NISTIR 5894

Teaching Computers to Read Handprinted Paragraphs

Michael D. Garris

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Computer Systems Laboratory
Information Access and User Interfaces Division
Gaithersburg, MD 20899-0001

NIST

NISTIR 5894

Teaching Computers to Read Handprinted Paragraphs

Michael D. Garris

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Computer Systems Laboratory
Information Access and User Interfaces Division
Gaithersburg, MD 20899-0001

September 1996

U.S. DEPARTMENT OF COMMERCE
Michael Kantor, Secretary

TECHNOLOGY ADMINISTRATION
Mary L. Good, Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS
AND TECHNOLOGY
Arati Prabhakar, Director

Teaching Computers To Read Handprinted Paragraphs

Michael D. Garris (mgarris@nist.gov)

National Institute of Standards and Technology
Building 225, Room A216
Gaithersburg, MD 20899

ABSTRACT

A set of algorithms is presented that enables a computer to automatically read a paragraph of handprinted text. These steps are broken down into: 1.) isolating the lines of handprint; 2.) segmenting the lines into character images; 3.) classifying the character images; and 4.) spell-correcting the classifications. A robust method of line reconstruction is presented that avoids statistical outliers and generates a spatial map representing the center of mass of each handprinted line. Components are organized into lines based on their overlap and proximity to bands within the map. This facilitates the efficient detection and splitting of vertically touching characters across lines of text. A new character segmentor is also presented that adapts automatically to writing style. The segmentor composes characters from multiple components and it separates touching characters from single components. Comparing the performance of the segmentor between a 1-line response and a paragraph demonstrates that a person's writing has a tendency to become more varied and irregular as spatial constraints are relaxed. An optimized Probabilistic Neural Network is used to classify the character segments, and a description of a dictionary-based correction scheme is provided. Recognition accuracies are reported on 500 handprinted paragraphs from *NIST Special Database 19* containing the Preamble to the U.S. Constitution. The new methods of line reconstruction and character segmentation improve word recognition accuracy by 4% (from 60% to 64%) over a system that uses connected components directly as character segments. This paper represents a significant step toward a general purpose unconstrained handprint recognizer, and points out where future work may be focussed.

Keywords: connected components, dictionary-based correction, handprint, optical character recognition, probabilistic neural network, unconstrained paragraph

1. INTRODUCTION

The focus of this paper is on the technology required for a computer to read a paragraph of unconstrained handprint. Examples of applications that stand to benefit from this technology are the automated reading of handprinted messages sent via facsimile machine and the processing of forms that contain multiple-line responses to open-ended questions such as, "Please describe the sequence of events leading up to the accident." The task of reading an unconstrained paragraph of handprint can be broken into several steps as illustrated in Figure 1: 1.) the handprint is isolated into individual lines; 2.) the lines are segmented into individual characters images; 3.) the characters images are classified; and 4.) the classifications are corrected using a dictionary.

The National Institute of Standards and Technology (NIST) has developed a public domain optical character recognition (OCR) system, called the NIST Form-Based Handprint Recognition System, that serves as a vehicle for technology transfer to those developing recognition technology and integrating it into various applications [1]. The system is a full source code distribution written in 'C' and distributed on ISO-9660 CD-ROM, and it can be obtained free of charge by contacting the author. The public domain OCR system reads the handprint entered on Handwriting Sample Forms (HSF) like the one shown in Figure 2. A large collection of these forms (3669) has been scanned and is distributed on CD-ROM as *NIST Special Database 19* (SD19) [2]. HSF forms contain single-line fields containing handprinted digits and a randomly ordered lowercase and uppercase alphabet. The focus of this paper is on the last field on the form which contains the Preamble to the U.S. Constitution. This is a relatively unconstrained paragraph of handprint; there are no demarcations provided within the large box to guide the trajectories or the spacing of the handwriting; and as a result, the handwriting in these fields is spatially representative of typical paragraphs of handprinted text.

This paper presents a number of new algorithms for conducting the steps listed in Figure 1. The algorithms used for steps 3 and 4 are currently distributed with the first release of the NIST system, whereas the techniques for steps 1 and 2 (those dealing with the segmentation of a handprinted paragraph) represent new technologies that improve overall recognition performance. Section 2 presents a new method for reconstructing the line trajectories from a handprinted paragraph whose image was scanned off-line (independent from the time it was written). Section 3 describes a new method of character segmentation that automatically adapts to the style of writing being processed. Section 4 discusses character classification and Section 5 documents a technique for dictionary-based correction using a limited-size lexicon. Results from using the new methods of line reconstruction and character segmentation are compared to the performance of the original NIST public domain system in Section 6, and conclusions are drawn in the final section.

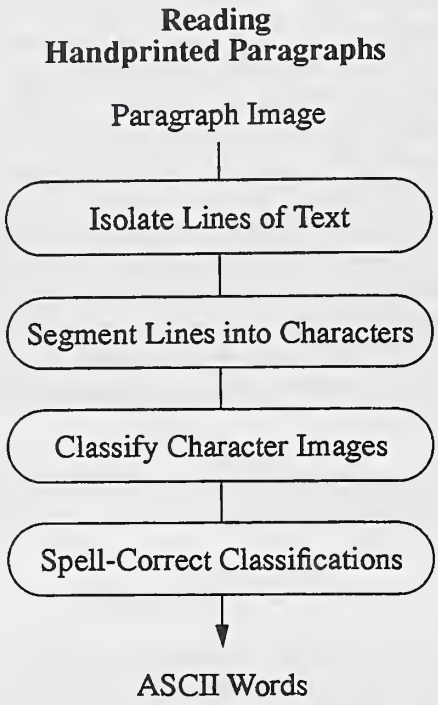


Figure 1. Steps to reading handprinted paragraphs with a computer.

2. ISOLATE TEXT LINES

In order to read a paragraph of handprinted text, its characters must be segmented and organized into lines. The approach taken in this work is bottom-up, beginning with the connected components in the image. A connected component is defined as a set of black pixels where each pixel is a direct neighbor of at least one other black pixel in the component. Eight-way connectivity is used in this application, where a neighbor is defined to be either directly adjacent or directly diagonal to a pixel. These components are sorted into lines of text and then segmented into character images. Connected components give reasonably good estimates of where the individual characters are in the paragraph. Some characters may be comprised of multiple components, while some components may contain multiple touching characters. Regardless, the connected components can be used to determine the positions of the lines of text within the paragraph, and they provide good initial character segmentations.

In principle, the lines of text can be represented by a graph connecting the nearest neighbors of horizontally adjacent components. However, handprinted text in the form of a paragraph is spatially unconstrained as there are no demarcations to guide the writer's line spacing and trajectories. As a result, the writing can be extremely varied. Lines within the same paragraph may bend and or slant at varying degrees of angle. In previous work, NIST developed a technique for tracking the trajectory of handprint when the image is captured off-line [3]. The technique (described in

Section 2.2) utilizes a search space defined by a cosine function (forming bubbles) to conduct a horizontally biased point-to-point search through the center points of the connected components within the text paragraph.

