Public domain optical character recognition

Michael D. Garris, James L. Blue, Gerald T. Candela, Darrin L. Dimmick, Jon Geist,
Patrick J. Grother, Stanley A. Janet, and Charles L. Wilson
National Institute of Standards and Technology,
Gaithersburg, Maryland 20899

Published in SPIE Document Recognition II, pp. 2-14, San Jose, February 1995.

ABSTRACT

A public domain document processing system has been developed by the National Institute of Standards and Technology (NIST). The system is a standard reference form-based handprint recognition system for evaluating optical character recognition (OCR), and it is intended to provide a baseline of performance on an open application. The system's source code, training data, performance assessment tools, and type of forms processed are all publicly available. The system recognizes the handprint entered on Handwriting Sample Forms like the ones distributed with *NIST Special Database 1*. From these forms, the system reads handprinted numeric fields, upper and lowercase alphabetic fields, and unconstrained text paragraphs comprised of words from a limited-size dictionary. The modular design of the system makes it useful for component evaluation and comparison, training and testing set validation, and multiple system voting schemes. The system contains a number of significant contributions to OCR technology, including an optimized Probabilistic Neural Network (PNN) classifier that operates a factor of 20 times faster than traditional software implementations of the algorithm. The source code for the recognition system is written in C and is organized into 11 libraries. In all, there are approximately 19,000 lines of code supporting more than 550 subroutines. Source code is provided for form registration, form removal, field isolation, field segmentation, character normalization, feature extraction, character classification, and dictionary-based postprocessing. The recognition system has been successfully compiled and tested on a host of UNIX workstations including computers manufactured by Digital Equipment Corporation, Hewlett Packard, IBM, Silicon Graphics Incorporated, and Sun Microsystems.* This paper gives an overview of the recognition system's software architecture, including descriptions of the various system components along with timing and accuracy statistics.

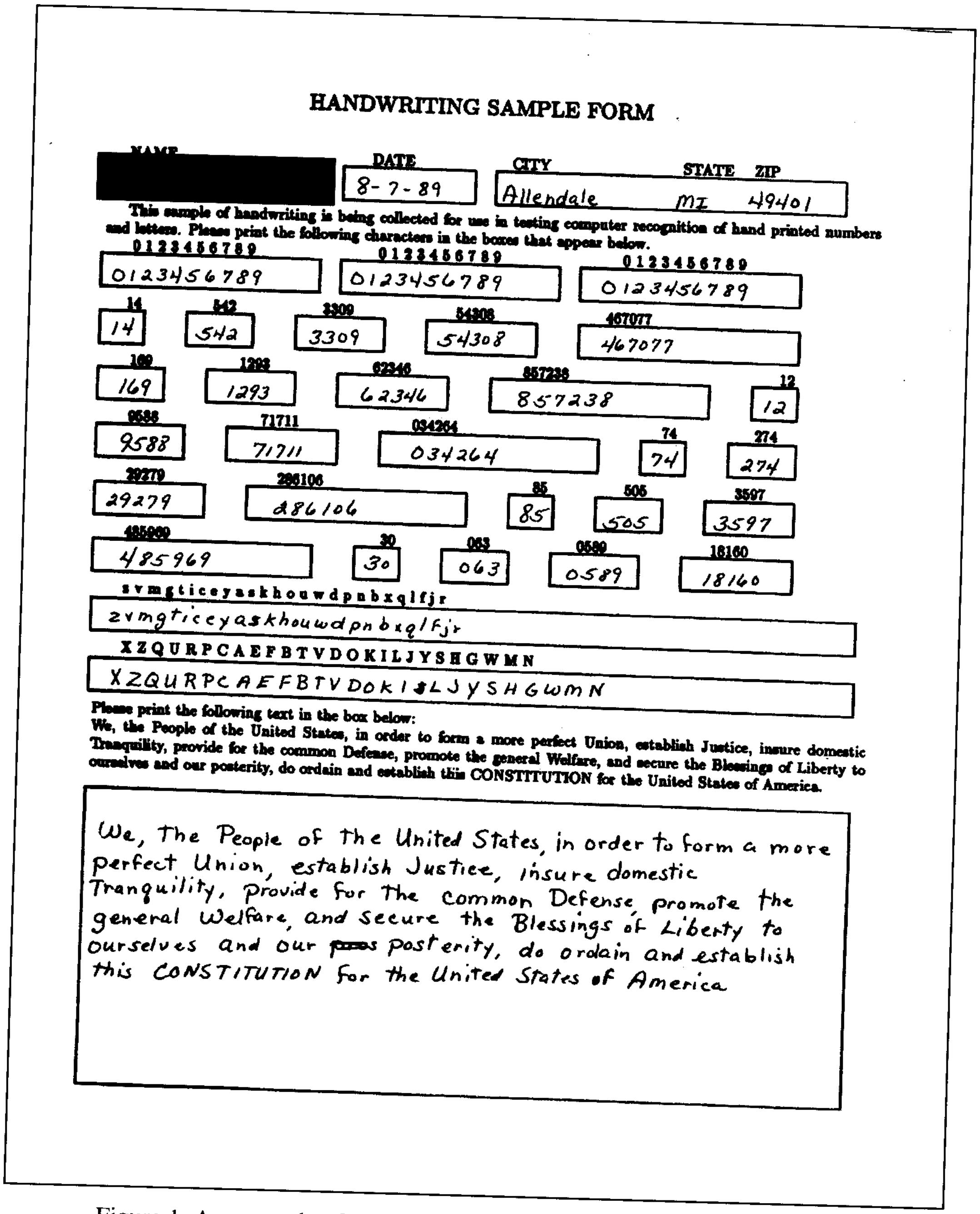
Keywords: CD-ROM, form processing, handprint recognition, neural network, optical character recognition, public domain, software distribution, standard reference system, training data

1. INTRODUCTION

A standard reference form-based handprint recognition system for evaluating optical character recognition (OCR) has been developed.[1] The system has been developed as an *open application*; the system's source code, training data, performance assessment tools, and types of forms processed are all publicly available. The system architecture and software organization is completely documented for those interested in technology integration. The source code for the standard reference system is written in C and is organized into 11 libraries. In all, there are approximately 19,000 lines of code supporting more than 550 subroutines. Source code is provided for form registration, form removal, field isolation, field segmentation, character normalization, feature extraction, character classification, and dictionary-based postprocessing. Any portion of the system may be used without restriction in commercial products.

