NISTIR 5310

Unconstrained Handprint Recognition Using a Limited Lexicon

Michael D. Garris

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Advanced Systems Division
Computer Systems Laboratory
Gaithersburg, MD 20899

December 1993

U.S. DEPARTMENT OF COMMERCE
Ronald H. Brown, Secretary

TECHNOLOGY ADMINISTRATION
Mary L. Good, Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
Arati Prabhakar, Director

Unconstrained Handprint Recognition Using a Limited Lexicon

Michael D. Garris

National Institute of Standards and Technology,
Gaithersburg, Maryland 20899

ABSTRACT

A word recognition system has been developed at NIST to read free-formatted text paragraphs containing handprinted characters. The system has been developed and tested using samples of handprint from *NIST Special Database 1*. This database of binary images contains 2,100 different writers' printings of the Preamble to the U. S. Constitution. Each writer was asked to print these sentences in an empty 70mm by 175mm box. The Constitution box contains no guidelines for the placement and spacing of the handprinted text, nor are there guidelines to instruct the writer where to stop printing one line and to begin the next. While the layout of the handprint in these paragraphs is unconstrained, a limited-size lexicon may be applied to reduce the complexity of the recognition application. The Preamble contains 38 unique words comprised of 35 unique upper-case and lower-case letters, ignoring punctuation marks.

The system is divided into four general components. 1) The Constitution box is located within a full-page image, and the handprint within the box is isolated. 2) The subimage containing the handprinted text is segmented using connected component labeling, and the resulting blobs are sorted into correct reading order. 3) The segmented blobs are classified using feature-based neural network recognition. 4) Words are parsed from the line-ordered classifications using the lexicon to locate word boundaries and to correct classification and segmentation errors. These components have been combined into an end-to-end hybrid system that executes across a UNIX file server and a massively parallel SIMD computer. The recognition system achieves a word error rate of 49% across all 2,100 printings of the Preamble (109,096 words). This performance is achieved with a neural network character classifier that has a substitution error rate of 14% on its 22,823 training patterns. This demonstrates the power of using a limited-size lexicon to parse words from a less than optimal character classifier. This paper discusses the word recognition system in detail.

1. INTRODUCTION

Automated recognition of handprint has been the topic of much research.[1-5] Digit recognition approaches usually incorporate some combination of segmentation and classification, but very little contextual information exists to decrease classification errors. Typically, only a limited amount of context exists for numeric fields on forms. An example are fields that must sum to a total, and the total is contained in another field. This lack of context causes digit recognition systems to require highly accurate segmentation and classification components. On the other hand, alphabetic fields on forms are contextually rich with information. For example, address models may be used to parse street numbers from street names, and lexicons may be used to resolve categorical information in fields such as names of people or occupations. This paper demonstrates how this second type of contextual information (limited-size lexicons) relaxes the dependence of a recognition system on extremely accurate segmentation and classification, providing robustness and counteracting the effects of increasing the dimensionality of a 10-class digit classifier to a much larger 29-class alphabetic classifier.

A word recognition system has been developed at NIST to read the Constitution box on the Hand Writing Sample Forms (HSFs) contained in *NIST Special Database 1* (SD1).[6,7] An example of this box is shown in Figure 1. SD1 contains 2,100 different writers' printings of the Preamble to the U. S. Constitution. The images in SD1 are binary and digitized at 12 pixels per millimeters. The box contains no guidelines for the placement and spacing of the handprinted text, nor are there guidelines to instruct the writer where to break the ends of lines and begin new ones. Each writer was simply asked to print the Preamble in the empty 70mm by 175mm box at the bottom of the provided form. The resulting handprinted text is positionally unconstrained.

The functional components of the word recognition system are shown in Figure 2. The system isolates the Constitution box on the form, extracts the handprinted text within the box, and then proceeds to read the handprinted information. Field isolation and handprint extraction are conducted using spatial histograms as discussed in Reference 8. To read the handprint, the extracted text is segmented using connected component labeling, and the resulting blobs are sorted into lines. As will be seen, this conceptually easy task requires a fairly complex computer solution. Once sorted into lines, the segmented blob images are classi-

fied using a feature-based neural network classifier.[9] The accuracy of this classifier is less than optimal for reasons discussed later. To counteract the effects of segmentation and classification errors, contextual information is used to improve system accuracy. A lexicon is constructed from the list of 38 unique words contained in the Preamble. This lexicon is used to parse the lines of classified blob images into words. In this way, contextual information is used to detect word boundaries and, more importantly, correct previous errors made in processing the text.

