Massively parallel implementation of character recognition systems

M. D. Garris, C. L. Wilson, J. L. Blue, G. T. Candela, P. Grother, S. Janet, and R. A. Wilkinson

National Institute of Standards and Technology,
Gaithersburg, Maryland 20899

Presented at SPIE's Conference on
Machine Vision Applications in Character Recognition and Industrial Inspection
San Jose, 1992

ABSTRACT

A massively parallel character recognition system has been implemented. The system is designed to study the feasibility of the recognition of handprinted text in a loosely constrained environment. The NIST handprint database, *NIST Special Database 1*, is used to provide test data for the recognition system.

The system consists of eight functional components. The loading of the image into the system and storing the recognition results from the system are I/O components. In between are components responsible for image processing and recognition. The first image processing component is responsible for image correction for scale and rotation, data field isolation, and character data location within each field; the second performs character segmentation; and the third does character normalization. Three recognition components are responsible for feature extraction and character reconstruction, neural network-based character recognition, and low-confidence classification rejection.

The image processing to load and isolate 34 fields on a scientific workstation takes 900 seconds. The same processing takes only 11 seconds using a massively parallel array processor. The image processing components, including the time to load the image data, use 94% of the system time. The segmentation time is 15 ms/character and segmentation accuracy is 89% for handprinted digits and alphas. Character recognition accuracy for medium quality machine print is 99.8%. On handprinted digits, the recognition accuracy is 96% and recognition speeds of 10,100 characters/second can be realized. The limiting factor in the recognition portion of the system is feature extraction, which occurs at 806 characters/second. Through the use of a massively parallel machine and neural recognition algorithms, significant improvements in both accuracy and speed have been achieved, making this technology effective as a replacement for key data entry in existing data capture systems.

1. INTRODUCTION

Improved recognition algorithms and the increased performance of computers has touched off an imaging revolution, offering a new method of information archival, retrieval, and processing for many existing and new applications. This paper presents a model system developed for automated document processing and data capture using neural network algorithms on a massively parallel computer. The recognition system has been designed to read handprinted information from structured forms, that is documents on which entry fields are consistently located, so that upon correct identification of the form type, relative field locations can be predicted.

Traditionally, documents of this nature have been stored as paper archives. Now they are beginning to be digitally scanned and archived onto magnetic or optical media as binary images. These documents are collected at centralized locations within a short period of time, such as during a decennial census, and then processed at a later date. Therefore the recognition system developed at NIST has been designed to process archived images independent of when digitization occurs. A massively parallel machine has been used in order to obtain high throughput and to study the feasibility of using new computer architectures for this type of application.

Character recognition, the classification of well formed and cleanly segmented characters, has been studied in great detail in the past.[1-5] In this paper, the complete design and performance of a working handprint recognition system is studied. The results suggest that the recognition component is only one of many important components needed to successfully convert laboratory research into an effective application technology. Areas in need of development include high-speed, high-bandwidth peripheral interfaces and the development of intelligent and efficient segmentation schemes. Section 2 defines the hardware architecture supporting the system. Section 3 lists and describes the functional components of the system. Section 4 discusses the software design of the system. Section 5 presents results based on testing the system using *NIST Special Database 1*.[6,7]

2. HARDWARE ARCHITECTURE

This model recognition system is implemented across two integrated computers.* The data storage and central processing control is supported by a Sun 4/470 UNIX server. The Sun has 32 Megabytes of main memory, approximately 10 gigabytes of magnetic disk, and two CD-ROM drives. Connected to the Sun 4/470 is an Active Memory Technology 510C Distributed Array Processor[8]. The parallel machine is a Single Instruction Multiple Data (SIMD) architecture and consists of two separate 32 by 32 grids of tightly coupled processors. One grid contains 1-bit processing elements 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 all low-level computationally intensive system tasks.