Due to its modular design, a component of the system may be easily replaced by an alternative algorithm. The same set of input data can be run through the augmented system, and performances between the standard reference system and the augmented system can be compared. The system can be retrained and tested in a controlled way so that the impact of different training set profiles can be compared, and a training set that provides maximum robustness can be determined. Developers may find that the techniques used in the standard reference system provide complimentary results to their own systems. If this is the case, then combining the recognition results from the two systems, or allowing the systems to vote may improve overall recognition performance as demonstrated in Reference 2.

* 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.



HANDWRITING SAMPLE FORM

NAME [redacted] DATE 8-7-89 CITY Allendale STATE MI ZIP 49401

This sample of handwriting is being collected for use in testing computer recognition of hand printed numbers and letters. Please print the following characters in the boxes that appear below.

0123456789	0123456789	0123456789
0123456789	0123456789	0123456789

14	542	3309	54308	467077
14	542	3309	54308	467077
169	1293	62346	857238	12
169	1293	62346	857238	12
9588	71711	034264	74	274
9588	71711	034264	74	274
29279	286106	85	505	3597
29279	286106	85	505	3597
485969	30	063	0589	18160
485969	30	063	0589	18160

s v m g t i c e y a s k h o u w d p n b x q l f j r

zvmgticeyaskhouwdpnbxqlfjr

X Z Q U R P C A E F B T V D O K I L J Y S H G W M N

XZQURPCAEFBTVDOKI~~I~~LJYSHGWMN

Please print the following text in the box below:
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.

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 1. An example of a completed form from *NIST Special Database 1.*

The standard reference system processes the Handwriting Sample Forms (HSF) distributed with *NIST Special Database 1*[3] (SD1). *NIST Special Database 1* contains 2,100 full page images of handwriting samples printed by 2,100 different writers geographically distributed across the United States with a sampling roughly proportional to population density. An example of a filled in HSF form is shown in Figure 1. All HSF forms use a single field template specifying the number of entry fields, their size and location. Each entry field on the form is demarcated by a box. The form is comprised of 3 identification boxes, 28 digit boxes of varying length, a randomly ordered lowercase alphabet, a randomly ordered uppercase alphabet, and a handprinted text paragraph containing the Preamble to the U.S. Constitution. The forms were scanned at 12 pixels per millimeter (300 dots per inch - dpi) binary.

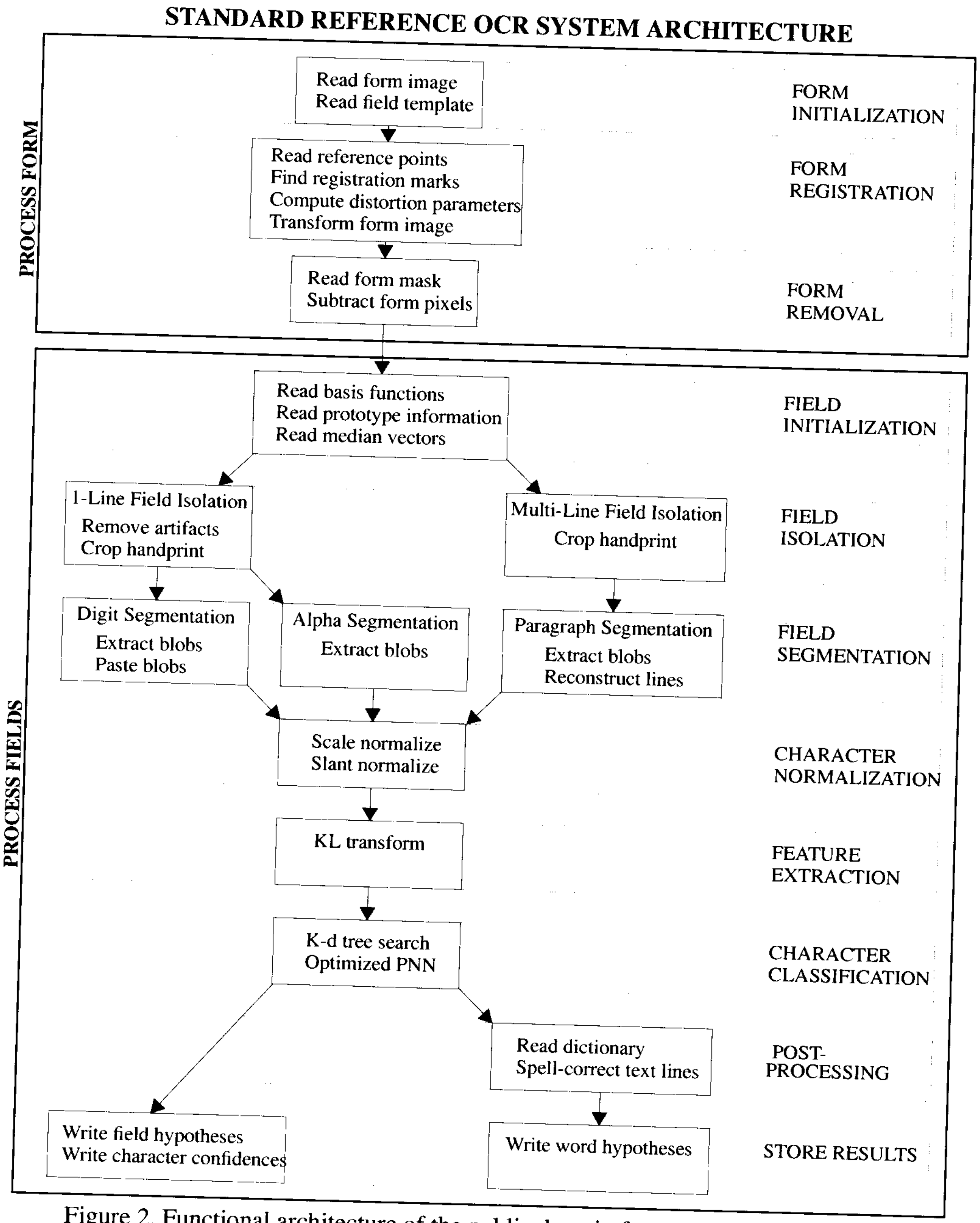


Figure 2. Functional architecture of the public domain form recognition system.