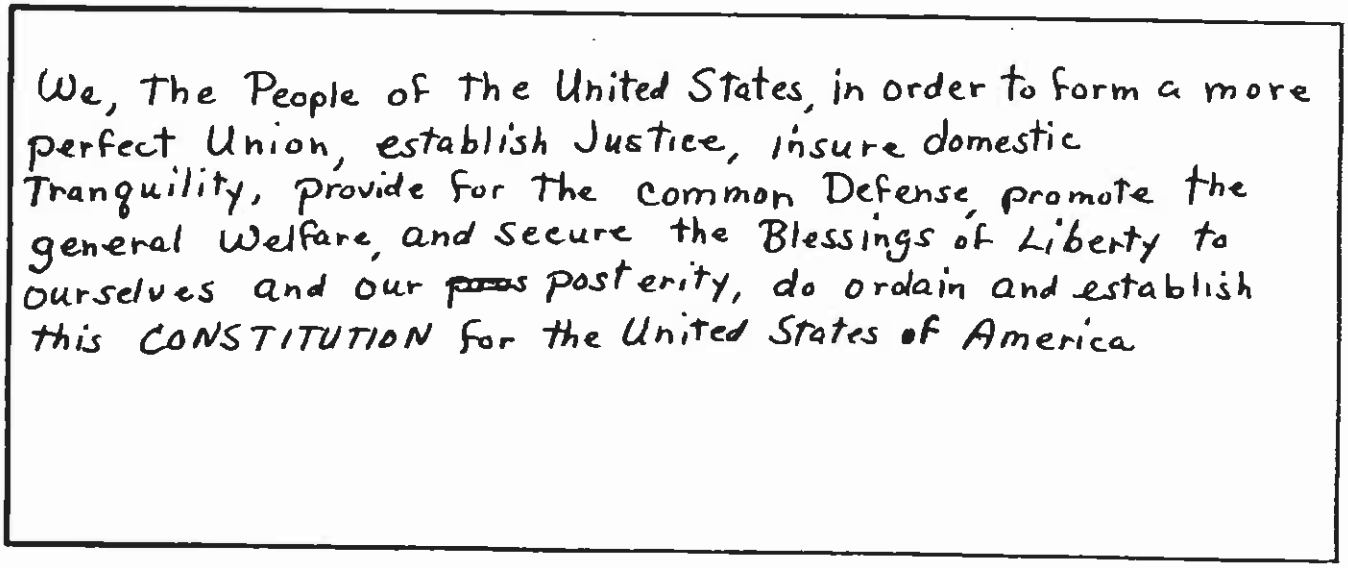


Figure 1. Constitution box from a Hand Writing Sample Form.

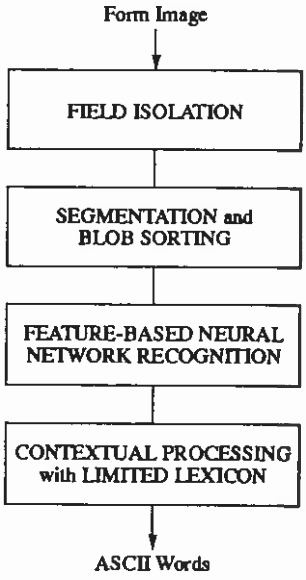


Figure 2. Functional components of an unconstrained handprint recognition system.

This model recognition system is implemented across two integrated computers.* Data storage, central processing control, and contextual processing are 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 a Cambridge Parallel Computing 510C Distributed Array Processor (DAP)[10]. The parallel machine is a Single Instruction Multiple Data (SIMD) architecture and

* The 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.

consists of two separate 32 X 32 grids of tightly coupled processors. One grid contains 1-bit processing elements and 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 low-level isolation, segmentation, and classification tasks.

Section 2 describes segmentation and the process of sorting blobs into lines, Section 3 discusses feature-based neural network recognition, Section 4 presents contextual processing using a limited-size lexicon, and Section 5 contains results using the system across SD1.

2. SEGMENTATION AND BLOB SORTING

As mentioned in the introduction, the recognition system uses connected component labeling to segment the handprinted text into blob images. The advantage of this technique is that it is applied globally across the image without local heuristics, making it fast on the parallel computer. A disadvantage is that blobs (groups of connected pixels) do not always represent single and complete characters. Connected component labeling identifies each and every disjoint piece of image information in the field. Errors are introduced because a blob may represent an individual disjoint stroke of a character, a blob may represent noise in the field, and a blob may represent multiple characters touching each other.

The location of each blob is identified by computing the geometric center of the smallest rectangle bounding the blob. The result is a 2-dimensional grid of blob centers that can be used to represent the line trajectories of the handprinted text. Initially it was anticipated that a simple sort of the x and y center coordinates would be sufficient to organize the blobs into reading order (left-to-right and top-to-bottom). Unfortunately, it was found that the sample of handprinted text used in this study fluctuated significantly within lines as well as across lines, and this fluctuation was exaggerated by the use of punctuation marks, causing techniques that use global line statistics to fail.

