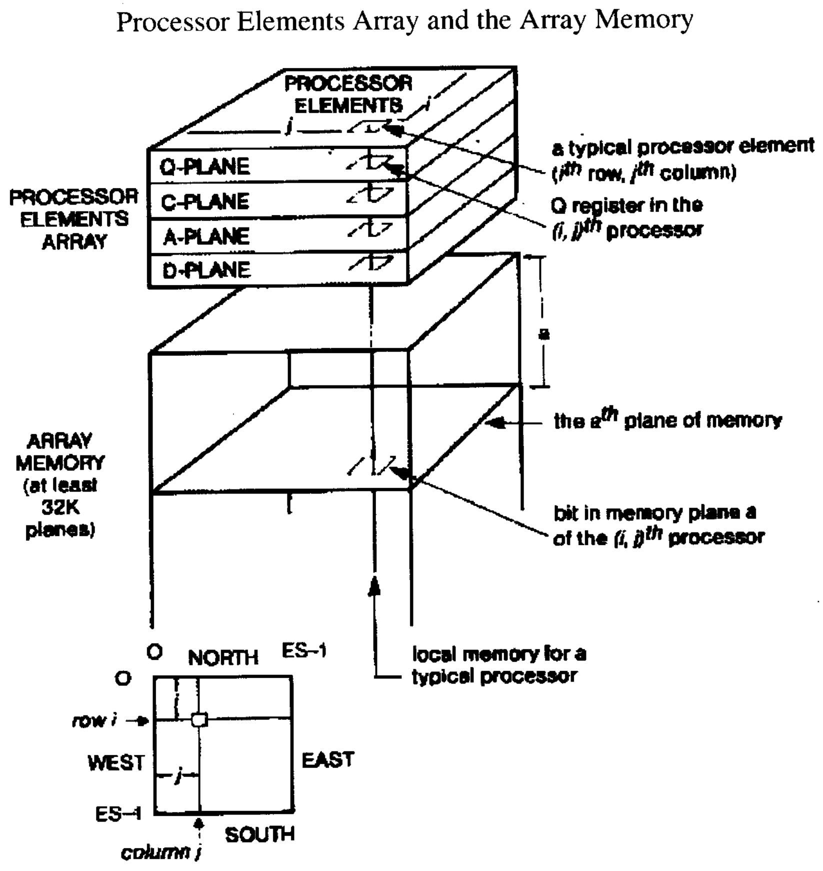
Several methods have been developed for segmenting document images. Segmentation using selective attention presented by Fukushima [5] appears to be promising, especially in separating connected characters. However, this work was done on a very small data set, only five characters, and no accuracy or timing results were given. Enhanced border following as demonstrated by Yamada [6] uses a new concept of connectivity to enhance the traditional border following algorithm. Again, however, no accuracy or timing results were given. Katusunori in his United States Patent [7], describes a way of reconnecting over-segmented characters but not how to segment them. His technique uses the context of the characters to decide if Japanese handwritten characters that have been segmented need to be concatenated to create valid characters and words.

4 Parallel Hardware Architecture

The Single Instruction Multiple Data (SIMD) architecture used for this study was an Active Memory Technology 510 Distributed Array Processor with 8-bit math coprocessor[1]. This machine consists of a 32 by 32 grid of 1-bit processor elements (PE) and a 32 by 32 grid of 8-bit processors. Operation of the PE array is controlled by a 4 MIPS RISC master control unit (MCU). All program instructions are stored in a separate program memory and are passed to the PE array through the MCU. A block diagram of this architecture is shown in figure 2.

All data is stored in a separate array memory. The array memory is organized in 32 by 32 1-bit planes with each of the bits in each plane connected to one PE. Data can also be passed between PEs along the grid. The cycle time of all PEs is 100 ns. This processor configuration is capable of performing ten billion binary operations per second; processing time increases proportionally with the precision of data items used. Two data mappings are

[1]DAP510c or equivalent commercial equipment may be 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.