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

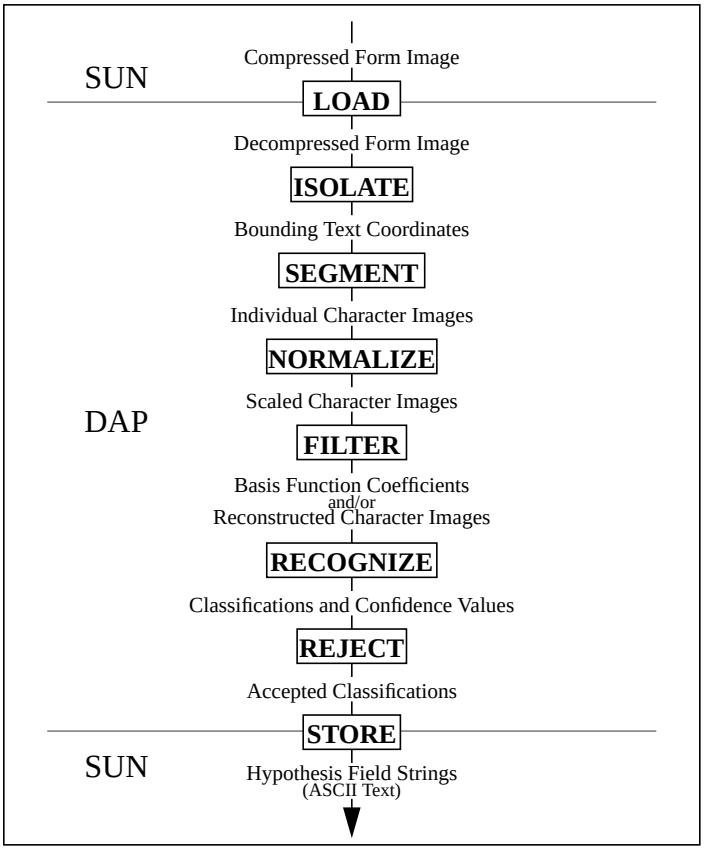

Fig. 1. The functional components of the recognition system distributed between the serial server and the massively parallel computer.

* 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.

3. FUNCTIONAL COMPONENTS

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

3.1 Load module

As stated in Section 2, the serial server is responsible for data storage and central processing control. An image requiring recognition is accessed by the server and then transmitted to the parallel computer. This process is represented by the load module in Figure 1. The serial server opens the image file, reads the image into its main memory, decompresses the image data if necessary, and transmits the image data and size attributes to the parallel computer.[7] The serial server then instructs the parallel computer to convert the received image data into the parallel computer's own binary representation. Image decompression is currently being implemented on the parallel machine in order to reduce the size of the data being transmitted and to speed up decompression. The remaining modules, except for the store module, run exclusively on the massively parallel computer.

3.2 Isolate module

Once the image is in the array memory of the parallel machine, global image correction for scale and rotation can be performed. Images of documents contain significant rotation, translation, and scale distortions introduced by photocopying and digitization. The isolate module first detects these global degradations and then de-skews the image. Next a spatial template defining the location of each entry field on the form is adapted to the image. The recognition system described in this paper has been set up to process forms of a single type, those from *NIST Special Database 1*. A form type identification module would be needed in order process images of multiple form types. The results from a form type identification module would dictate what spatial template is appropriate for a given input document image. Once the spatial template has been applied to the image, pixel coordinates bounding the handprint within each entry field are found. The entire process of de-skewing and handprint isolation involves column and row-oriented pixel shifts and localized histograms, all of which are extremely efficient on the parallel computer.

3.3 Segment module

The recognition component of the system classifies individual characters, which requires isolated handprint to be separated into individual images, one character per image. NIST has conducted significant research on segmenting text images with a massively parallel computer.[8] The results of this work have been incorporated as a segment module within the recognition system. The current module utilizes traditional image processing techniques, exploiting the use of spatial histograms, connected component labeling, and adaptive cutting statistics. The segmentation time is 15 ms/character, and segmentation accuracy is 89% for handprinted digits and alphabetic characters. More intelligent and efficient (neural-based) methods of segmentation are currently being researched at NIST and elsewhere.[9-13] Upon successful segmentation, each handprinted character is copied to a separate character image.