The system has been designed to read all but the first three fields on the HSF form. The functional architecture of the recognition system is illustrated in Figure 2. This diagram represents the processing of handprinted fields that contain all digits, fields that contain all lowercase or all uppercase letters, and fields that contain a text paragraph of mixed upper and lowercase letters. As can be seen from the figure, there is a large overlap in the functional components used across these different types of fields.

This report provides an overview of the standard reference system in terms of its components and functionality. Section 2 briefly describes the functional components of the standard reference system. System performance and timing statistics are presented in Section 3. Section 4 summarizes this paper. A complete description of the system including software organization, file formats, source code compilation, system invocation, and functional components is found in Reference 1.

2. SYSTEM COMPONENTS

This section documents the overall functionality of the standard reference form recognition system. The system components shown in Figure 2 are organized into two functional groups (process form and process field). The first is responsible for processing the input form image by dividing the image into separate fields. The second is responsible for reading the field values handprinted on the form and storing the system results.

2.1 PROCESS FORM

2.1.1 Form initialization

Initializing the system to process a form image requires reading the completed form from an input image file[4] and retrieving a spatial field template. The standard reference system is designed to read only one type of form. If the system was extended to read multiple forms, then a form identification capability would need to be integrated. The appropriate spatial field template would then be used based on the type of form recognized by the system. In all, there are 34 entry fields on an HSF form, each represented as a rectangular region in the field template.

2.1.2 Form registration

Currently, form recognition technology requires stringent form design and printing standards in order to ensure successful OCR.[5] No matter how much care is taken in printing the original forms, many forms processing applications must deal with second and third generation photocopy, or even worse facsimile. These factors generally produce significant distortions in the final image, including rotation, translation, and scale that must be accounted for in order for the recognition system to reliably locate and recognize the data entered in each field on a form. These types of distortions are detected and removed through a process known as form registration.

The standard reference system uses a registration technique based on Linear Least Squares[6] (LSQ) where a set of predefined registration marks on an input form image are matched to marks on an undistorted (registered) blank form image. Global estimates of rotation, translation, and scale are automatically computed and applied so that the input image is transformed to line up as well as possible, in a least squares sense, with the registered blank form. Six points distributed across the form are used by the standard reference system. The recognition system uses spatial histogram projections to locate the position of these registration marks within the input form image. The technique employed in the system carefully reduces the scope of successive histogram projections, alternating between horizontal and vertical projections, until the desired structure is accurately isolated. Using this technique, the standard reference system has been engineered and tested to tolerate up to 5 degrees of rotation in combination with 1.27 cm (0.5 inches) of translation.

Once the registration marks are located on a form, parameters estimating the amount of rotation, translation, and scale are computed. The estimation of distortion parameters is embedded in a technique for detecting form registration failures. The technique transforms the points located on the form using the distortion parameter estimates. If the distance between the transformed points and the known points on a registered form is too great, then the point contributing the greatest distance is removed from the calculation and the distortion parameters are recomputed. The algorithm loops until either there are too few points remaining to accurately compute distortion parameters, or the maximum error distance from all the remaining points falls below a specified threshold. If too few points remain, the form registration is determined to have failed. Otherwise, form registration is determined to be successful, and the last set of distortion parameters computed are used to transform the entire input form image.

Equation (1) represents horizontal translation, rotation, and scale using the unknown quantities Δx, m_{x_x}, and m_{x_y}, whereas Equation (2) represents vertical translation, rotation, and scale using the unknown quantities Δy, m_{y_y}, and m_{y_x}. These equations are written such that hypothesis points $((x_{h_i}, y_{h_i}); i=1, ..., n)$ are linearly dependent on their associated reference points $((x_{r_j}, y_{r_i}); i=1, ..., n)$, where n is the number of registration marks on the form. Hypothesized points refer to the location of registration marks found by the recognition system on a completed form. Reference points are where the marks should be located if the input image had absolutely no distortion whatsoever. A solution to these two distortion equations is sought such that they hold true for all n registration marks.

$$x_{h_i} = \Delta x + m_{x_x} x_{r_i} + m_{x_y} y_{r_i} \qquad (1)$$

$$y_{h_i} = \Delta y + m_{y_y} y_{r_i} + m_{y_x} x_{r_i} \qquad (2)$$

These two equations can be separately represented in matrix form as the linear system shown in Equation (3), where **M** is an *n*x3 matrix containing the reference points.

$$\mathbf{q} = \mathbf{M}\mathbf{p} \qquad (3) \qquad\qquad \mathbf{M} = \begin{bmatrix} 1 & x_{r_i} & y_{r_i} \\ 1 & x_{r_2} & y_{r_2} \\ \vdots & \vdots & \vdots \\ 1 & x_{r_n} & y_{r_n} \end{bmatrix} \qquad (4)$$

For Equation (1), the vector **p** contains the horizontal distortion parameters, **p**=$(\Delta x, m_{x_x}, m_{x_y})^T$, and the vector **q** contains the x-coordinates of the hypothesis points, **q**=$(x_{h_1}, \ldots, x_{h_n})^T$. For Equation (2), the vector **p** contains the vertical distortion parameters, **p**=$(\Delta y, m_{y_y}, m_{y_x})^T$, and the vector **q** contains the y-coordinates of the hypothesis points, **q**=$(y_{h_1}, \ldots, y_{h_n})^T$. **M** is the same in both systems of equations.