Processor Elements Array and the Array Memory
PROCESSOR ELEMENTS
i
j
PROCESSOR ELEMENTS ARRAY
Q-PLANE
C-PLANE
A-PLANE
D-PLANE
a typical processor element (i^{th} row, j^{th} column)
Q register in the $(i, j)^{th}$ processor
a
the a^{th} plane of memory
ARRAY MEMORY (at least 32K planes)
bit in memory plane a of the $(i, j)^{th}$ processor
local memory for a typical processor
O
O
NORTH
ES-1
row i
WEST
EAST
ES-1
SOUTH
column j

Figure 2: Array processor architecture for a massively parallel computer.

particularly well suited to the DAP structure: a vector mode in which successive bits of a single word are mapped into a row of the array, and a matrix mode in which successive bits of a word are mapped into layers of the array memory vertically. Operations in both of these modes of operation are used in the segmentor implementation presented in this paper.

The use of a massively parallel computer allows the application of techniques that would not be implemented on a serial computer due to their computational expense. Image processing and analysis can be done with fewer system cycles since many pixels of the image can be manipulated or analyzed at once. In the case of the DAP510c, up to 1024 pixels can be worked with at the same time.

5 First-Generation Line Segmentor

The first-generation line segmentor 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 spatial binary histograms. A vertical spatial binary histogram simply counts the number of "on" pixels in each column while a horizontal spatial histogram counts the pixels in each row.

Vertical voids are now identified by a count of zero in the vertical histogram. Figure 3 shows the vertical spatial histogram of an input image. The valleys in the histogram are equivalent to a count of zero as described above.

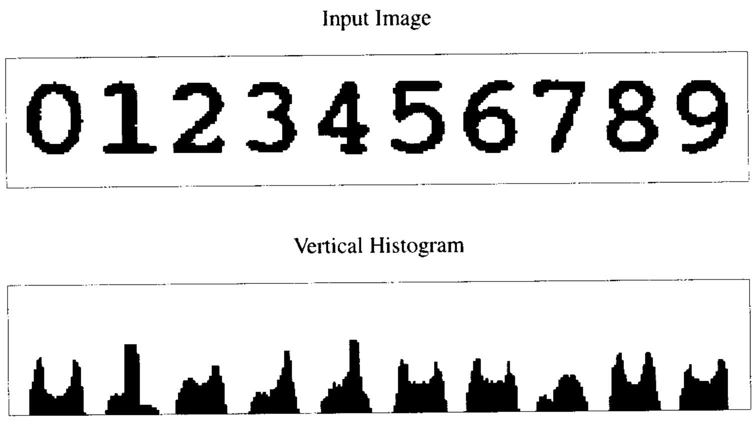


Figure 3: Histogram of an image

One problem that can be noticed almost immediately is that it may be very hard to find a count of zero in a histogram. Most often this is caused by salt and pepper noise. By applying a threshold to a spatial histogram some of the noise may be ignored. The basic spatial histogram uses a thresholding level equivalent to zero. If the thresholding level is set to two, then voids are found where the count is two or less. Using this process will produce more accurate histograms and reduce noise errors.

There are three distinct limitations with this simplistic method. First, the first-generation line segmentor is meant only to segment a line of characters, as opposed to a page of lines. Second, 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 4. And finally, connected characters, as in figure 4 part B, will not be separated because a vertical void can not be found between them.

Figure 5 shows the the image given to the first generation segmentor on top and the segmentation cuts to that image on the bottom.

6 Knowledge-Based Segmentor Enhancments

To further improve the segmentor, it was necessary to use *a priori* knowledge about the image being segmented. The additional information is used for improving machineprinted text segmentation is quite different from that used for handprinted text. The knowledge that can be applied for machineprinted characters is that the width and height of characters

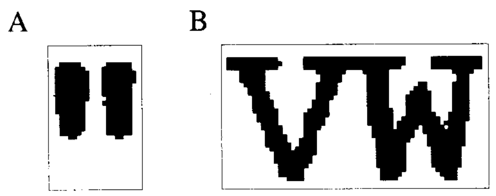


Figure 4: Examples of (A) a vertical void and (B) a connected character

from a fixed-width font are consistent throughout the font and also have a consistent aspect ratio. Adaptive rules are used to determine the width of the font from the statistics gathered about the line being segmented. This technique will be further explored during the second-generation segmentor discussion. For handprinted characters, the knowledge used is that printing is most often written with a slant, and that no one uses a consistent slant. By gathering statistics about the data height and location of vertical voids, a statistically based rule can be produced that is specific to each page image. An example of a segmentor using this kind of knowledge based rule is the third generation segementor. Once again the use of a massively parallel computer makes this technique much more feasible.