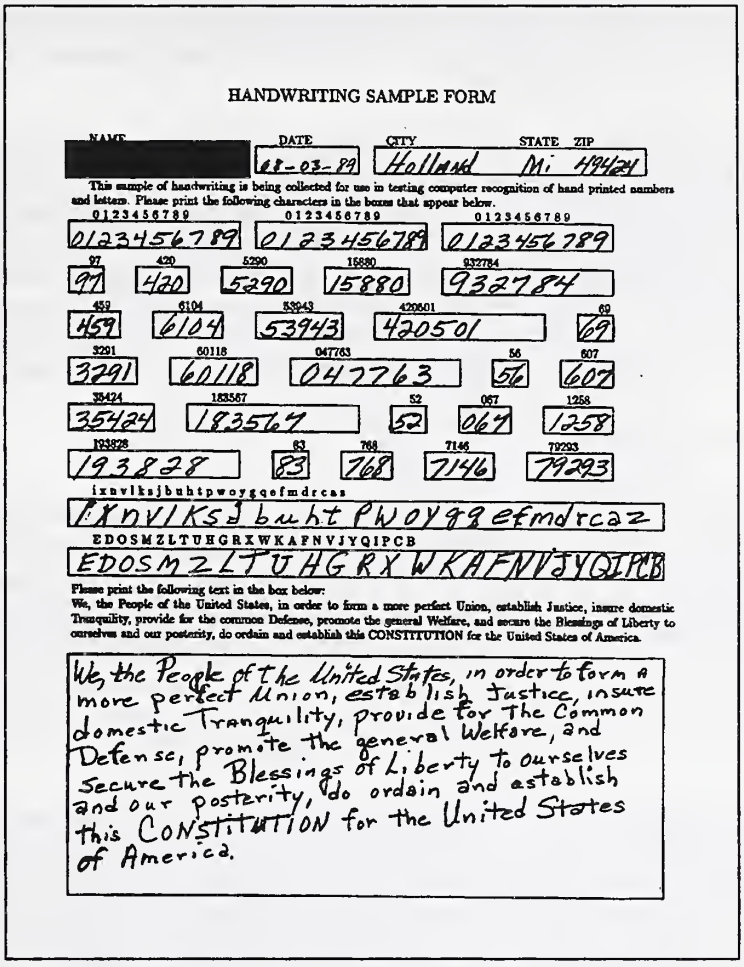


Figure 2. An example HSF form from SD19.

2.1 Computational Geometry Considered

In an attempt to improve upon the line reconstruction in [3], algorithms from more traditional Computational Geometry (CG) were considered. Papers like those presented by both Ittner [4] and Toussaint [5] give support for using CG algorithms to manipulate various sorts of image data for document analysis purposes. CG provides a suite of general purpose algorithms that organize data into various types of topological graphs. Originally it was thought that graphs such as the Voronoi diagram, Delaunay triangulation, and the minimum spanning tree may prove helpful in sorting the handprinted components into lines [6]. Seeking efficient implementations of these algorithms, a reference was found on the use of buckets that effectively reduces the time to construct the Voronoi diagram to that of $O(n)$ [7], whereas naive implementations of these types of graph-building algorithms operate in $O(n^2)$, and more clever algorithms still only operate in $O(n \log n)$ time [8].

Bucketing typically involves dividing up the data space into a uniform grid and assigning each data point to its respective spatial bin (or bucket). Data points are then subsequently referenced by their bucket address for coarse positional processing. Detailed positional processing can then be efficiently localized to only those data points located in closely neighboring buckets. Utilizing a bucket sort is most effective when the data is uniformly distributed. While

machine print is extremely uniform, handprint still retains some uniformity in terms of character and line spacing. A bucketing scheme was designed that divided the paragraph's image space into a grid of cells whose dimensions were based on the average width and height of the connected components. Rather than computing the Voronoi Diagram, etc. a simple nearest-neighbor method based on processing adjacent buckets was used to construct a Nearest-Neighbor (NN) graph from the center points of the components. The algorithm executed very efficiently. The construction of the NN graph from one of the Preamble paragraphs in SD19 took approximately 2 seconds on a Sun Microsystems SPARCStation 2 with an 80 MHz Weitek CPU[1].

The results of the NN-graph were then visually compared to the lines reconstructed using the older bubble-based point-to-point search. While the old method took on about 2.5 seconds a paragraph, its results were determined to be significantly better. The generation of the NN graph was indeed quicker, but unfortunately it frequently joined characters from neighboring lines, thus proving that NN distances alone are not sufficient to reliably organize handprint into text lines. It was concluded that considerable postprocessing would be required to satisfactorily clean up the NN graph to correctly represent text lines thus nullifying any increase in speed afforded by the bucketing scheme. At this point it was decided to focus work on improving the older point-to-point search as it utilizes nearest neighbor relations in conjunction with certain expectations of localized line trajectory based on simple writing style attributes.

2.2 Previous Point-to-Point Search

The location of each connected component (or blob) is identified by computing the geometric center of its smallest bounding rectangle. The result is a 2-dimensional grid of blob centers that can be used to represent the line trajectories of the handprinted text. A technique for searching the 2-dimensional grid of blob centers was developed that takes into account local writing fluctuations to sort the blobs into correct reading order. Reference [3] provides a detailed description of this technique which conducts a point-to-point search using a local search space defined by the interior of the function:

$$S = a\cos(b\theta) \qquad -\frac{\pi}{2b} < \theta < \frac{\pi}{2b} \tag{1}$$

This function, which is similar to an antenna sensitivity model, forms a tear-drop shaped bubble that is desirable for this application because it is horizontally biased. At values of b near 0.1, the function's shape is circular, and as b increases the shape continuously forms into a tear-drop. The variable, a, controls the length of the bubble along its horizontal axis of symmetry. By increasing a, the length of the bubble is increased and the search is extended. The variables, a and b, are controlled using a linear function defined by $\bar{h}$ (the average height of the blobs in the paragraph). Using the linear control function, the size and shape of the bubble can be continuously modified as shown in Figure 3. In these three examples, average blob heights of 16, 32, and 48 are used, respectively.

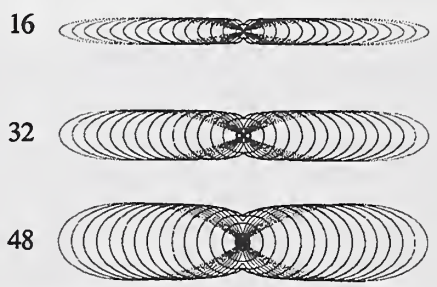


Figure 3. Adaptation of bubbles to the writer's average component height.

1. Specific hardware and software products identified in this paper were used in order to adequately support the development of the technology described in this document. 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.

The search begins from the blob center closest to the top-left corner of the paragraph. The blob is added to an empty list, and a bubble is initialized, tested, and then grown via incrementing *a* until either a neighboring blob center is found or *a* exceeds a threshold (300 pixels). In general, the new blob is added to the list and the search resumes from the center of the new blob. However, if the new blob's center does not meet given criteria as described in [3], then its center is added to the list, but the search continues from the current blob center and does not advance to the center of the new blob. This way the system does not naively follow erratic line trajectories, minimizing the chances of crossing over into adjacent lines.

During the development of this approach, it was determined that as local fluctuations in the handprint become excessive (rather than force the system to make a *guess)* the point-to-point search should be preempted. The search is restarted from top-leftmost blob not yet assigned to a line in the image. This action is also taken at the end of a text line when no new neighboring blob centers are found to the right of the current blob. Each restart involves starting a new list, and the entire search process is terminated when every blob in the image has been assigned.