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. A point-to-point search is conducted based on a local search space defined by 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. The interior of the function is used as a locally constrained search space. Through empirical study a technique for controlling the shape of S was developed. 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.

A linear control function, $b=L(a)$, is used to modify the shape of the bubble, b, as the length of the bubble, a, is increased. This function is defined by the slope of the line connecting two empirically derived points. One point used to define the control line, $(a1, b1)$, is calculated:

$$a1 = h \times 0.375 \qquad b1 = 0.1 \tag{2}$$

where h is the average blob height for the writer. If for example the writer's average blob height is 32 pixels, this point on the linear control function defines a circular bubble with a radius of 12 pixels (1mm). The second point used to define the control line, $(a2, b2)$, is calculated:

$$a2 = h \times 4.7 \qquad b2 = 2.0 \tag{3}$$

If the writer's average blob height is 32 pixels, this second point defines a tear-shaped bubble with a horizontal length of 150 pixels. If a writer's handprint is small, the bubbles used in the search are adapted to be smaller, and if a writer's handprint is large,

the bubbles used in the search are adapted to be larger. In addition, the bubble defining the search space continuously changes from circular to tear-drop in shape as the extent of the search increases.

Using the linear control function L, 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. If a search is to be conducted relative to the right of a blob's center, then only the portion of the function with $x>0$ is used. If a search is to be conducted relative to the left of a blob's center then the portion of the function with $x<0$ is used. The search is conducted by initializing a to a starting length and then testing to see if any other blob centers are located within the boundary of the bubble. If points are found, then the nearest blob is selected. Otherwise, a is incremented and the bubble is enlarged and lengthened and a test for blobs in the new bubble is conducted. This continues until a center point of a neighboring blob is found, or a exceeds some threshold. In Figure 3, successive bubbles are overlaid from a common center point where a is initialized to 12, incremented by 20, and terminated at 300.

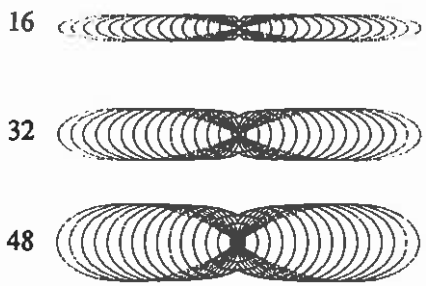


Figure 3. Adaptation of bubbles to the writer's average blob height.

The search begins from the blob center closest to the top-left corner of the field. 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. The threshold used in this system is 300 which is equal to 1 inch (12 pixels per millimeter equals 300 pixels per inch). 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, 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, such as may happen when a comma is found.

Figure 4 illustrates this heuristic as it is used to control the advancement of the search. In order to advance, the new blob center must be within the area defined by the union of two region. The first region is the area bounded by two lines with slope -0.25 and +0.25 projecting from the current blob center. The second region is the area bounded by two horizontal lines with y-intercepts at $-(0.25*h)$ and $(0.25*h)$, centered about the last blob added to the list. The top diagram in Figure 4 show bubbles projected from the current blob center (1). The closet neighboring blob center is (2), however (2) is not within the given criteria represented by the region filled with gray. Therefore, (2) is added to the list with (1), but the current bubble position does not advance to (2). The middle diagram in Figure 4 shows the search continuing with bubbles being projected from (1). The next closest blob center is (3). Notice that the gray region has changed from the top diagram. The triangular slope-based region remains anchored to (1), but the horizontal region, based on the writer's average blob height, is now defined in relationship to (2). Blob center (3) is added to the list with (1) and (2), and because (3) is within the new gray region, the current blob center advances to (3) as shown in the bottom diagram in Figure 4. Note that once a blob center is added to the list, it is not considered again in the search process.

It was observed during the development of this approach that the heuristic described above, when tuned to handle isolated cases, did not yield proper results in other cases. 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 a blob not yet included in any lists and closest to the top-left of 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 to a list.

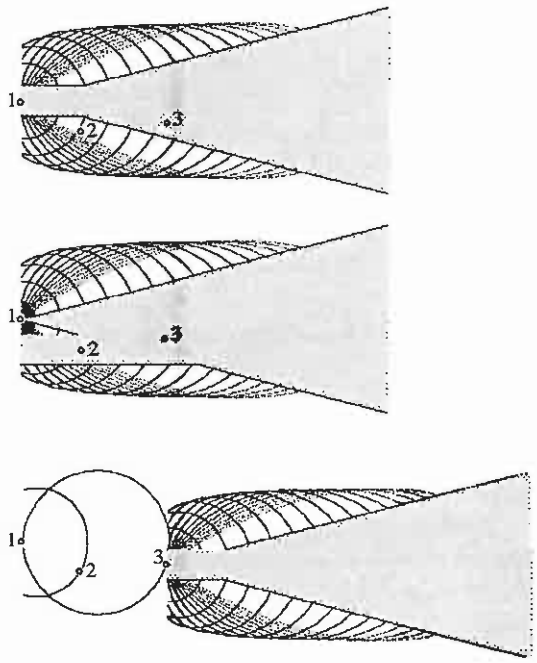


Figure 4. Heuristic used to control the advancement of the search.