3.4 Normalize module

The processing elements in the parallel computer are arranged in a 32 by 32 grid. Operations on vector and matrix data types of arbitrary dimension are provided by the compiler, allowing the machine to pipeline its processing and to compute optimally when vector and matrix dimensions are kept at multiples of 32. Therefore, all segmented characters are spatially normalized to fit within a 32 by 32 pixel image. Segmented character images bounded by a rectangle smaller than 32 by 32 are scaled up while character images bounded by a rectangle larger than 32 by 32 are scaled down. In both cases, the original character's aspect ratio of width to height is preserved. The normalize module provides uniformity in size and position for further processing and is very inexpensive on the parallel machine.

3.5 Filter module

One of the most complex modules in the recognition system is the filter module. Most issues identified by neural network experts as preprocessing and information representation are represented in this single system module. This component takes individual character images and filters the image data to provide an input vector to a recognition classifier, which is a neural network for the system studied in this paper. The filtered data may be an enhanced character image and/or a reduced feature vector representing the input character image. The filter options applied in the recognition system presented in this paper include a shear transform for removing the slant from handprint, spatially tuned Gabor basis functions,[1,14] and ranked principal component functions using the discrete Karhunen Loeve (KL) transform.[15] They may be used independently or in combination.

Gabor functions are a set of incomplete nonlinear functions which reduce random image noise and smooth irregularities in image structure by acting as spatially localized low-pass filters. Gabor functions provide the minimum combination of uncertainty in position and spatial frequency resolution, and they match the visual receptor field profiles of mammalian eyes.[16] The KL transform is a statistical method that expands characters in terms of their variance. It assumes no model of the human perception mechanism, but directly references how, statistically, handprinted characters are formed. The eigenvectors of the covariance matrix formed from a sample of characters are the principal components of this variance; those with the highest eigenvalues are more relevant descriptors of the character images. The eigenvectors can be derived by applying one of the traditional numerical methods for eigensolutions of the covariance matrix. The eigenvectors form a minimal orthogonal basis set of which any character is a linear combination.

Gabor and KL basis functions can be used in two different ways. Each basis function applied separately to the character image produces a coefficient value. A feature vector of coefficient values can be computed by applying a set of basis functions to a single character image. This feature vector can then be used in place of the original character image by a neural network. The basis functions can alternatively be used to reconstruct the character image. Computing a feature vector and reconstructing a character image can be done at 806 characters/second on the massively parallel computer. The coefficient value vectors are especially useful in reducing the input dimensionality to multi-layer perceptrons (MLP). For example, 32 tuned Gabor functions, when individually applied to a 32 by 32 pixel image, produce a vector of 32 coefficient values. A network can then be trained on these 32 coefficients rather than on the original 1,024 pixel character image. This dimensional reduction is important for the generalization capabilities of the network.[17] The reconstructed characters are especially useful to self-organizing shape-based classifiers. For example, a Gabor reconstructed character is enhanced by emphasizing the body of the character, reducing both the variations along its edges due to digitization and by normalizing its stroke width.[14]

3.6 Recognition module

The functionality of the recognition module has been the focus of most published character recognition research. When most people speak of character recognition, they are referring to tasks performed by this single module. As can be seen in Figure 1, this highly studied component is just one of many modules necessary to implement a successful automated document processing and data capture system. This module takes input from the filter module as described above and classifies it. The recognition system designed at NIST classifies filtered data via one of several neural networks.

One of these neural networks used in the recognition system is a self-organizing algorithm developed at NIST called Feedforward Association Using Symmetrical Triggering (FAUST).[18] FAUST provides a parallel, multi-map, self-organizing, pattern classification procedure similar to those known to exist in the mid-level visual cortex.[19] This neural network uses a feed-forward architecture which allows multi-map features stored in weights acting as associative memories to be accessed in parallel and to trigger a symmetrically controlled parallel learning process. This method allows features of different data type, such as binary image patterns and multi-bit statistical correlations, to be updated in parallel. The three essential features of FAUST are: 1) Different feature classes use individual association rules for pattern comparison. 2) Different feature classes use individual learning rules for pattern modification. 3) All feature classes contribute symmetrically to learning.

The use of a multi-layered perception, a more traditional neural network architecture, has been also been studied.[20] This network classifies by generating feedforward activations across a network containing an input layer, one hidden layer, and an output layer. Using the MLP architecture trained with Gabor feature vectors, character recognition accuracy for medium quality

machine print is 99.8%.[14] Using the MLP architecture trained with KL feature vectors, handprinted digit recognition accuracy is 96% and recognition speeds of 10,100 characters/second can be realized on the massively parallel computer.