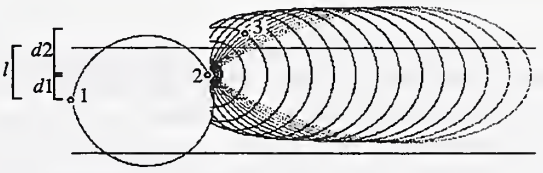


Figure 4. Heuristic used to preempt the search.

The criterion for preempting the search and beginning a new list is illustrated in Figure 4. In this illustration, the search is currently being conducted from blob center (2), and (3) has been located as the next nearest neighbor. The two parallel horizontal lines in the diagram represent the area bounded by $-\bar{h}$ and $+\bar{h}$ centered about the previous blob center (1). If the vertical distance between (1) and (3), ($d1 + d2$), exceeds the limit l, then the search is preempted and (3) is not added to the current list. Blob (3) is left unassigned so that it can be added to a list later in the search process.

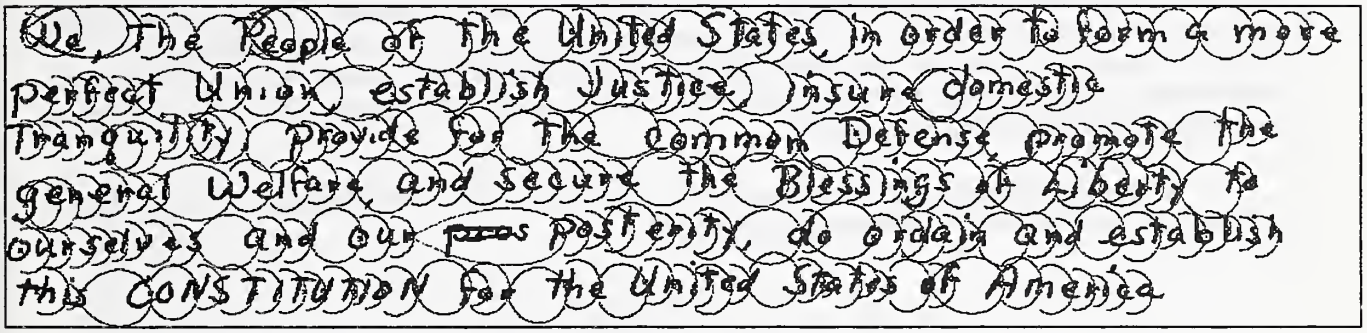


Figure 5. Traces of the bubbles used to sort the connected components into lines.

The point-to-point search produces multiple lists of connected components. Some of the lists represent complete lines of text, and other lists may represent only fragments of the text lines printed. A final merging process is required so that only lists containing complete text lines remain [3].

2.3 Line-Band Image Map

The line reconstruction described above, although an improvement over a plain NN graph, is still too susceptible to including characters from other lines and crossing lines. Just as important is the fact that the method does not detect when two lines touch (i.e. characters across two lines touch each other vertically). In its original form, the point-to-point technique has no way to divide these touching lines.

Improvements contributed by the work in this paper are primarily accomplished by encapsulating the old point-to-point method into a more sophisticated process that generates an image-based map. The one fundamental change actually made to the original method is the use of one additional preemption heuristic. From Figure 4, if the distance between points (2) and (3) is greater than $\overline{h}$, then the search is preempted. The remainder of the original method is directly used without change.

In order to improve the quality of extracted lines, the connected components are sorted into three categories: small, normal, and tall. A component is too small if it is dot-sized which is defined to be

If $(w < (2 \times esw))$ && $(h < (3 \times esw))$ then Dot

where w is the pixel width and h is the pixel height of the component and esw is an estimate of the stroke width (discussed later). This allows a small diagonal stroke to be considered too small. A component is considered too tall

If $(h >= (ech \times 1.5))$ then Too Tall

where ech is an estimate of the height of the handwriting (also discussed later). Those blobs determined to be too small or too tall are ignored when conducting the point-to-point search. In this way, only those components considered to be typical in height are processed, eliminating statistical outliers from the search. As the point-to-point search proceeds, one additional category of components is collected. Those components that cause the preemption heuristics associated with Figure 4 to trigger are collected into a category known as problems. This prohibits the erratic steps that cause preemption from being added to the line trajectories. Much more reliable trajectories are generated as a result of filtering out these deviant cases. An example of the resulting piecewise linear trajectories is plotted in Figure 7. These results are computed from the image shown in Figure 6. Notice the component center points that are labeled small, tall, and problems.

These piecewise linear trajectories estimate the vertical center of each line of handprint. In order to determine which connected components belong to which lines, some notion of distance from a component to its neighboring lines must be computed. Measuring the distance from the center point of a blob to the vertical center of a line is not a good characterization of the component's true position relative to the line. A more natural measurement is based on overlap. To conduct such measurements, the center trajectories of each line of text are expanded to an envelope of space that represents the center of mass of the line. In this way, the piecewise linear trajectory is expanded to include a spatial band that encompasses the majority of the handprinted text along the specific line. Percentages of overlap and distances are computed in relationship to these bands.

To construct these line-bands, the y-coordinates from the tops of every component linked along a line are smoothed using the following 3-point smoothing function:

$$\hat{y}_i = \left\lfloor \frac{y_{i-1} + (2.0 \times y_i) + y_{i+1}}{4.0} \right\rfloor \qquad (2)$$

The span between each successive pair of smoothed top points is linearly interpolated. The bottoms of every component linked together are similarly smoothed and interpolated. The resulting area between the smoothed and interpolated tops and bottoms forms the envelope representing the mass of the line. Smoothing reduces the effect of large descenders, punctuation, and isolated erratic writing. An example of the line-bands computed for the image in Figure 6 are is shown in Figure 8. Each line's band in the image is filled with a unique pixel value representing its line index. The left and right ends of each band are extended horizontally to the edges of the image or until the band intersects an already existing band. This resulting image is called a *line-band map*.

We, the People of the United States, in order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the Common Defense, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our posterity, do ordain and establish this CONSTITUTION for the United States of America.

Figure 6. Example of a paragraph whose lines are written with a pronounced slant.