The criterion for preempting the search and beginning a new list is illustrated in Figure 5. In this illustration, the search is currently being conducted from blob center (2), and (3) has been located as the next nearest neighbor. In the previous step, (2) was found by searching from (1), and both (1) and (2) have been added to the current list. The distance, $d1$, is the vertical distance between (1) and (2), and the distance, $d2$, is the vertical distance between (2) and (3). The two parallel horizontal lines in the diagram represent the area bounded by $-h$ and $+h$ centered about the previous blob center (1). The sum of $(d1+d2)$, the vertical distance between (1) and (3), exceeds the limit, l, representing the region bounded by the horizontal lines; therefore 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.

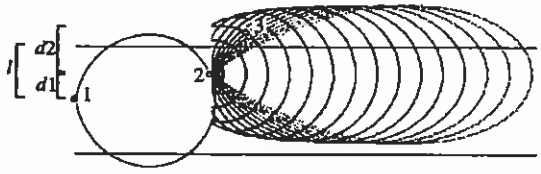


Figure 5. Heuristic used to preempt the search.

It is surprising how well this preemptive heuristic works. At times, more frequently with some writers than with others, the local writing fluctuations become excessive and the search is restarted. Often the restart resumes on the next line and the point-to-point search is successful in tracking the next line. The search is top-down by nature, so that the blobs in the line above the area of excessive fluctuation are likely to be assigned to a previous list. Eventually the left-most blob involving the fluctuation is the closest remaining blob to the top-left of the field, and the point-to-point search resumes from that blob. All neighboring blobs from the line above and the line below have been previously assigned leaving only the blobs comprising the fluctuation exposed. This greatly reduces the system's *guess-work* and thereby reduces system errors. Figure 6 shows the results of segmenting the image in Figure 1 and using the bubble technique to sort the blobs into lines. A bubble is traced from each point where a neighboring blob was found, and each bubble reflects the actual size and the shape of the search space used to locate the neighbor.

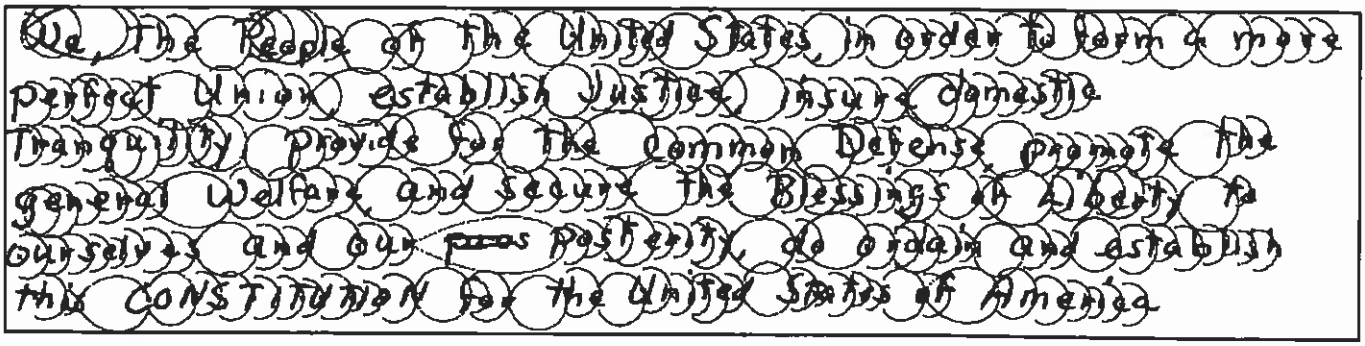


Figure 6. Traces of the bubbles used to sort the blobs into lists.

The point-to-point search produces multiple lists of blobs. 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, upon completion, only lists containing complete text lines remain. Two heuristics are used for merging the blob lists; they are illustrated in Figure 7 and Figure 8. The lists are sorted in descending order according to the number of blobs in each list. The longest list is first compared against all other lists, applying the first heuristic and then the second to each comparison. If two lists meet the merging criteria, they are merged and the looping process is restarted by resorting the lists. Otherwise, the next longest list is compared to the remaining shorter lists, and so on until all the lists are looped through and no merging takes place. When two lists are merged, their blob centers are appended into one larger list and then sorted on their x-coordinates. In Figure 7, two lists are merged if the end point of the shorter list is within a vertical distance of -(0.75**h*) and +(0.75**h*) of the larger lists's corresponding end point.