Equation (3) must be solved to determine the distortion parameters **p.** By using more than three registration marks, the linear system of equations becomes over-determined. However, the elements of **p** can be estimated using a method of Linear Least Squares (LSQ), whereby both sides of Equation (3) are multiplied by $\mathbf{M}^T$. The resulting term ($\mathbf{M}^T\mathbf{M}$) multiplied to **p** is square (3x3), and its inverse exists so that the LSQ solution of **p** can be written as

$$\mathbf{p} = [\mathbf{M}^T\mathbf{M}]^{-1}\mathbf{M}^T\mathbf{q} \qquad (5)$$

Distortions parameters for translation, rotation, and scale are computed by substituting the appropriate elements of **M** and **q** from either Equation (1) or (2) into Equation (5). In the case of reading HSF forms, the model recognition system uses six registration marks (*n*=6). This LSQ method computes a linear mapping that minimizes the total error between all the reference and hypothesis points, so that the impact of error at any one point is decreased as the number of points used increases, causing registration quality to improve.

The parameter estimates Δx, m_{x_x}, m_{x_y}, Δy, m_{y_y}, and m_{y_x} are substituted back into Equations (1) and (2) and the black pixels in the input form are transformed by computing (x_h, y_h). This approach is efficient because it only computes a transformation for those pixels in the image that are black.

2.1.3 Form removal

Upon registration, the pixels making up the form in the input image are known to correspond to the pixels in the registered blank form. The form information is erased from the registered input image by applying the blank form as a mask. For each pixel in the input image, an output pixel value is computed according to Equation (6), where *o* is the output pixel, *r* is the pixel from the registered input image, and *m* is the corresponding pixel from the mask. In this way, *o* is set to black only when *r* is black and *m* is white.

$$o = r \,\&\, (\sim m) \qquad (6)$$

The LSQ method for form registration minimizes error, but does not absolutely remove all error. Detection of a registration mark even within an undistorted input image may be somewhat inaccurate and there is always a certain amount of discrete round-off error when implementing pixel-based transformations. To compensate for these small amounts of error, the blank registered form image can be dilated[7] a number of times (four times in the case of HSF forms). This broadens all form structures in the blank form image so that coverage is improved when overlaid with the registered input image.

2.2 PROCESS FIELDS

2.2.1 Field initialization

A collection of data is necessary to conduct feature extraction and recognition for the specific set of characters contained in each type of field on the form. For example, when processing a numeric field, feature extraction and classification components only need to be trained to recognize digits. To process a text paragraph, feature extraction and classification components need to be trained to recognize both upper and lowercase characters. A set of functions (called basis functions) are required to compute feature vectors from each segmented character image. Also needed are the classifier weights required by the system's neural network to recognize the derived feature vectors.

2.2.2 Field isolation

Through the process of form registration, the input form has been transformed to line up with the spatial field template. The rectangular coordinates in the template are used to extract subimages of the fields from the registered input image. At times, the extracted field images contain residue left over from form removal and other types of spurious noise. Spatial histograms are computed and pixel densities within histogram bins are analyzed to distinguish black pixels comprising noise from black pixels comprising handprint. The details of these histogram analyses are documented in Reference 1. Once the edges are found, the handprint is extracted from the isolated field image.

2.2.3 Field segmentation

The feature extraction and classification techniques used by the system are designed to classify images containing a single character. Therefore, the isolated field image of multiple characters must be segmented into individual plausible character images (one character per image). To do this, the system uses connected components or *blobs* to define these plausible character images. A blob is defined to be a group of pixels all contiguously neighboring or *connecting* each other. In general, each blob is extracted and assumed to be a separate character.

The blob segmentor is used independent of the field type being processed. Unfortunately, a blob is not guaranteed to be a single and complete character. If two characters touch, then a single blob will contain both characters as a single composite image. A blob may also contain only one stroke of a character that is comprised of several disjoint pieces. For example, writers often print the top horizontal stroke of a 5 so that it does not connect the bottom portion of the digit. In this case, the two pieces of the same five will be treated incorrectly as two independent characters. To avoid this type of error, a blob pasting step has been developed for digit fields. The decision to join two blobs is based on a simple heuristic that tests neighboring blobs. The heuristic tests the *current* blob with it neighbor, the *next* blob. If the difference between the next blob's bottom coordinate minus the current blob's top coordinate is less than half the current blob's height, then the two blobs are pasted back together as a new plausible character image.

No blob pasting is conducted on segmented upper and lowercase letters; however, the segmented character images extracted from the text paragraph on the form must be re-assembled into proper reading order by organizing the blobs into their appropriate lines. It was found that the handprint in the text paragraph fluctuates significantly within lines as well as across lines, and this fluctuation is exaggerated by the use of punctuation marks, causing techniques that use global line statistics to fail.

In light of this, a localized point-to-point technique was developed to organize the segmented blobs into text lines. The location of each blob is identified by computing the geometric center of the smallest rectangle bounding the blob. These centers are used to form a 2-dimensional grid that can be used to reconstruct the line trajectories of the handprinted text. The technique developed searches the grid, taking 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 interior of the function in Equation (7). This function forms a tear-drop shaped bubble when plotted in polar coordinates that 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 in the horizontal direction. The technique uses a linear control function to continuously change the search space from circular to tear-drop in shape as the extent of the search increases. 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.

$$S = a\cos(b\theta) \qquad -\frac{\pi}{2b} < \theta < \frac{\pi}{2b} \tag{7}$$

As each blob center is located, it is either appended to an existing phrase list or a new phrase list is started. Reference 8 describes a set of heuristics used to control the search and the updating of phrase lists. Upon completion, some of the lists represent whole lines of text, and other lists represent only fragments of the text lines printed. A final merging process is conducted so that, upon completion, only lists containing complete text lines remain. The heuristics used to conduct the merging of phrase lists are also described in Reference 8. As a result of the merging, the blobs segmented from the text paragraph are gathered into lines. The last step sorts the resulting lines vertically on the y-coordinate values of the first blob in each line. When finished, the correct reading order has been reconstructed. Figure 3 shows the results of segmenting a text paragraph from an HSF form 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.