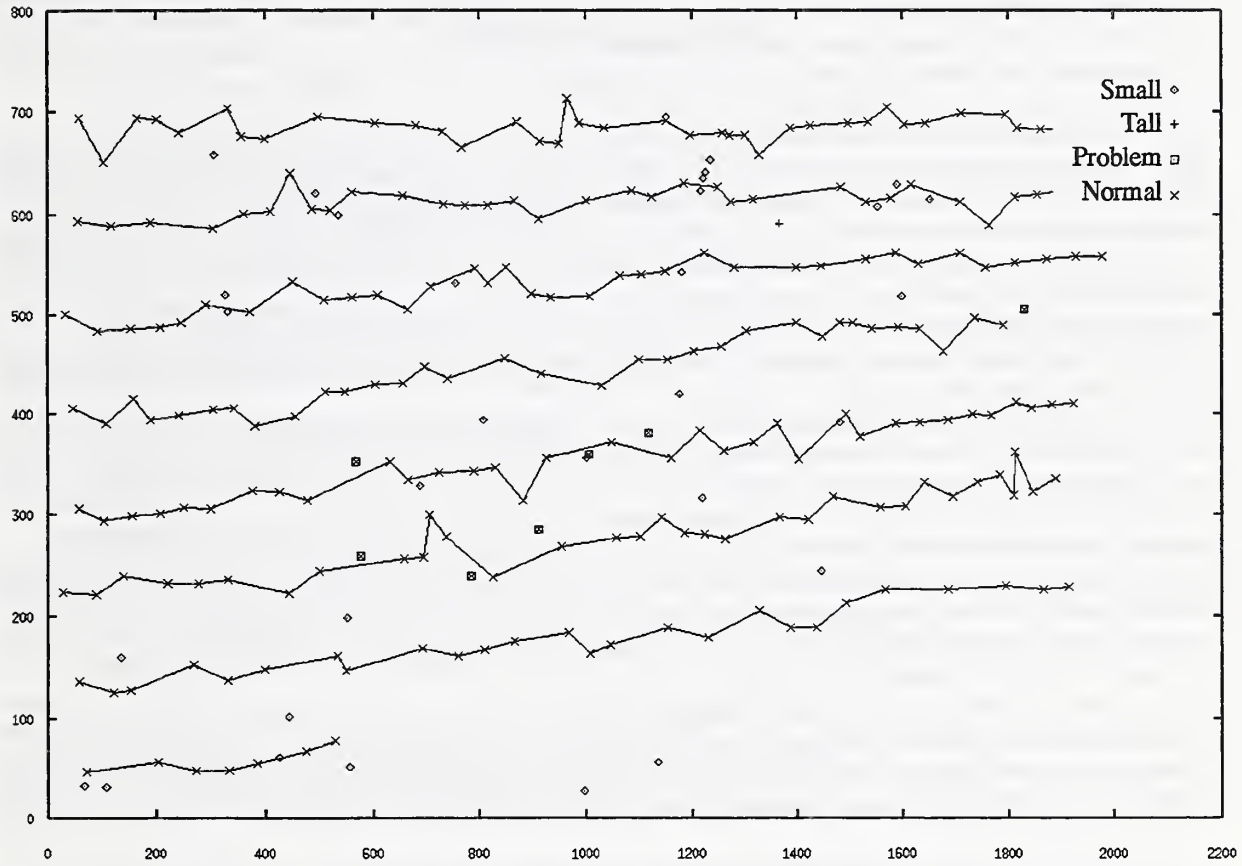


Figure 7. Component centers from Figure 6 linked into lines of text.

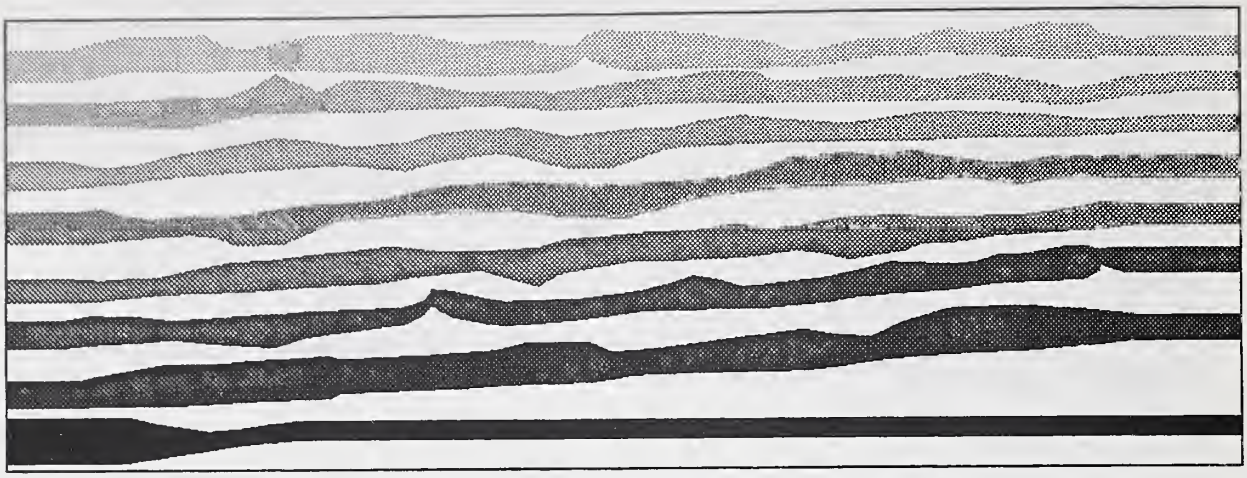


Figure 8. The line-bands computed for the image in Figure 6.

All the components in the image (including those labeled small, tall, and problems) are analyzed with respect to the line-band map. Each component is overlaid on the map in its original position and used as a mask. A histogram is calculated, totalling (by unique band index) the number of overlapping black pixels in the component. For example, if a component overlaps bands 3 and 4, then the histogram will contain the total number of black pixels in the component that overlap with band 3, and separately it will contain the total number overlapping band 4.

If a component is sufficiently tall (greater than the 90th percentile of heights of all the components in the paragraph image) and overlaps two or more line-bands, then the component likely contains two vertically touching characters that belong to different lines. Single characters typically overlap neighboring bands from the top extending downwards, such as with the drawing of long descenders. It was observed that most characters belonging to a specific band will almost entirely overlap that band vertically. Therefore, the following test was designed. If the component overlaps two or more bands, and the lower band overlaps a substantial part of the component (greater than 12.5% of the component's black pixels), and the component vertically overlaps a significant portion of the lower band (75% of the entire bands height), then the component is most likely two vertically touching characters and it should be split.

A tall component is split across multiple lines by dividing it vertically into relatively equal pieces. Given n overlapping bands, n-1 cuts are initiated at successive increments of (h/n), where h is the height of the entire component. To make the cuts, the component image is rotated 90° on its side, and starting at the first cut position, a search (or trace) is initiated from the top of the rotated component downwards and from the bottom of the component upwards. The trace is free to fall vertically until it hits the edge of a character at which point it flows along the contour falling progressively downwards until either the bottom of the image or a local minima is reached. The trace upwards performs similarly in the opposite direction. The character is then cut at the minimum distance between these two trace paths. Each piece cut from the original component is rotated back and assigned to its respective line-band.

Two pairs of characters touch across the lines in Figure 6. There is the 'p' in "peo*p*le" touching the 'f' in "per*f*ect", and the 'h' in "establis*h*" touching the 'f' in "*f*or". Both of these cases are automatically detected and the results of the split are shown in Figure 9. The segmentation on the right is much better than that on the left. The development of more robust methods for separating vertically touching characters is left for future work.

If the component is not sufficiently tall to be considered two vertically touching characters, but it still overlaps one or more line-bands, then the component is assigned to the band of maximum (total black pixels) overlap. If the component overlaps no bands, then the distance is measured from its center point to the edge of the nearest band above and below. Taking the minimum distance, the component is assigned to the closer of the neighboring bands. At this point, all the components in the image have been assigned to their respective lines. The components are sorted within each line on their center x-positions, and the isolated lines of handprint are ready to be segmented into character images.