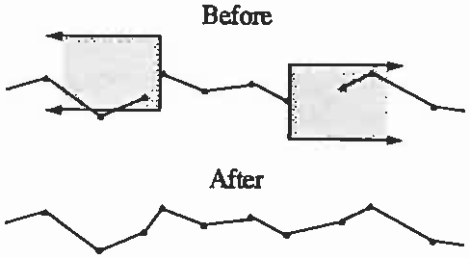


Figure 7. Heuristic for merging blob lists based on end point positions.

The second merge heuristic is illustrated in Figure 8. In this case, the blob centers comprising the longer list are fitted using linear least squares to produce a slope and y-intercept. A perpendicular distance is computed between each blob center in the shorter list and the line fitted to the longer list, and the distances less than (0.5**h*) are counted. This area along the fitted line is represented by the gray region in the diagram. The two lists are merged if the count is greater than (0.1**n*), where *n* is the number of blobs in the longer list. This facilitates merging of lists that lie along the same line trajectory, but whose end points are somewhat erratic.

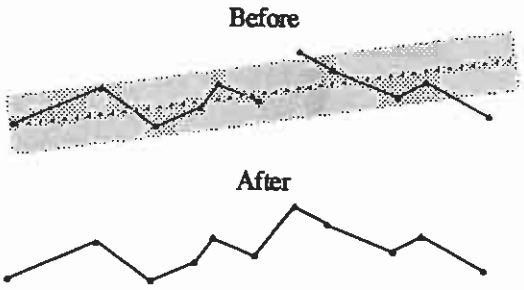


Figure 8. Heuristic for merging blob lists based on line trajectories.

3. FEATURE-BASED NEURAL NETWORK CLASSIFICATION

The classification of blob images is discussed in this section. After segmentation and prior to classification, the blob images are spatially normalized to fit within a 32 X 32 image, sheared to remove slant from the handprint, and filtered into ranked principal components using the discrete Karhunen Loeve (KL) transform.[11] The recognition system uses these KL features as input to a Multi-Layer Perceptron (MLP)[12]. The MLP classifies by generating feedforward activations across a network containing an input layer, one hidden layer, and an output layer. In this study, a network with 64 input neurodes, 64 hidden neurodes, and 29 output neurodes is used. The network is trained using Scaled Conjugate Gradient (SCG).[13] Note that the training of this network is done once, off-line from the running of the recognition system.

The KL transform is a statistical method that expands characters in terms of eigenvectors whose eigenvalues are variances. The eigenvectors are the principal components of the covariance matrix formed from a sample of characters. Those eigenvectors with the highest eigenvalues are more relevant descriptors of the character images. Givens and Householder reductions are used to tridiagonalize the covariance matrix, and the eigenvectors are computed using the QR algorithm.[14] The eigenvectors form a minimal orthogonal basis set of which any character is a linear combination. A feature vector of coefficient values is computed by projecting a character image onto the set of eigenvectors. This feature vector is then truncated and used in place of the original character image as input to a neural network, reducing the input dimensionality of the MLP. This dimensional reduction is important for the generalization capabilities of the network.[15] In this study, 64 KL basis functions were used in place of the original 1,024 pixels contained in the 32 X 32 normalized blob images. The network was trained on feature vectors computed from 22,823 character images from NIST Special Database 3 (SD3).[16]

As stated in the introduction, the MLP used in this study demonstrates less than desired accuracies. It has been shown that for applications involving relatively small training sets, Probabilistic Neural Networks (PNN) and Radial Basis Functions (RBF) are more efficient and outperform MLPs.[17,18] First, the classification of alphabetic characters is of higher dimension than the classification of digits. Digit recognition was emphasized at the First Census Optical Character Recognition Systems Conference sponsored by the Bureau of the Census and hosted by NIST.[19] The classification of upper cases letters and lower case letters was also included in the Conference, but the studies were conducted on much smaller testing sets. It was demonstrated that error rates as low as 3% without rejecting any classifications can be achieved on large samples of digits. However, error rates only as low as 5% to 6% were demonstrated on upper-case letters. The difference in performance between digit recognition and upper-case recognition can be attributed in part to the larger class size, 26 versus 10, for upper-case letters. Error rates of 10% to 15% were demonstrated for lower case letters in the Conference. Adding lower-case recognition to a classifier greatly increases the level of ambiguity among the classes. For example, it is very difficult to distinguish one writer's "S" from another writer's "s". This ambiguity is compounded when scale normalization is used by the recognition system.

Considering upper and lower-case and ignoring punctuation, the Preamble contains 35 unique characters. These 35 characters are condensed to 29 classes in order to alleviate some of the ambiguities described above. Figure 9 lists the 29 classes and shows which characters were collapsed into a single class. Rather than training the MLP to distinguish the 35 unique characters in the Preamble, the network was trained to classify images into these 29 classes. Notice that letters such as "Z" do not occur in the Preamble and therefore are not included in the classifier.