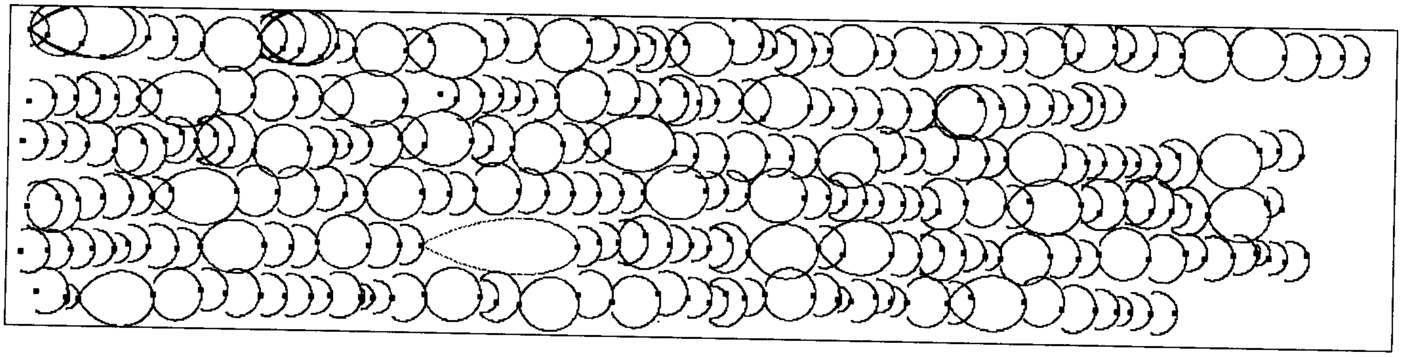


Figure 3. Traces of the bubbles used to sort the blobs into lines.

2.2.4 Character normalization

To improve the classification performance of character images, a scale-normalization technique has been developed. Scale-normalization attempts to remove size variations in handprint by scaling all segmented character images to a uniform size. The normalization method is described in Reference 1. In addition to scaling, the technique also applies a simple morphological operator in an attempt to normalize the stroke width within the character image. If the black pixel content of a character image is significantly high, then the image is eroded (strokes are thinned). If the black pixel content of a character image is significantly low, then the image is dilated (strokes are widened).

The slant within a character image is removed by a technique that uses horizontal shears in which rows in the image are shifted left or right in an attempt to straighten the character in the image. A slope factor f, defining a linear shifting function is calculated in Equation (8), where t_r is the vertical position of the top row, b_r is the vertical position of the bottom row, t_l is the horizontal position of the leftmost black pixel in the top row, and b_l is the horizontal position of the leftmost black pixel in the bottom row. The slope factor is used to compute a shift coefficient as defined in Equation (9), with r being a vertical row index in the image and m equal to the vertical middle of the image. This causes the shifting to be centered about the middle of the image. A positive value of the shift coefficient causes a row to be shifted s pixel positions to the right, and a negative value causes a row to be shifted s pixel positions to the left. When finished, the leftmost pixels in the top and bottom rows line up in the same column. By effectively reducing these two sources (size and slant) of variation, a character classifier is left to deal primarily with variations due to character shape.

$$f = \frac{t_l - b_l}{b_r - t_r} \tag{8}$$

$$s = (r - m)f \tag{9}$$

2.2.5 Feature extraction

The Karhunen Loeve (KL) transform of a segmented character image is obtained by projecting the image onto the orthonormal eigenvectors of the covariance matrix of a large number of prototype images.[9] This transform requires computing the covariance matrix, and then diagonalizing it to produce eigenvectors.[10] The KL transform has many optimal properties and is widely used in the pattern recognition field.[11] The production of this transform is also known as *principal factor* or *principal components* analysis. The resulting eigenvectors are used as basis functions for feature extraction by the system. Computing the KL transform is very expensive, so it is done once off-line, and the resulting basis functions are loaded during field initialization.

The pixels of a scale-normalized character image define a vector **u**, whose elements are obtained by considering the 2-dimensional N by N image as a vector of N^2 elements. This vector is formed by concatenating the rows of the image together, and each binary element is converted so that black pixels are represented as 1 and white pixels are represented as -1. The mean vector is computed from all the training images and is subtracted from all the training images forming a set of *sample* vectors. Each sample vector comprises a column in the sample matrix, **U**. The covariance matrix R is symmetric and is formed as the outer product of the P sample vectors as in Equation (10). The covariance matrix is diagonalized using standard linear algebra routines such as those in EISPACK[12], producing the eigenvalues and corresponding eigenvectors in descending order of largest eigenvalue. The covariance matrix **R** has N^2 eigenvectors as the columns of Ψ defined in the Equation (11), and the only nonzero elements of Λ are the eigenvalues on its diagonal. Equation (12) defines the KL transform **v** of a vector **u** to be the projection of the vector minus the mean vector μ onto the eigenvector basis Ψ.

$$\mathbf{R} = \frac{1}{P}\mathbf{U}\mathbf{U}^T \qquad (10)$$

$$\mathbf{R}\Psi = \Psi\Lambda \qquad (11)$$

$$\mathbf{v} = \Psi^T(\mathbf{u} - \mu) \qquad (12)$$

Typically, only a subset of the eigenvectors corresponding to the largest eigenvalues are used in the transformation. The initial dimensionality of **u** is N^2. By selecting only the top k eigenvectors, the dimensionality of the transformed feature vector **v** is reduced to k. For a more complete discussion of the effect of feature dimensionality please refer to Reference 13.