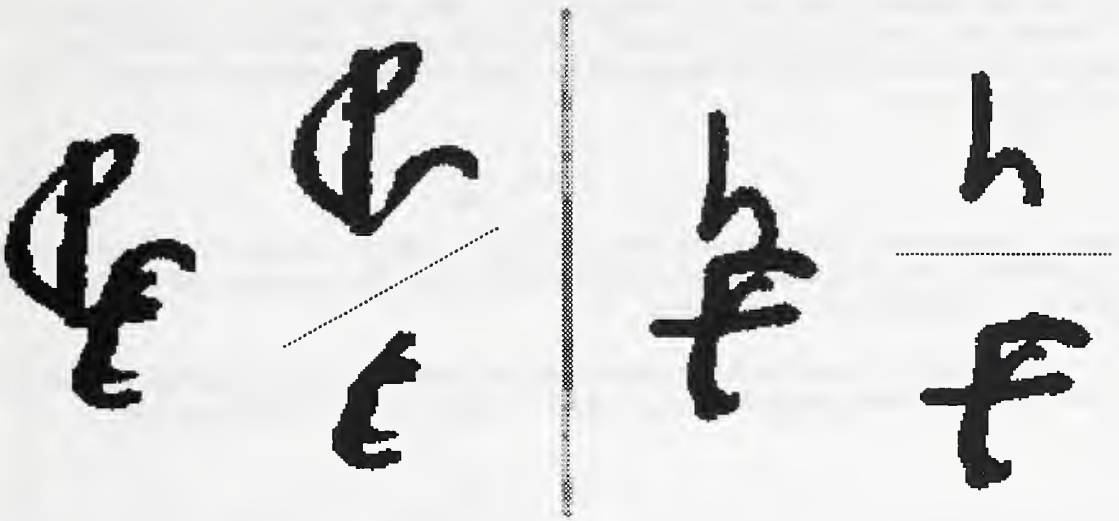


Figure 9. Results of splitting vertically touching characters.

3. SEGMENT CHARACTERS

In reference [9], a method for constructing character segments of handprint from connected components is documented. The method uses simple statistical features to adapt to variations in writing style. The method was successfully applied to the single-line responses on the HSF forms in SD19, where the same technique was tested on sequences of digits, lowercase, and uppercase characters. The method is able to compose single characters from multiple components, and it splits components of multiple touching characters. In light of its successful application, it was anticipated that the segmentor would work well in this study for segmenting the isolated lines of handprint.

The segmentation method as described in [9] was used to segment the lines of handprint from a number of different Preamble paragraphs. Upon visual inspection of the resulting segments, it was decided that the method in its original form does not perform satisfactorily on handprinted paragraph lines. It was determined that the heights of characters within the Preamble paragraph have higher variance than those written in the 1-line fields on the HSF form. It is concluded that variations in writing significantly increase as spatial constraints are relaxed. This increase in variation makes it difficult for the original segmentation method to accurately model the writing style in the paragraphs.

The writing style is characterized by estimating the stroke width (*esw*) and the height of the handwritten characters (*ech*). The estimated stroke width is computed as the median horizontal run length within the image of isolated handprint. The writing height, ech, is used to estimate (not the height of each individual character, but rather) the maximal height of the writing as a whole, similar to measuring the point size (or the height of uppercase characters) in a typical machine print font. With the original studies on single-line responses, the maximum height of the components in the field was sufficient for this estimate. Due to the higher variance in height and the increased chance of substantial outliers in the paragraph's characters, the maximum no longer holds as a good height estimate. Experiments were run, and it was determined that taking the 90th percentile height from among the components in the line is sufficient to characterize the height of the handprint. This improved the performance of the segmentor, but still not to a satisfactory level.

According to the segmentation method in [9], the black pixels in an image can be thought to be part of larger meta-pixels that are basically one stroke width square. This larger pixel is referred to as a *standard stroke pixel*. Extending this notion further, a *standard stroke area* can be defined as ($esw \times ech$). By their very definition, characters are configurations of a finite and relatively small number of strokes. Measuring the image in terms of standard strokes provides a relative measure of density that adapts to the handwriting style itself, thus allowing for writer-normalized units that characterize or model the style of handprint for the purposes of segmentation.

A connected component may contain multiple touching characters, and one must be able to detect to separate the characters. Detection must be simple to compute, relatively reliable, and adaptable to the ch of a specific writer. To accomplish this, two simple features are computed. One feature, *standard stroke* sures the density of the black pixels within the component image in units of standard stroke area. The sta count (*ssc*) is defined as

$$ssc = \frac{p}{ssa} \tag{3}$$

where p is the black pixel count of the component and *ssa* is the standard stroke area. This measures how many standard strokes can be constructed from the total density of black pixels in the component, and its value automatically adapts to the characteristics of the handwriting.

The other feature computes the *aspect ratio* of the component. The larger a component's width is to its height, the more likely it contains multiple characters. Originally, the aspect ratio (*ar*) was computed as

$$ar = \frac{w}{ech} \tag{4}$$

where w is the width of the component. This is slightly different than the true aspect ratio of the component because it uses the standard unit height, *ech*, rather than the actual height of the component.

In reference [9], a training set of components containing single and touching characters extracted from 1-line fields was used to compute a scatter plot of (*ar*, *ssc*) feature points. As can be seen in that report, there is significant separation between the classes of single and touching character images, and a linear discrimination function was empirically derived to separate the classes. This linear function is not only used to detect touching characters, but it is also instrumental in directing the splitting of the characters. Details of how this takes place are provided in the reference.

Due to poor segmentation performance on the lines extracted from paragraphs, a training set of single and touching character components was extracted from the first 20 Preamble paragraphs in SD19, and the scatter plot analysis was repeated on these new components. The separation seen previously was no longer as prominent with the new component data. Upon reflection, it was determined that it was inappropriate to compute *ar* by dividing by *ech* due to the widely varying heights of characters in the paragraph. A second scatter plot analysis was conducted where the true aspect ratio of the component was used (i.e. *ar* was computed by dividing the component's width by its actual height).

In this case, *ar* is truly relative to the component itself while *ssc* remains in units based on the writing style. As a result, the feature points when plotted are spaced further apart, and the separability of the single and touching classes is enhanced, but as can be seen in Figure 10 there remains significant overlap between the classes. Each of the 20 training paragraphs were examined independently, and it was observed that some writers' handprint exhibited greater separation than others. In other words, the overly simplistic handwriting style model used in the segmentor characterizes some writers better than others.

The lower dashed line displayed in Figure 10 plots the linear discrimination function used in [9]. Those components whose feature points lie below the detector line are assumed to be single characters, and those above the line are determined to contain multiple characters. Using the original detector line on the data derived from the Preamble paragraphs causes many single characters (plotted as diamonds) to be incorrectly detected and split as multiple touching characters. Upon careful inspection, a new linear function (plotted as the solid line) was derived which strikes a compromise. The tension requiring compromise can primarily be attributed to the fact that the handprinted paragraph contains large uppercase letters such as 'W's, and it contains numerous small lowercase characters. The disparity between the size of large and small characters can in itself vary greatly depending on the writing style. Keep in mind that many combinations of these characters may touch. Ideally, we would like to detect when even two very small characters touch, but this would require (using this limited model) miss-identifying large single characters as multiple touching characters. So, the solid detector line in the Figure 10 attempts to detect as many touching characters as pos-

sible while preserving the larger single characters. As can be seen, the separation of single and touching classes is less than desired. Improving on this model is left for future work.