Class	Members	Class	Members	Class	Members
1	A	11	O o	21	d
2	B	12	P p	22	e
3	C c	13	S s	23	g
4	D	14	T	24	h
5	F f	15	U u	25	l
6	I	16	V v	26	n
7	J	17	W w	27	q
8	L	18	Y y	28	r
9	M m	19	a	29	t
10	N	20	b		

Figure 9. The 29 classes used to train the MLP.

The low performance of the MLP used in this study is also largely due to an undersized training set. The higher dimensionality of this 29-class network demands a larger and more robust training set than a 10-class digit network. NIST has demonstrated digit classifiers that achieve less than 5% error rates with no rejection when trained on as little as 7,400 exemplars[8,9]. The 29-class network use in this study on the other hand was trained on 22,823 exemplars and upon completion of its training only achieved a 14% error rate on the exemplars in the training set. In fact, one of the principle motivations for developing this system was to build a much larger training set for future experiments by automatically boot-strapping a database labeling system as described in Reference 20.

4. CONTEXTUAL PROCESSING USING A LEXICON

The box containing the handprinted Preamble is isolated on the HSF form, and the handprint is extracted. The field image containing the handprinted text is segmented and the resulting blob images are sorted into correct reading order. Each blob image is spatially normalized, sheared, and converted into a 64-coefficient KL feature vector. The feature vector for each blob is classified into any of 29 classes using the MLP. These steps were applied to the image in Figure 1 producing the text shown in Figure 10. Notice the large number of errors contained in classifier output, and also notice there are no inter-word spaces recognized at this point in the process. This section presents a technique developed to correct these classification errors and to detect word boundaries within text lines like the ones shown in Figure 10.

```
Line 1:   ILaUThePeOPleOFtheUhlteaSTCteSlnOrdertOFOrMaMOre
Line 2:   PtrFeCtUhOhlOtablIShJUStICellnSUredOMeStlC
Line 3:   TranqUIlhtYJPrOIdeFUrTheCOMaOnDeFMSePrOlnUrerhe
Line 4:   genIralWWMrCandSeeUrttheBIISSlngSOFLIbehtYtO
Line 5:   OUrSeIVOSandOUrPOStlrItYdOOrdalnCndNtabIISh
Line 6:   MIJUNSTITUTIONFOrgeUnltedSrareSMFAMerdCa
```

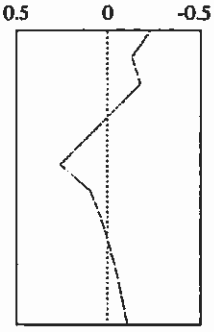
Figure 10. Classifier output prior to contextual processing.

The Preamble is comprised of 38 unique words, and these words are used to construct the lexicon shown in Figure 11. The technique uses the lexicon 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 11. Lexicon constructed from the text contained in the Preamble to the U. S. Constitution.

The technique is illustrated by the example shown in Figure 12. In this example, a portion of the first line of text in Figure 10, "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 generation of these signals will be discussed later. 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 12 characters, which is the length of the longest word in the lexicon, "CONSTITUTION". Notice the text line is converted to all upper-case letters prior to forming the hypothesized words. 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 to match the hypothesized word to the lexicon match.[21] 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 12. 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:

$$e = \frac{n}{l-g} \tag{4}$$

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 minimizes 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 used to compute the signal is t. This is a translation term based on the linear function, T, that biases longer hypothesized words over shorter ones. In this way, matches to the word "DOMESTIC" are favored over matches to the word "DO", and "INSURE" is favored over the word "IN". The linear translation function used in this study is defined by the empirically derived points (2, 0.5) and (12, 0.4); such that t=0.5 for hypothesized words of length 2, and t=0.4 for hypothesized words of length 12. The translation term is determined by locating the point on the line at the position corresponding to the length of the hypothesized word's dictionary match. If p is the length of the dictionary match, then $t=T(p)$. Signal value, s, is then computed:

$$s = (1.0 - e) - t \tag{5}$$

The signals listed in the first column of Figure 12 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, the lexicon match for that word is pushed onto a stack and the alignment is used to synchronize the processing. As can be seen in Figure 12, characters that match between the hypothesized word and the lexicon match are represented by 0's. The alignment elements between the left-most 0 and the right-most 0 comprise an *alignment span*, and it corresponds to the characters in the lexicon match. The ends of this alignment span demarcate the boundaries of the word within the original line of text. Processing the signals in this fashion is done recursively. If a portion of the fan-out remains to the left of the selected word's alignment span, then the remaining piece of fan-out may contain another word. Remember the maximum hypothesized word is 12 characters which is long enough to hold 3 or 4 small words from the lexicon simultaneously. The remaining left portion is processed recursively, recalculating new signal values and searching for words within that piece. As lexicon matches are selected, they are pushed onto the stack.The recursion continues until all of the fan-out to the left of the top-level selected word has been exhaustively processed. The selected lexicon matches are then popped off the stack in correct reading order, and a new fan-out is rebuilt beginning with the first character to right of the top-level selected word's

alignment span. For example in Figure 12, the hypothesized word, "STCTES" is selected with a maximum signal of 0.254. The next fan-out will begin with "L", starting from the position in the text line, "lnOrdertOFOr". 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.