6.1 Second Generation Line Segmentor

The second generation line segmentor is designed to segment machineprinted text. The enhancement to the first generation is the use of statistically adaptive rules to decide which segmented images are too large or too small. This decision allows large images, which are connected characters, to be broken down or fragmented into individual characters. Small images, characters with vertical voids, will be concatenated into larger images.

Figure 6 shows the order of processing done by the second-generation segmentor. On the top is the input image. The second line depicts the segmentation cuts found without using the statistically adaptive rules. The segmentation cuts produced after applying the adaptive rules are shown in the bottom line.

6.2 Restrictions

Using statistically adaptive rules will remove several of the limitations found in the first generation segmentor. This method cannot remove all connected and disjoint errors, but does produce significantly improved results over the first-generation segmentor. The reason some of the errors will continue to occur is that the rules are based on statistics. An error on the extremes of the population statistics will continue to be an error. The second generation segmentor only works for segmenting a line of characters, but is able to separate the majority of the connected characters and can concatenate disjoint characters like the double quote.

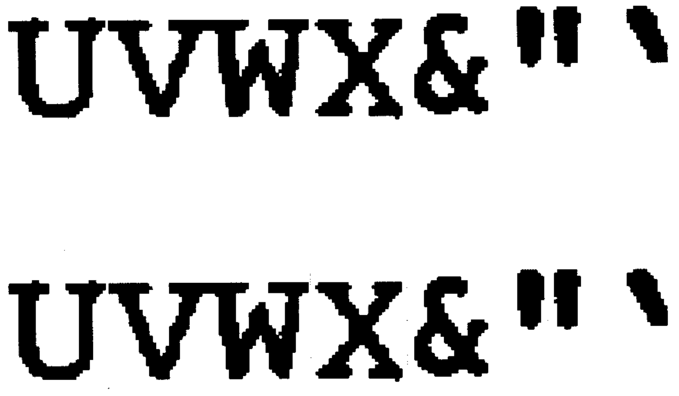


Figure 5: Example of segmentation cuts found using First Generation Segmentor

6.3 Third-Generation Line Segmentor

Handprinted text can be segmented using a third-generation segmentor which uses knowledge about hand printing to assist in segmentation. Determining the sizes of handprinted characters is done using a statistical adaptation rule. These rules are used to decide when a segmented image is potentially too large 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 it has a slope. If no slope is found then it is assumed that the image is a character image. Otherwise the slope is used to shear the image which is then passed to the segmentor recursively. Shearing an image is shifting rows of pixels in order to slant the image without rotating the image. A shear is used because rotating then unrotating can cause the image to be distorted whereas shearing and unshearing will not cause distortion. This process will continue until no new slope can be found or the segmentor has been called ten times recursively, the recursive limit. The recursive call limit has not appeared to be a problem because it has to date not been called more than three times recursively, but the limit is needed for this to work on the DAP510c.

The segmentation of the "smtfcw" handprinted string by the third-generation segmentor is shown in figure 7. 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 segmentor is used to segment the "tf" image. The segmented characters are shown in line four.

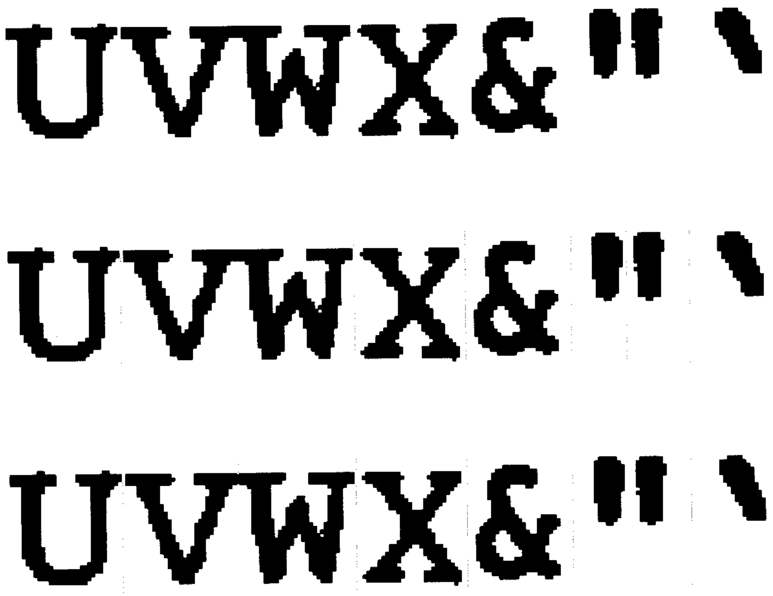


Figure 6: Example of segmentation cuts found using the Second Generation Line Segmentor

6.4 Restrictions

There are several possible restrictions on functionality that can occur with the third generation segmentor. Connected characters are not separated and characters with vertical voids are not concatenated. No mechanisms have yet been implemented to handle these conditions. 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. Disjoint characters can only be separated by a straight line. It is possible for disjoint characters not to have a separating straight line. This segmentor is meant to work only with an image of a line of text as opposed to a page of text.