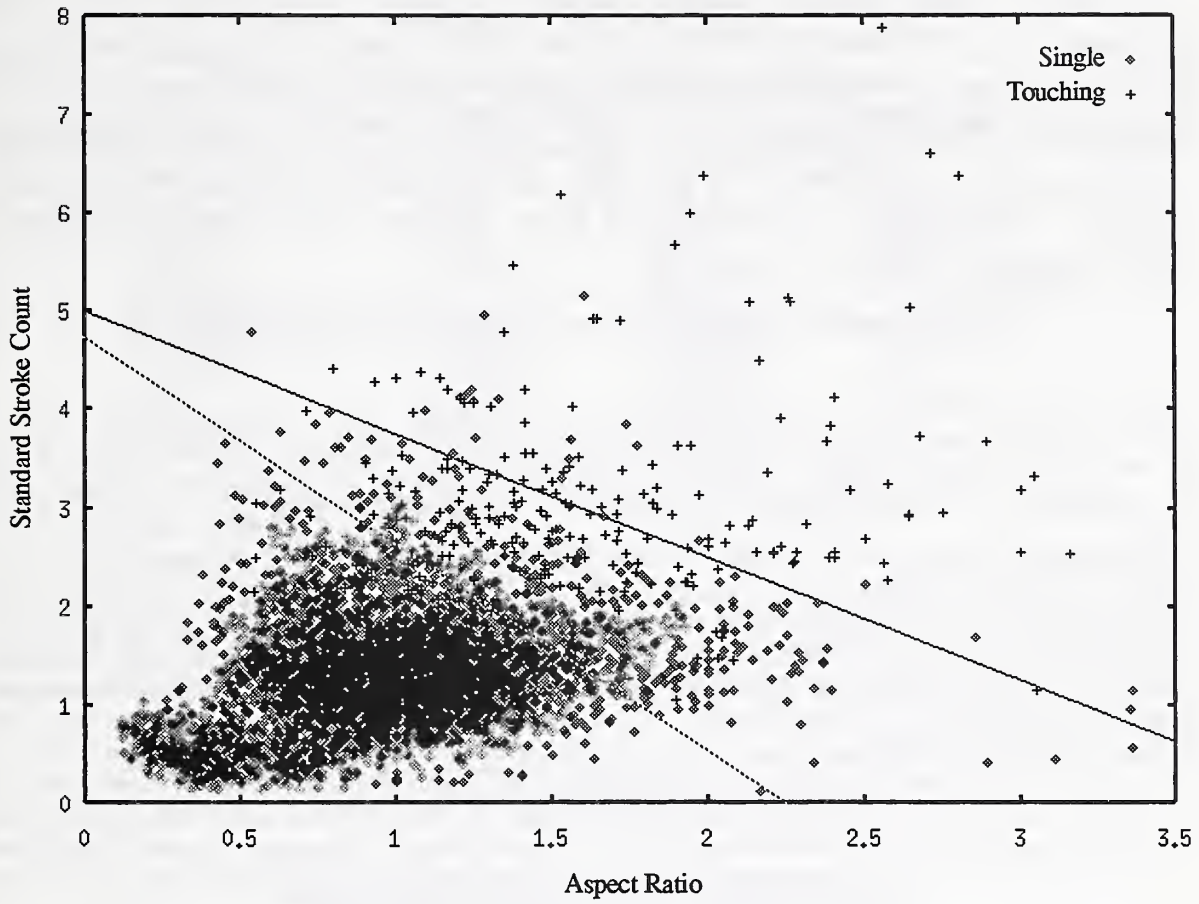


Figure 10. Scatter plot of (ar, ssc) feature points derived from Preamble paragraph characters. (Dashed line is old detector function. Solid line is new detector function.)

An example of the segmentor's performance on a paragraph containing a significant number of touching characters is show in Figure 12. In this example, the segmented images have been scaled down slightly and pulled apart to emphasize the segmentation results. Many of the horizontally touching characters have been separated. Notice the numerous vertically touching characters have also been detected and split.

4. CLASSIFY CHARACTERS

Once character segments are created they are slant and size normalized to be 32×32 pixels as described in reference [1]. The Karhunen Loève (KL) transform is then computed on the normalized character images [10]. The KL transform performs dimension reduction and acts as a spatially localized low-pass filter. With the transform, this application used the 64 top-ranked eigenvectors to compute, from the normalized character images, feature vectors containing KL coefficients which were then classified using an optimized Probabilistic Neural Network (PNN) [11]. The PNN implementation used takes advantage of a k-d tree [12] preclassification search that significantly reduces the number of prototype vectors used to compute the PNN discrimination function. This greatly decreases the computation time required [1].

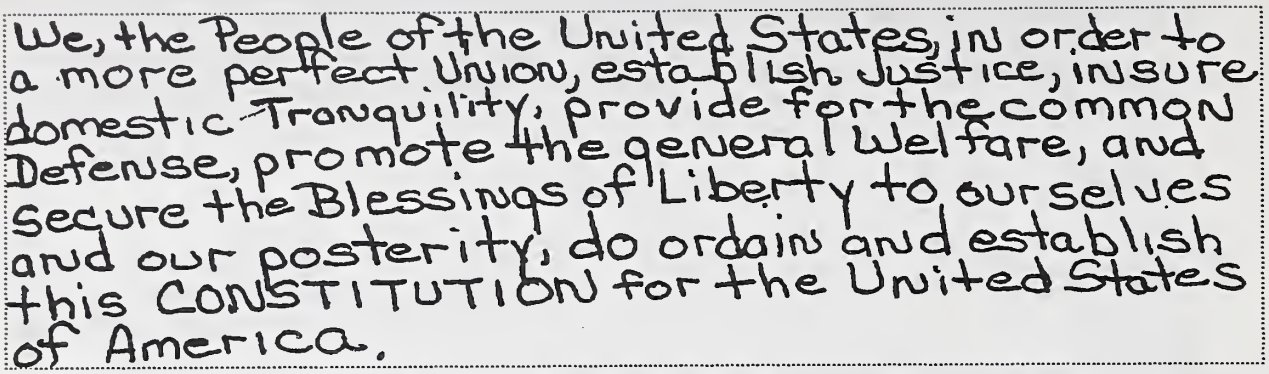


Figure 11. Example of a paragraph with many touching characters (vertically as well).

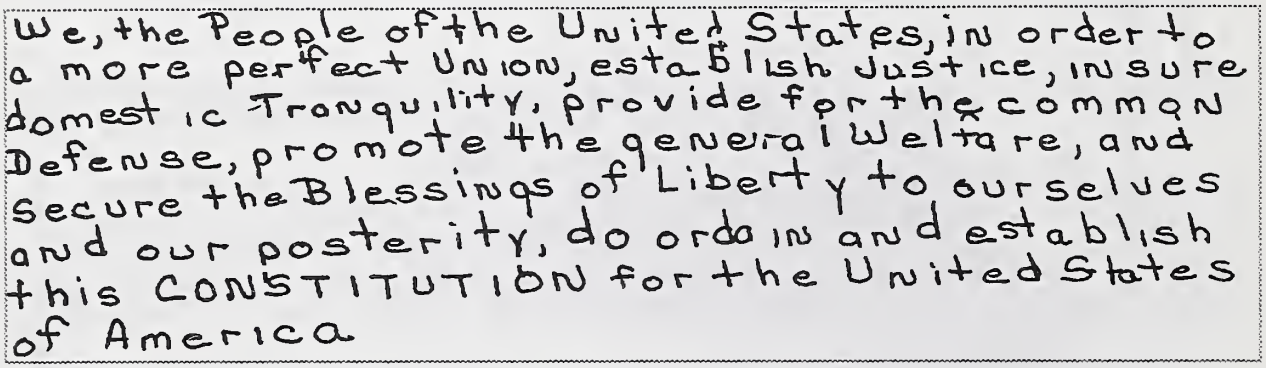


Figure 12. Character segments created from Figure 11.