2.2.6 Character classification

The classification of KL features produced by projecting segmented character images onto the covariance's eigenvectors have been studied extensively.[14,15] The feature vectors are used in place of the pixels in the character images to train character classifiers. A large number of prototypes (tens of thousands) are required to train these classifiers, so they are computed off-line and loaded during field initialization.

Classification studies reported in References 13 and 15 show that Probabilistic Neural Networks[16] (PNNs) outperform more popular neural networks and statistical classifiers, such as Multi-Layer Perceptrons[17] (MLPs), in terms of accuracy. Therefore, the standard reference system uses PNN to conduct character classification. With PNN, each training vector (or prototype) $\mathbf{x}_j$ becomes the center of a kernel function that takes its maximum at the vector and decreases gradually as one moves away from the vector in feature space. An unknown feature vector **y** is classified by computing, for each class i containing M_i prototype vectors, the sum of the values of the class-i kernels at **y**, multiplying these sums by factors involving the estimated *a priori* probabilities, and finding which of L classes has the highest resulting discriminant value. PNN assigns the class with the highest discriminant value to the unknown vector **y**.

Many forms are possible for the kernel functions; we have obtained our best results using radially symmetric Gaussian kernels defined in Equation (13). In the resulting discriminant functions, σ is a smoothing parameter that may be optimized by conducting experiments on a testing set, $p(i)$ is the *a priori* probability of class i, and M_i is the number of training prototypes in class i.

$$D_i(\mathbf{y}) = \frac{p(i)}{M_i}\sum_{j=1}^{M_i} exp\left(-\frac{1}{2\sigma^2}d^2(\mathbf{y}, \mathbf{x}_j^{(i)})\right) \qquad (13)$$

The PNN algorithm, in its traditional implementation, requires all the distances $D_i(y)$ to be computed each time an unknown vector is classified. Similar methods are used in k-nearest neighbor classifiers.[13] This computation is very expensive, so up till now, the slow processing times incurred by software implementations of PNN have outweighed the accuracy benefits of the classification. Several optimizations have been added to the traditional PNN implementation in order to decrease computational intensity and improve processing times.

The first optimization takes advantage of pruning those prototypes that do not significantly contribute to the computation of discriminant values. Due to the presence of the exponential in Equation (13), the closer a training prototype is to the unknown vector, the more significant the prototype's contribution to its discriminant value. In light of this, Equation (13) can be approximated by excluding prototypes whose exponential term contributes less than $10^{-\lambda}$ times the largest term. Formally, the j^{th} prototype of any given class can be deleted according to Inequality (14), where the subscript c denotes the closest training prototype, and distances are calculated according to Equation (15).

$$d^2(\mathbf{y}, \mathbf{x}) > 2\lambda\sigma^2 \ln 10 + d^2(\mathbf{y}, \mathbf{x}_c) \qquad (14)$$

$$d^2(\mathbf{y}, \mathbf{x}) = \sum_{i=1}^{k} (x_i - y_i)^2 \qquad (15)$$

The associated error can be constrained by setting λ to a sufficiently large positive number. This parameter should not be less than log(P/L), where P is the number of prototypes, and L is the number of classes. This ensures that classification results will not change between the optimized and traditional PNN implementations.

It is important to note that the training prototype with smallest distance to the unknown vector $\mathbf{x}_c$ (thus contributing the maximum exponential term to its discriminant value) can be determined on the fly, and distances to each prototype only need to be computed once. Also, the discriminant computation can be made more efficient by taking advantage of the fact that the distance calculations in Equation (15) can be preempted once they become sufficiently large to trigger the deletion criterion. This makes pruning prototypes very efficient, which in turn greatly reduces the computation of discriminant values and achieves a factor of 4 speed up in the system.

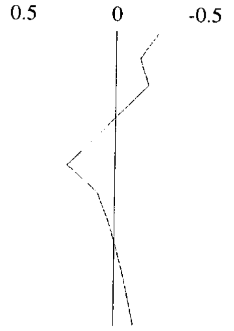
A further optimization has been integrated into the PNN classifier. This step utilizes a search tree to reduce the number of prototypes used in the PNN calculations. As stated earlier, distances must be recomputed between an unknown feature vector and a set of training prototype vectors every time an unknown vector is to be classified. This method of classification is expensive; the time is proportional to N, since N distances must be calculated. There is a large literature on faster methods.[18] Among the best are the *k-d tree* methods[19,20] of Bentley, which often have average searching time proportional to log(N). For our case, k is large and N is relatively small (N is much smaller than 2^k) and the training points are sparse in k-dimensional space. Therefore, the logarithmic behavior is not found. Some slight variations on the k-d tree give searching time proportional to sqrt(N), even for large k. While not as good as log(N), this search time is a substantial improvement over time proportional to N. A detailed description of this search method is presented in Reference 21.

The k-d tree is traversed producing a relatively small yet viable set of prototypes. This small set of prototypes is then used to calculate approximated PNN discriminant values according to the deletion criterion defined in Inequality (14). In very rare cases, no close prototypes are found in the tree search. When this occurs, all the training prototypes are used in the approximated PNN calculation. The PNN exponential activations are normalized to estimated probabilities by dividing by their sum and used as classification confidence values. The standard reference system uses the optimized version of PNN to achieve a factor of 20 improvement in processing time over the traditional PNN[16], and the speed improvement is realized without any loss in classification accuracy. The optimizations introduced now enable applications to capitalize on the robustness of the PNN algorithm without compromising processing time.

2.2.7 Dictionary-based postprocessing

When processing textual fields, other types of context, such as dictionaries and language models, can be used to improve the accuracy of the system. The text paragraph at the bottom of an HSF form contains a handprinted rendition of the Preamble to the U.S. Constitution. This text is comprised of 38 unique words that are compiled into a short dictionary. A postprocessing technique illustrated by the example shown in Figure 4 uses the dictionary to detect words within a text line that contains combinations of segmentation and classification errors.