7 Page Segmentor

Implementing a page segmentor that will work on machineprinted text or isolated lines of handprinted text is a matter of finding a line of data and sending it as an image to the appropiate line segmentor. Some problems that can arise are sloped lines of text, varying character height, and salt and pepper noise. The sloped line problem is solved by finding the four points that produce a bounding quadrilateral of the line of text. Once that quadrilateral

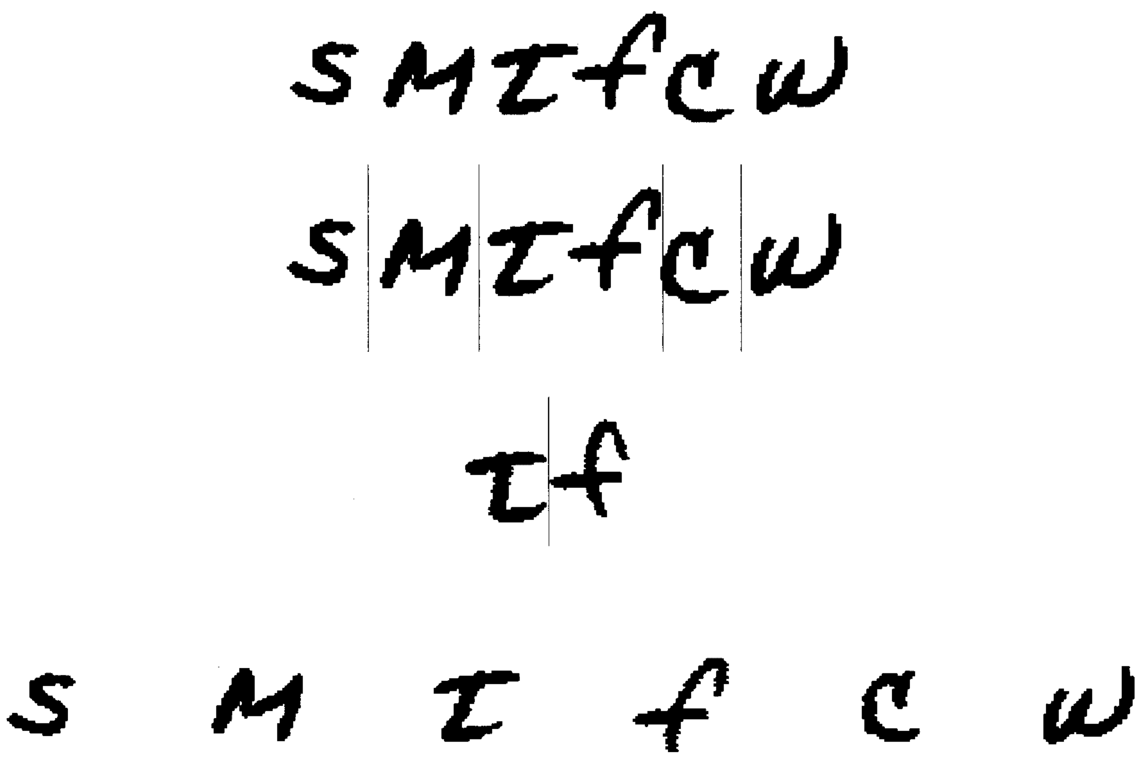
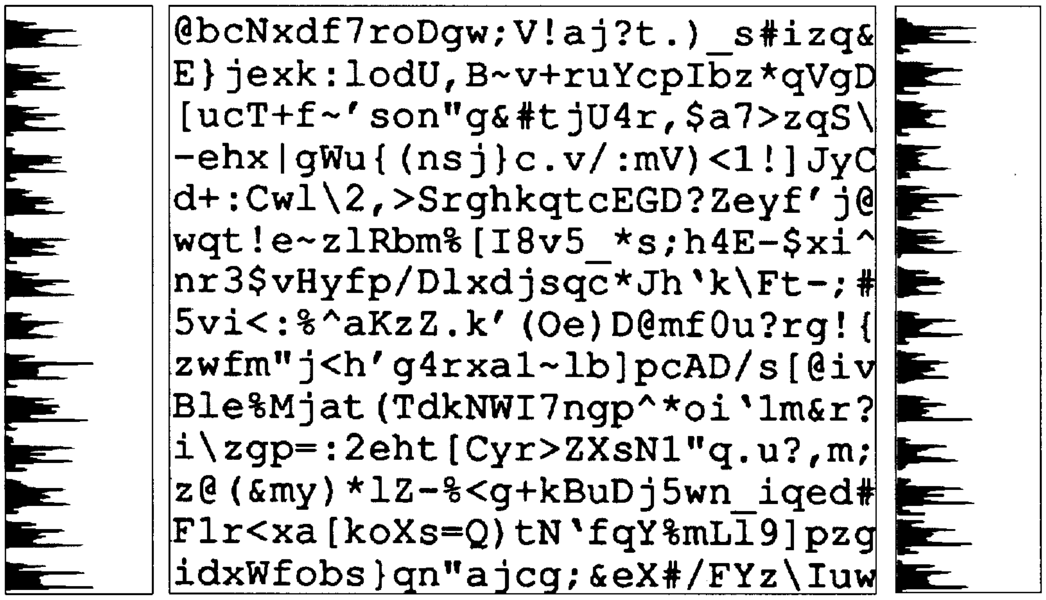