WEJTHEPEOPIEOPTHEUNITEASTATFSJLNORDERTO
AMOREPQRFKTUNIONJEBTAEIBHJUSTICEJINSURE
DOMDLCITRONGUIIJTYIPROVIDEFPRTHFCOMMQN
DEFENBELPROMOTETHEGENEMIWELNRELAND
SECURETHEBLCSSINPOFLIBBHYTOOURSELUES
ANDOURPOSTERLTYIDOORBINANDQDADLISH
THISCONETITUTIBNFORTHEUNIFEDSBTES
OFAMERICA

Figure 13. Classified text from segments in Figure 12.

Once classified, the paragraph is represented by a single string of ASCII letters as shown in Figure 13. (Note that the line breaks in the figure were artificially added so that the text corresponds to the original image.) The text in this figure was recognized from the segments shown in Figure 12. As can be seen, there are no inter-word gaps identified. Also, no attempt is made to remove punctuation marks (i.e. commas and periods) which are most frequently labeled as spurious 'I's, 'J's and 'L's. The task is left to the lexical postprocessing to filter out the spurious characters, identify word gaps, and correct for the incorrect segmentation and missclassification of other characters.

Characters extracted from the upper and lowercase alphabet fields written by the 1000 writers in hsf_4 and hsf_6 of SD19 were used as training prototypes. All together, the training set contained 48,625 labeled character images. The classifier was trained to generate case insensitive output. To do this, both upper and lowercase examples of each letter were merged together as a single class. For example, the PNN was taught to classify both handprinted 'a's and 'A's as the single class 'A'. This is a fairly difficult classification problem which is reflected in the quality of the recognition shown in Figure 13. Other training schemes may prove more successful. For example, the classifier

could be trained to make distinct lower and uppercase classifications except for those characters where both cases have similar shape (ex. 'o' and 'O', 'c' and 'C'). Experimenting with different training schemes is left to future study.

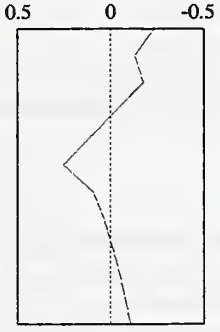
5. LEXICAL POSTPROCESSING

The Preamble to the U.S. Constitution is comprised of 38 unique words, and these words are used to construct the dictionary (lexicon) shown in Figure 14. The lexicon is used to detect words within text lines, identifying word boundaries and correcting any segmentation and classification errors existing within the text lines.

A	FOR	ORDER	THE
AMERICA	FORM	OUR	THIS
AND	GENERAL	OURSELVES	TO
BLESSINGS	IN	PEOPLE	TRANQUILITY
COMMON	INSURE	PERFECT	UNION
CONSTITUTION	JUSTICE	POSTERITY	UNITED
DEFENSE	LIBERTY	PROMOTE	WE
DO	MORE	PROVIDE	WELFARE
DOMESTIC	OF	SECURE	
ESTABLISH	ORDAIN	STATES	

Figure 14. Lexicon constructed from the text contained in the Preamble to the U. S. Constitution.

An illustration of the method of dictionary-based correction used in this study is shown in Figure 15. A detailed description of this method is found in [3]. In the figure, a portion of classifier output, "STCTESLNORDE", is being processed. The graph plots the floating point numbers listed in the first column. These numbers form a signal which is processed in order to locate words within the text. The second column is a fan-out of hypothesized words beginning with the character 'S' and adding one successive character from the text line forming a new hypothesized word on each row down the column. The maximum length of a hypothesized word is equal to the length of the longest word in the lexicon, "CONSTITUTION". The third column lists the best match from the lexicon for each hypothesized word in the second column. The fourth column lists alignments that are produced using the Levenstein Distance [13] to match the hypothesized word to the lexicon match. In the alignments, 0 represents a correct character, 1 represents a substituted character, 2 represents an inserted character, and 3 represents a deleted character. These alignments are used to generate the signals listed in the first column and plotted in the graph.



Signal	Hypothesis	Match	Alignment
-0.230	S	THIS	2220
-0.127	ST	STATES	022022
-0.174	STC	STATES	022021
-0.031	STCT	STATES	001022
0.111	STCTE	STATES	001002
0.254	STCTES	STATES	001000
0.096	STCTESL	STATES	0010003
0.040	STCTESLN	STATES	00100033
-0.005	STCTESLNO	STATES	001000333
-0.043	STCTESLNOR	STATES	0010003333
-0.075	STCTESLNORD	STATES	00100033333
-0.103	STCTESLNORDE	STATES	001000333333

Figure 15. Signals generated from a fan-out of hypothesized words.

A signal value, s, is computed from two terms, e and t. The first term, e, is an error term and is computed:

$$e = \frac{n}{l - g} \tag{5}$$

where n is the number of errors (1's, 2's, and 3's) in a hypothesized word's alignment, l is the total number of characters in the alignment, and g is the number of contiguous groupings of 1's and 3's. The Levenstein Distance strictly mini-

mizes the amount of error in the alignment without regard for the resulting configuration of alignment elements. The variable g is used to favor hypothesized words whose alignments contain contiguous groupings of correct characters (0's) over alignments containing many discontinuities. The second term, t, is a translation term based on a linear function that biases longer hypothesized words over shorter ones [3]. In this way, matches to the word "DOMESTIC" are favored over matches to the word "DO", and "INSURE" is favored over the word "IN". Signal value, s, is then computed:

$$s = 1.0 - e - t \tag{6}$$

The signals listed in the first column of Figure 15 are searched top to bottom. Only those hypothesized words with $s>0$ are considered to contain possible words. All other hypothesized words in the fan-out are ignored. The hypothesized word with the largest signal strength is selected. If this word is a substring of a hypothesized word further down the list, such as "DO" in "DOMAIN", and the word containing the substring has a signal strength, $s>0$, then the longer word is selected in place of the word with maximum signal.

Once a hypothesized word is selected from the fan-out, a new fan-out is built beginning with the first character to right of last correct character in the selected word's alignment. (Actually, there is some recursive processing of the alignment required first; these details are provided in [3].) In Figure 15, the hypothesized word, "STCTES" is selected with a maximum signal of 0.254. The next fan-out begins at 'L', starting from the position in the text line, "LNORDE...". If no hypothesized words are selected within the current fan-out, then the processing advances one character in the text line, and the fan-out begins from that point.

Through this approach, segmentation and classification errors are corrected, and word boundaries are automatically identified. An example of the results of dictionary-based postprocessing can be seen in Figure 16. These results were obtained from the raw classifications shown in Figure 13. Although not perfect, these results have been significantly improved through the use of the line-band map and adaptive segmentor.