5. RESULTS

The contextual processing described in Section 4 was applied to the text lines listed in Figure 10. The result is shown in Figure 13. The selected lexicon matches have been directly overlaid in reverse video machine font onto the original field image. As can be seen, the recognition system does a reasonably good job in reading the handprinted text. To appreciate the performance of the contextual processing, the results in Figure 13 should be compared with the classifier output in Figure 10. A few of the shorter words in the lexicon are deleted from the text, such as the word "IN" on the first line, and "A" is substituted for "WE". A few mistakes are made on longer words as well. The classifier output for the word "United" is "Uhltea", which is poor enough that the contextual processing matches "TE" to "THE", and the "a" substituted for "d" results in a match of "A" from the lexicon. This results in "A" being displayed on top of "THE" in the figure.

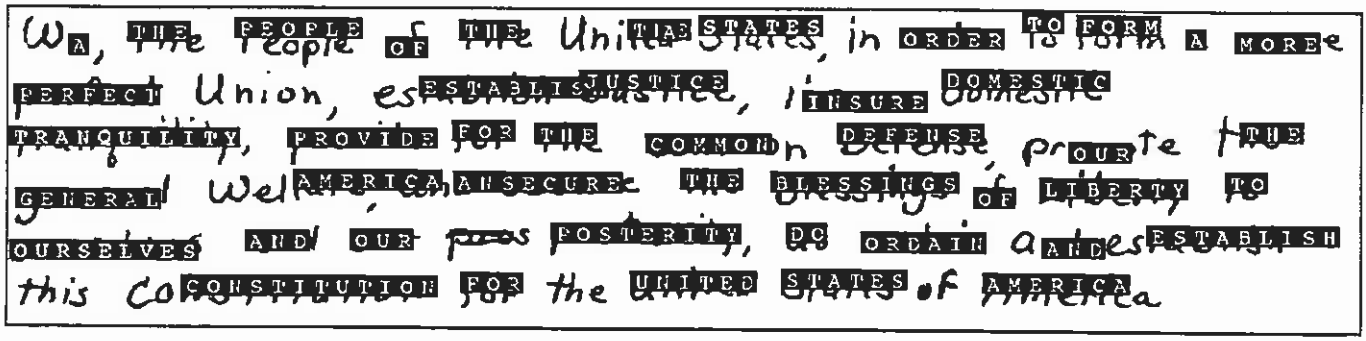


Figure 13. Recognition system results after contextual processing.

The functional components discussed in the previous sections were integrated into an end-to-end word recognition system. A form image is passed into the system, and ASCII words are received out. The recognition system was used to read all 2,100 Constitution boxes in SD1. A system hypothesis file was generated for each box, and these files were scored using the NIST Scoring Package[22,23,25] according to the methods outlined in Reference 24. Word-level scores were generated by tokenizing each word in the lexicon and then substituting the reference and hypothesized texts with these tokens. Scoring alignments were than conducted on the tokens and statistics were accumulated. The system achieved a 49% word error rate using equation CHAR8 from Reference 24 across the sample of 109,096 potential words (669,262 characters). Running across both the serial Sun 4/470 and parallel DAP 510c, the system took on average 106 seconds to process each Preamble with the majority of the time (77%) spent on contextual processing which was performed entirely on the Sun.

6. CONCLUSIONS

A word recognition system designed to read free-formatted text paragraphs containing handprinted characters was presented. Components of the system were described, including the segmenting and sorting of blobs into correct reading order, classifying the blobs using a feature-based neural network, and correcting segmentation and classification errors using contextual processing base on a lexicon. A technique using a search space based on a function taken from antenna modeling was presented for tracking the local fluctuations in handprinted text, and a recursive technique based on matching character classifier output to a lexicon using the Levenstein Distance was presented. The recognition system was used to process the 2,100 different writers' handprinted versions of the Preamble to the U. S. Constitution contained in *NIST Special Database 1*. The system was scored using the NIST Scoring Package and achieved a word error rate of 49% across the sample of 109,096 potential words (669,262 characters). This demonstrates the power of using a limited-size lexicon to parse words from the output of a less than optimal character classifier that has an error rate of 14% on its own training set. The system runs across a Sun 4/470 serial computer and a DAP 510c parallel SIMD computer. On average, the recognition system took 106 seconds to process each printing of the Preamble, and the majority of the time (77%) was spent doing contextual processing on the serial machine.