Figure 7: Example of cuts found using Third-Generation Line Segmentor

is found the image can be copied into a rectangular blank area which can encompass the quadrilateral and then treated as input into any of the previous line segmentors. The varying character height is no problem as long as all characters in a line are of similiar heights. The statistical adaptation rules are able to handle the varying heights from line to line because each rule is created for the current line being segmented. The use of thresholded spatial histograms reduces but does not remove problems caused by salt and pepper noise.

Finding a line of text on an image of a page consists of finding the bounding quadrilateral for the line of text. This is done by finding the left side of the line image's top and bottom and the right side's top and bottom. Spatial histograms can be used on a subimage of the image, about a 5 cm wide strip down each side, to find where the data is located. Figure 8 showns a subimage of a page with histograms on the left and right. The histogram to the left encompasses the first 2 centimeters of every row in the subimage, which is equivalent to six charcters in this example, while the one on the right encompasses the last 2 centimeters. Using subimages on the sides reduces the amount of work done to find the points and solves the sloped line problem. The use of separate histograms means that data found on each side must be matched up with the corresponding data on the other side. To accomplish this, an assumption is made that each line in the image of a page begins within the left 5 cm and ends within the right 5 cm. Now the data zones on the left and right should match one to one and are used to produce the desired quadrilateral. Once again, since each line is independent



```
@bcNxdf7roDgw;V!aj?t.)_s#izq&
E}jexk:lodU,B~v+ruYcpIbz*qVgD
[ucT+f~'son"g&#tjU4r,$a7>zqS\
-ehx|gWu{(nsj}c.v/:mV)<1!]JyC
d+:Cwl\2,>SrghkqtcEGD?Zeyf'j@
wqt!e~zlRbm%[I8v5_*s;h4E-$xi^
nr3$vHyfp/Dlxdjsqc*Jh`k\Ft-;#
5vi<:%^aKzZ.k'(Oe)D@mf0u?rg!{
zwfm"j<h'g4rxal~lb]pcAD/s[@iv
Ble%Mjat(TdkNWI7ngp^*oi`lm&r?
i\zgp=:2eht[Cyr>ZXsN1"q.u?,m;
z@(&my)*lZ-%<g+kBuDj5wn_iqed#
Flr<xa[koXs=Q)tN`fqY%mLl9]pzg
idxWfobs}qn"ajcg;&eX#/FYz\Iuw
```

Figure 8: Left and right histogram of a partial page

of the other lines, the use of a massively parallel computer allows for reduced computational time, since lines can be processed simultaneously.

Once the quadrilateral is found, the image is passed to a line segmentor. The image within the quadrilateral is copied. This copy is then given to a line segmentor to be segmented into character images. The line segmentor needs no information that the work is being done on only a part of the page.