The second column in the figure lists a *fan-out* of hypothesized words beginning with the character S and adding one successive character from the text line "STCTESLNORDE", forming a new hypothesized word on each row down the column. The third column lists the best match from the dictionary for each hypothesized word in the second column. The fourth column lists alignments that are produced using the Levenstein distance[22] to match the hypothesized word to the dictionary 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.



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 4. Signals generated from a fan-out of hypothesized words.

These signal values listed in the first column and plotted in the graph are used to detect words within the fan-out. A signal value, s, is computed from two terms, e and t as defined in Equation (16). The first term, e, is an error term and is computed according to Equation (17), 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 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 the linear function $t=T(p)$ that biases longer hypothesized words over shorter ones. In this way, matches to the word "DOMESTIC" are favored over matches to the word "DO". 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, p.

$$s = 1.0 - e - t \qquad (16)$$

$$e = \frac{n}{l - g} \qquad (17)$$

The signals listed in the first column of Figure 4 are searched top to bottom, and only those hypothesized words with s>0 are considered to contain possible words. The hypothesized word with the largest positive 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. In the provided example, the hypothesized word "STCTES" is selected with a maximum signal of 0.254, the corresponding dictionary match "STATES" is output by the system, and fan-out processing resumes with L (starting from the position in the text line "LNORDE ..."). A complete description of this technique is provided in Reference 8. Through this approach, segmentation and classification errors are corrected, and word boundaries are automatically identified.

2.2.8 Store recognition results

The text recognized within each field on the form is stored as a hypothesis string. The raw classifications produced by the character classifier are stored for digit, upper and lowercase fields. A real-valued confidence value is also stored for each character classification reported. This enables rejection models to be run on the system's output. The results stored for the text paragraph are the words identified and corrected by the dictionary-based postprocessing. No confidence values are stored for this type of field. Writers completing the forms were instructed to handprint the font information provided above each field. Therefore, the fields are self-referenced, minimizing the cost of labeling the data entered on each form and automatically providing reference strings for measuring recognition performance.

3. PERFORMANCE AND TIMING STATISTICS

NIST has developed a recognition system testing methodology that has been implemented as the NIST Scoring Package[23-26]. The scoring package has been developed to measure the performance of character recognition systems and automated form processing systems. Recognition performance is measured by reconciling the system's hypothesized field values to reference field values (the characters the writer was instructed to enter in the field).

In a sample of the first 500 writers from SD1, the standard reference system achieves a character output accuracy of 92.9% on numeric fields with no character rejections. Character output accuracy, defined as *CHAR8* in Reference 26, divides the number of segmented character images correctly classified by the total number of characters that can possibly be recognized on the completed forms. The system achieves a character output accuracy of 75.3% on lower case fields and 84.5% on upper case fields without the use of context-based postprocessing. The standard reference system achieves a character decision accuracy of 95.4% on numeric fields with no rejections. Character decision accuracy, defined as *CHAR3* in Reference 26, divides the number of segmented character images correctly classified by the total number of segmented character images presented to the system's classifier. Characters deleted by the system are not included in this metric. At a rejection rate of 4.6%, the system achieves a character decision accuracy of 97.4% on numeric fields.

The system achieves a field accuracy of 79.1% on numeric fields with no characters rejected. Field accuracy, defined as *CHRFLD1* in Reference 26, divides the total number of fields correctly recognized by the total number of fields processed by the system. In order for a field to be considered correctly recognized, no remaining characters in the field value after rejection can be substituted, inserted, or deleted. The recognition system achieves a word accuracy of 60.5% when applying a limited-size dictionary to the character classifications made on the text paragraph. The word accuracy is computed by tokenizing each word, using the Scoring Package to align the word tokens, and then accumulating the number of substituted, inserted, and deleted words.

In February of 1994, the Second Census Optical Character Recognition Systems Conference was sponsored by the Bureau of the Census and run by NIST. Ten different organizations submitted system results on the task of recognizing a small handprinted portion of the 1990 Census Long Form from both microfilm and paper. This part of the form contains three questions related to occupation. The details of the conference and the conclusions drawn from the results are presented in Reference 2.

The NIST standard reference recognition system is similar to the NIST system used in the conference. But despite their similarities, the application of these two systems is significantly different. First, the image quality of the HSF forms distributed with SD1 is better than the quality of images scanned from the Census Long Forms. Second, the standard reference recognition system uses a limited-size dictionary when processing the text paragraphs. This is in contrast to the conference where dictionaries of more than 60,000 multiple-word phrases were used.

Nonetheless, some comparisons can be made at the word recognition level. The word accuracy of the standard reference system was 61% on the HSF form's text paragraph. The average field in the conference contained two words so that this level of accuracy, if sustained on the conference test, would have resulted in a 37% field accuracy rate. In the conference, NIST achieved a 25% field accuracy. Based on these numbers, it is probable that the standard reference recognition system is better than the conference system. The expected word accuracy for the best conference system was about 76%, so on a word basis we would expect the standard reference recognition system to have about 15% (76%-61%) more errors than the best conference system. This is comparable to the median system reported at the conference. The difference between the best conference systems and the NIST standard reference system is primarily due to the fact that the standard reference system does not use any techniques for oversegmenting characters and reconstructing words.

Figure 5 lists all the different computers on which the recognition system was successfully compiled and tested. The last column in the table shows the average user time required by each machine to process a single form. These averages were compiled from the times produced on 10 different forms.