WE THE PEOPLE THE UNITED A STATES ORDER TO
A MORE UNION THE JUSTICE INSURE
DO TRANQUILITY PROVIDE FOR THE COMMON
DEFENSE PROMOTE THE GENERAL WELFARE AND
SECURE THE BLESSINGS OF LIBERTY TO OURSELVES
AND OUR POSTERITY DO FOR IN AND A
THIS CONSTITUTION FOR THE UNITED STATES
OF AMERICA

Figure 16. The results of dictionary-based postprocessing on the raw classifications shown in Figure 13.

6. RESULTS

As mentioned in the introduction, the techniques presented in Sections 2 and 3 (isolating the lines of handprint and segmenting the lines into character images) are new. In order to evaluate their impact on recognition, these new methods were integrated into the NIST public domain Form-Based Handprint Recognition System and run across the HSF forms from the first 500 writers (hsf_0) in SD19. This section compares the results of the original system against those achieved using the new techniques. Both versions of the recognition system used the PNN classifier described in Section 4 and the dictionary-based correction in Section 5. Methods for form registration, form removal, and field isolation (steps related to forms processing, but not specifically germane to reading unconstrained paragraphs) were also the same between the two systems, and are described in reference [1].

The recognized text from both the original and augmented recognition systems was scored using the NIST Scoring Package [14]. In order to generate word-level accuracies, the recognized text was tokenized so that each unique word in the Preamble was represented by a single identifying character. The scoring package aligned the sequences of recognized work tokens against a string of reference tokens (the ground truth), accumulating the number

of substituted, inserted, deleted, and correctly recognized words. The original public domain system achieved a word accuracy over 60% (15439/25532), where as the system using the new techniques achieved 64% (16343/25532). The original NIST public domain OCR system uses connected components directly as character segments, so the system has no way of dealing with touching lines, touching characters, and fragmented characters. Using the new methods for line reconstruction in conjunction with the new character segmentation improved overall word recognition by 4%. Section 3 discussed how the performance of the new character segmentor is somewhat compromised in its ability to detect and split touching characters. Running the augmented version of the system without character splitting (within the lines) resulted in an accuracy of only 61.4% (15664/25532). Therefore, using the new method to divide horizontally touching characters, even with its limitations, does significantly contribute to improved recognition.

In terms of execution time, the extraction of connected components took on average 1.4 seconds per Preamble paragraph, the isolation of text lines took on average 1.0 seconds; and segmenting of the lines into character images took only 0.24 seconds. In all it took on average 2.6 seconds to convert the paragraph image into isolated character images. The original NIST system, which was coded less efficiently and used less robust algorithms, took on average 2.9 seconds (2.7 seconds to extract the components and 0.23 to isolate the text lines) to convert the paragraph image into isolated characters. There are 268 alphabetic characters in the Preamble, so the new methods working together generate character segments at a rate greater than 100 characters/second. For comparison, the PNN classifier takes on average 4.4 seconds per paragraph. The times of all the other steps pale in comparison to the dictionary-based correction which requires 37.4 seconds per paragraph. All the times reported in this section were computed on a Sun Microsystems SPARCstation 2 with an 80MHz Weitek CPU.

7. CONCLUSIONS

A set of algorithms have been presented that enable a computer to automatically read a paragraph of handprinted text. A new technique was introduced that reconstructs lines of text from the connected components within a handprinted paragraph. This method is very robust as it avoids tracking components that are statistical outliers, and the method is able to detect and split characters from multiple lines that touch each other vertically. A new character segmentor was also presented that adapts automatically to writing style. The method composes characters from multiple components and it separates touching characters from single components. The method was shown to contain certain limitations in its ability to account for the widely varying character heights within the unconstrained paragraphs. However, its use did significantly improve recognition. The performance of the segmentor was compared between 1-line responses and the paragraphs, and it was concluded that a person's writing has a tendency to become more varied and irregular as spatial constraints are relaxed. Recognition results were computed using an optimized PNN in conjunction with a dictionary-based correction scheme. On a testing set of 500 handprinted paragraphs containing the Preamble to the U.S. Constitution, the new methods improved word recognition accuracy nearly 4% (from 60% to 64%). The work in this paper represents significant steps toward a general purpose unconstrained handprint recognizer. Potential applications include forms processing and the recognition of handprinted facsimile. The Preamble paragraphs are representative of spatially unconstrained handwriting, thus the new method of line reconstruction is an elegant solution having generally broad applicability. On the other hand, the Preamble paragraphs are in terms of lexical content *very* constrained. Few applications have such constraints; therefore, the results reported in this paper are (in general) somewhat optimistic. Future work must include a more general and efficient solution to spell correction.

REFERENCES

[1] M. D. Garris, J. L. Blue, G. T. Candela, D. L. Dimmick, J. Geist, P. J. Grother, S. A. Janet, and C. L. Wilson, "NIST form-based handprint recognition system," Technical Report NISTIR 5469 and CD-ROM, National Institute of Standards and Technology, July 1994.

[2] P. J. Grother, "Handprinted forms and character database, *NIST Special Database 19*," Technical Report and CD-ROM, National Institute of Standards and Technology, March 1995.

[3] M. D. Garris, "Unconstrained handprint recognition using a limited lexicon," Proceedings of *Document Recognition*, SPIE, vol. 2181, pp. 36-46, San Jose, February 1994.

[4] D. J. Ittner, "Automatic inference of textline orientation," Proceedings of *Second Annual Symposium on Document Analysis and Information Retrieval*, pp. 123-133, Las Vegas, April 1993.

[5] G. T. Toussaint, "Computational geometry for document analysis," Proceedings of *Third Annual Symposium on Document Analysis and Information Retrieval*, pp. 23-42, Las Vegas, April 1994.

[6] S. Fortune, "Voronoi diagrams and Delaunay triangulations," in *Computing in Euclidean Geometry*, Editors D. Z. Du and F. K. Hwang, World Scientific Publishing Co., 1992.

[7] T. Asano, M. Edahiro, H. Imai, M. Iri, and K. Murota, "Practical use of bucketing techniques in computational geometry," in *Computational Geometry*, Ed. G. T. Toussaint, North-Holland, 1985, pp. 153-195.

[8] S. Fortune, "Sweepline algorithm for Voronoi diagrams," *Algorithmica*, vol. 2, 1987, pp. 153-174.

[9] M. D. Garris, "Component-based handprint segmentation using adaptive writing style model," Technical Report NISTIR 5843, National Institute of Standards and Technology, June 1996.

[10] P. J. Grother, "Karhunen Loève feature extraction for neural handwritten character recognition," Proceedings of *Applications of Artificial Neural Networks III*, SPIE, vol. 1709, pp. 155-166, Orlando, April 1992.

[11] D. F. Specht, "Probabilistic neural networks," *Neural Networks*, Vol. 3(1), pp 109-119, 1990.

[12] J. L. Bentley, "Multidimensional binary search trees used for associative searching," Communications of the ACM, Vol. 18, pp. 509-517, 1975.

[13] H. G. Zwakenberg, "Inexact alphanumeric comparison," *The C Users Journal*, pages 127-131, May 1991.

[14] M. D. Garris and S. A. Janet, "Scoring Package release 1.0," *NIST Special Software 1* and CD-ROM, vol. SP, National Institute of Standards and Technology, October 1992.