7.1 Page Segmentor Limitations

There are several limitations to the page segmentor, all of which are related to finding lines of data. As with most image processing, noise can cause problems. The page segmentor is able to handle noise in several ways, such as thresholded histograms and statistically adapted rules. The rotation of the image can also effect the page segmentor. If the image is rotated too much in either direction, the histograms down the sides will not be accurate. This results in no data being found or incorrect creation of quadrilaterals. Any characters that extend below the baseline such as "j", "g", and "q" can be cut off when the quadrilateral is formed. This is rare; it was noticed only once in 10,000 segmented characters.

8 Results

The data used to test the segmentors is of two types, machineprinted and handprinted. The machineprinted text is a randomly generated page with uppercase letters, lowercase letters, digits, and 28 special characters. Each page has 60 examples of each uppercase letters, lowercase letters, and digit, and 30 examples of each special character. Several different examples of fonts were used. The fonts used include Courier, Helvetica, and Times Roman. The handprinted data used came from the *NIST Special Database 1*: Binary Images of Printed Digits, Alphas, and Text [8]. Only the fields of digits and alphas were used. This produced an example of each uppercase and lowercase letter for each writer and 13 examples of each digit per writer. The database contains 2100 forms, each by a different writer. All 2100 forms were used, first stripping the bounding boxes from the fields of interest, then being segemented.

The page segmentor, using the second generation line segmentor, was used to segment the randomly generated pages of machineprinted text. For the handprinted text the third-generation line segmentor was used on the fields that had been isolated.

Table 1 shows the comparison of performance for the three segmentors discussed. The third generation segmentor's statistics for hand print segmentation can be broken down into digits, upper case and lower case. The segmentor did best on digits at 90.3% accuracy followed by lower case at 89.6% and upper case at 88.2%.

Segmentor	Time Per Character	Machine Print Accuracy	Hand Print Accuracy
First Generation	30ms or less	97.9%	60.0%
Second Generation	30ms or less	99.2%	N/A
Third Generation	30ms or less	N/A	89.8%

Table 1: Segmentor Comparison Statistics

9 Conclusions

Simple image processing techniques which are computationally expensive become feasible when using a massively parallel machine. Most image processing; such as gathering statistics, computing histograms, and analyzing images, can be computed much faster when implemented on a parallel machine. The process of segmenting a textual image into character images is greatly improved when a parallel machine is used.

General knowledge about the problem domain can be used to increase the accuracy of a segmentor. Applying statistically adaptive rules using this general knowledge to the task of error correction has been shown to have significant positive affects on segmentor performance. Segmentation of images of machineprinted text was increased from 97.9% accuracy to 99.2% while images of handprinted text increased from 60% to 89.6%. The use of a massively parallel machine allowed processing speeds of 30 milliseconds per character to be achieved for all segmenting.

References

[1] M. D. Garris, R. A. Wilkinson, and C. L. Wilson, "Analysis of a Biologically Motivated Neural Network for Character Recognition," *Procedings: Analysis of Neural Network Applications*, ACM Press, pp. 160-175, May 1991

[2] C. L. Wilson. R. A. Wilkinson, and M. D. Garris, "Self-Organizing Neural Network Character Recognition on a Massively Parallel Computer," *Procedings of the IJCNN*, Vol. II, pp. 325-329, June 1990

[3] J. Fisher, S. C. Hinds, and D. P. D'Amato, "A Rule-Based System For Document Image Segmentation," *Proceedings 10th International Conference on Pattern Recognition*, Vol. I, pp. 567-572, 1990

[4] P. J. Bones, T. C. Griffin, and C. M. Carey-Smith, "Segmentation of Document Images," *Proceedings of the SPIE Image Communications and Worksations*, Vol. 1258, pp. 78-88, 1990

[5] K. Fukushima, T. Imagawa, and E Ashida, "Character Recognition with Selective Attention," *Proceedings of the IJCNN*, Vol. I, pp. 593-598, July 1991

[6] M. Yamada, and K. Hasuire, "Document Image Processing Based on Enhanced Border Following Algorithm," *Proceedings 10th International Conference on Pattern Recognition*, Vol. 2, pp. 231-236. 1990

[7] K. Ooi, H. Sasaki, and S. Ariyoshi, "Handwritten Character String Recognition System," *United States Patent Number 5,020,117*, May 28, 1991

[8] C. L. Wilson, and M. D. Garris, "Handprinted Character Database," *NIST Special Database 1*, **HWDB**, April 18, 1990