Man.	Model	O.S.	# Proc.*	RAM	Time
DEC	Alpha	OSF/1 V1.3	1	32 Mb	28.3
HP	Model 712/80	HP-UX 9.03	1	64 Mb	31.4
IBM	RS6000	AIX 3.2.5	1	128 Mb	27.4
SGI	Challenge (IP19)	IRIX 5.2	8	512 Mb	22.9
SGI	Indigo 2 (IP22)	IRIX 4.0.5H	1	128 Mb	26.4
SGI	Onyx (IP19)	IRIX 5.1.1.3	4	512 Mb	22.4
Sun	SPARCserver 4/470	SunOS 4.1.1	1	32 Mb	125.9
Sun	SPARCstation IPC	SunOS 4.1.2	1	8 Mb	169.5**
Sun	SPARCstation 2 (Weitek 80MHz CPU)	SunOS 4.1.3	1	64 Mb	81.8
Sun	SPARCstation 10	SunOS 4.1.3	1	32 Mb	63.0
Sun	SPARCstation 10	SunOS 5.2 (Solaris)	2	128 Mb	39.6

Figure 5. Table of timings from different computers on which the standard reference recognition system was tested.
*All computers, including those with multiple processors, were compiled and tested serially.
**The Sun IPC was run in a small memory mode due to its limited memory.

4. SYSTEM SUMMARY

This report documents the NIST standard reference form recognition system in terms of its components and functionality.[1] The system has been successfully compiled and tested on a number of different vendors' UNIX workstations. The system's source code is written in C and is organized into 11 libraries. In all, there are approximately 19,000 lines of code supporting more than 550 subroutines.

Source code is provided for a wide variety of utilities that have application to many other types of problems. These utilities include form registration, form removal, field isolation, field segmentation, character normalization, feature extraction, character classification, and dictionary-based postprocessing. The system contains a number of significant contributions to OCR technology, including an optimized PNN classifier that operates a factor of 20 times faster than traditional software implementations of the algorithm. The modular design of the standard reference system makes it useful for OCR benchmarking, component development and testing, training and testing set validation, and multiple system voting schemes.

Distributions of the standard reference recognition system can be obtained free of charge on an ISO-9660[27] format CD-ROM by sending a letter of request to the primary author. Any portion of this system may be used without restrictions. The system software was produced by NIST, an agency of the U.S. government, and by statute is not subject to copyright in the United States. Recipients of the standard reference recognition system assume all responsibilities associated with its operation, modification, and maintenance.

5. 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, National Institute of Standards and Technology, July 1994.

2. J. Geist, R. A. Wilkinson, S. Janet, P. J. Grother, B. Hammond, N. W. Larsen, R. M. Klear, M. J. Matsko, C. J. C. Burges, R. Creecy, J. J. Hull, T. P. Vogl, C. L. Wilson, "The Second Census Optical Character Recognition Systems Conference," Technical Report NISTIR 5452, National Institute of Standards and Technology, May 1994.

3. C. L. Wilson and M. D. Garris, "Handprinted character database," *NIST Special Database 1*, vol. HWDB, National Institute of Standards and Technology, Apr. 1990.

4. M. D. Garris, "Design and collection of a handwriting sample image database," *Social Science Computer Review*, vol. 10, no. 2, Duke University Press, 1992, pp. 196-214.

5. M. D. Garris and D. L. Dimmick, "Evaluating form designs for optical character recognition." Technical Report NISTIR 5364, National Institute of Standards and Technology, Feb. 1994.

6. C. R. Wyle, *Advanced Engineering Mathematics*, second edition, New York: McGraw-Hill, 1960, pp. 175-179.

7. A. K. Jain, *Fundamentals of Digital Image Processing*, New Jersey: Prentice-Hall, 1989, pp. 384-389.

8. M. D. Garris, "Unconstrained handprint recognition using a limited lexicon," in *Proc. Document Recognition*, vol. 2181, pp. 36-46, SPIE, San Jose, Feb. 1994.

9. P. J. Grother, "Karhunen Loeve feature extraction for neural handwritten character recognition," in *Proc. Applications of Artificial Neural Networks III*, vol. 1709, pp. 155-166. SPIE, Orlando, Apr. 1992.

10. 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, 1989.

11. K. Fukunaga, *Introduction to Statistical Pattern Recognition,* second edition, New York: Academic Press, 1990.

12. B. T. Smith, et al., "Matrix eigensystem routines - EISPACK Guide," second edition, vol. 6 of *Lecture Noted in Computer Science*, New York: Springer-Verlag, 1976.

13. J. L. Blue, G. T. Candela, P. J. Grother, R. Chellappa, and C. L. Wilson, "Evaluation of pattern classifiers for fingerprint and OCR applications," *Pattern Recognition*, vol. 27, no. 4, pp. 485-501, 1994.

14. 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 *Conf. on Character Recognition and Digitizer Technologies*, vol. 1661, pp. 269-280, SPIE, San Jose, Feb. 1992.

15. P. J. Grother and G. T. Candela, "Comparison of handprinted digit classifiers," Technical Report NISTIR 5209, National Institute of Standards and Technology, June 1993.

16. D. F. Specht, "Probabilistic Neural Networks," *Neural Networks*, vol. 3(1), pp 109-119, 1990.

17. 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*, vol. 1, Cambridge: MIT Press, 1986, pp. 318-362.

18. For a bibliography and selected papers, see, for example, B. V. Dasarathy, "Nearest neighbor (NN) norms: NN pattern classification techniques," IEEE Computer Society Press, 1991.

19. J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, pp. 509-517, 1975.

20. J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Transactions on Mathematical Software*, vol. 3, pp. 209-226, 1977.

21. J. L. Blue, "Fast nearest-neighbor searches in high dimensional spaces," National Institute of Standards and Technology, to be published.

22. H. G. Zwakenberg, "Inexact alphanumeric comparison," *The C Users Journal*, pp. 127-131. May 1991.

23. M. D. Garris and S. A. Janet, "Scoring Package release 1.0," *NIST Special Software 1*, vol. SP, National Institute of Standards and Technology, Oct. 1992.

24. 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, Oct. 1992.

25. 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, Feb. 1993.

26. 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, Aug. 1993.

27. ISO-9660, "Information processing - volume and file structure of CD-ROM for information interchange," Standard by the International Organization for Standardization, 1989.