7. REFERENCES

1. C. L. Wilson, R. A. Wilkinson, and M. D. Garris, "Self-Organizing Neural Network Character Recognition on a Massively Parallel Computer," *International Joint Conference on Neural Networks*, Vol. II, pp. 325-329, San Diego, 1988.

2. K. Fukushima, T. Imagawa, and E. Ashida, "Character Recognition with Selective Attention," International Joint Conference on Neural Networks, Vol. I, pp. 593-598, Seattle, 1991.

3. G. E. Hinton, and C. K. I. Williams, "Adaptive Elastic Models for Hand-Printed Character Recognition," *Advances in Neural Information Processing Systems*, R. Lippmann, Vol. IV, pp. 512-519, Denver, 1991.

4. L. D. Jackel, H. P. Graf, W. Hubbard, J. S. Densker, D. Henderson, and Isabelle Guyon, "An Application of Neural Net Chips: Handwritten Digit Recognition," *International Joint Conference on Neural Networks*, Vol. II, pp. 107-115, San Diego, 1988.

5. G. L. Martin and J. A. Pittman, "Recognizing Hand-Printed Letters and Digits," *Advances in Neural Information Processing Systems*, D. S. Touretzky, Vol. 2, 405-414, Morgan Kaufmann, Denver, 1989.

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

7. M. D. Garris. Design and Collection of a Handwriting Sample Image Database, *Social Science Computing Journal*, Vol. 10, pp. 196-214, Duke University Press, 1992.

8. M. D. Garris, 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*, Vol. 1661, pp. 269-280, San Jose California, SPIE, February 1992.

9. P. J. Grother and G. T. Candela, "Comparison of Handprinted Digit Classifiers," Technical Report NISTIR 5209, National Institute of Standards and Technology, June 1993.

10. P. M. Flanders, R. L. Hellier, H. D. Jenkins, C. J. Pavelin, and S. Van Den Berghe, "Efficient High-Level Programming on the AMT DAP," IEEE Proceedings: Special Issue on Massively Parallel Computers, Vol. 79(4), pp. 524-536, April 1991.

11. P. J. Grother, "Karhunen Loeve Feature Extraction for Neural Handwritten Character Recognition," In *Proceedings: Applications of Artificial Neural Networks III*, Orlando, SPIE, April 1992.

12. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart, J. L. McClelland, et al., Vol. 1: Foundations, pp. 318-362, MIT Press, Cambridge, 1986.

13. M. F. Moller, "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning," Technical Report PB-339, Aarhus University, 1990.

14. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes, The Art of Scientific Computing (FORTRAN Version)*. pp. 349-363, Cambridge University Press, Cambridge, 1989.

15. I. Guyon, V. N. Vapnik, B. E. Boser, L. Y. Bottou, and S. A. Solla, "Structural Risk Minimization for Character Recognition," *Advances in Neural Information Processing Systems*, R. Lippmann, Vol. IV, pp. 471-479, Denver, 1991.

16. M. D. Garris and R. A. Wilkinson, "Handwritten Segmented Characters Database," Technical Report Special Database 3, **HWSC**, National Institute of Standards and Technology, February 1992.

17. C. L. Wilson, "Evaluation of Character Recognition Systems," In *Neural Networks for Signal Processing III*, IEEE, New York, pp. 485-496, September 1993.

18. J. L. Blue, B. T. Candela, P. J. Grother, R. Chellappa, and C. L. Wilson, "Evaluation of Pattern Classifiers for Fingerprint and OCR Applications," *Pattern Recognition*, to be published.

19. R. A. Wilkinson, et al, "The First Census Optical Character Recognition System Conference," Technical Report NISTIR 4912, National Institute of Standards and Technology, July 1992.

20. R. A. Wilkinson, M. D. Garris, J. Geist, "Machine-Assisted Human Classification of Segmented Characters for OCR Testing and Training," D. P. D'Amato editor, Vol. 1906, pp. 208-217. SPIE, San Jose, 1993.

21. H. G. Zwakenberg, "Inexact Alphanumeric Comparison," *The C Users Journal*, pp. 127-131. May 1991.

22. M. D. Garris and S. A. Janet, "Scoring Package Release 1.0," Technical Report Special Software 1, **SP**, National Institute of Standards and Technology, October 1992.

23. M. D. Garris and S. A. Janet, "NIST Scoring Package User's Guide, Release 1.0," Technical Report NISTIR 4950, National Institute of Standards and Technology, October 1992.

24. M. D. Garris, "Methods for Evaluating the Performance of Systems Intended to Recognize Characters from Image Data Scanned from Forms," Technical Report NISTIR 5129, National Institute of Standards and Technology, February 1993.

25. M. D. Garris, "NIST Scoring Package Cross-Reference for Use with NIST Internal Reports 4950 and 5129," Technical Report NISTIR 5249, National Institute of Standards and Technology, August 1993.