In all, three different recognition modules have been analyzed in the recognition system. These include FAUST recognition on sheared character images (FAUST system), MLP recognition of Gabor feature vectors (Gabor system), and MLP recognition of KL feature vectors (KL system). The use of a modular system design, discussed later in this paper, enables easy integration of different recognition modules. Note that the training of these different networks is done once, off-line. The FAUST system was trained on 2,000 unique character images from 50 different writers resulting in 256 self-organized associative memories for both matched patterns and statistical relevance in pixel position. Both the recognition modules of the Gabor system and the KL system were trained using Scaled Conjugate Gradient (SCG).[21] The Gabor system was trained on the same data set as the FAUST system while the KL system was trained on a much larger data set, 7,400 character images from 940 different writers. The difference in training set sizes is strictly historical. Training for the FAUST system takes about 5 minutes on the parallel computer. A typical SCG training session using the massively parallel computer converges within 10 minutes of initial computation. Once a network has been trained, its weights are archived and later loaded as part of the setup at the beginning of a recognition system run.

3.7 Reject module

The model recognition system has a component responsible for pruning low-confidence classifications from the system. The reject module tags confusable classifications as unknown recognitions. The current reject mechanism is simply implemented as a static threshold applied to the winning activation from the recognition module. Analysis has shown that the recognition networks are more accurate in classifying some characters than with others. Therefore, the system takes an independent threshold scalar, one for each character class being recognized. These thresholds can be tuned to a given application's tolerance of recognition errors. In the case of digit recognition, there are 10 thresholds. Work is currently underway to create a dynamic reject module using a neural network. By tagging confusable recognitions, postprocessing such as manual key data entry and context correction can be applied to the system's hypothesized text.

3.8 Store module

The final component of the recognition system is the store module. Like the load module, this component interacts between the parallel machine and the serial server. Both hypothesized text and confidence levels for each character are transferred to the server for storage and analysis. The serial server requests the data resident on the parallel machine be converted to the binary representation of the server, and then requests the data be transmitted. The server formats the hypothesized text into a report by entry field and stores the report to disk. NIST has developed a testing methodology for analyzing the performance of recognition systems. By presenting a system with referenced image data, automated performance analyses can be computed on the system's output. The description of this methodology is left to a future paper.

3.9 Component Observations

At this point there are two observations to be made. First, in Figure 1 there is a consistent and dramatic reduction in the volume of data flowing through the system from input to output. The original binary document images have been digitized at 300 pixels per inch. At input, the system is given 8,000,000 pixels of binary image data, which after successful isolation and segmentation are reduced to about 133,120 pixels. After feature extraction in the filter module, the segmented character images are reduced to 33,280 bits of coefficient values, and after recognition the classifications are stored as 1,040 bits of ASCII text. This represents an 8,000 to 1 data compression from input to output. Each successive module contributes to the reduction of problem complexity and/or required data bandwidth. A second thing to note is that this initial system's modules are essentially linearly connected. Future system embellishments may include the development of more sophisticated feedback and error recovery loops.

4. SOFTWARE DESIGN

The model recognition system is comprised of approximately 22,500 lines of source code. About 12 man years have been invested in the development of the current system. Figure 2 lists the amount of source code supporting each of the system modules. Notice that the off-line training accounts for about 10,000 lines of code. The source code written in the 'C' programming language

on the serial server is primarily responsible for two things. The server's software controls the system by invoking system modules on the parallel computer, and it defines the file formats required to load and store data to and from the parallel machine. The system modules implemented on the parallel computer are written in parallel FORTRAN. The modules carry out the tasks described in Section 3. There is a great deal of modularity built into the software design, and shared libraries are used extensively.

COMPONENT	'C'	FORTRAN
MAIN	328	
LOAD	2,537	1,111
ISOLATE		1,218
SEGMENT		797
NORMALIZE		1,110
FILTER		1,770
RECOGNIZE		645
REJECT		52
STORE	2,087	850
TOTAL	4,952	7,553

TRAINING	'C'	FORTRAN
FAUST	1,939	5,316
MLP	1,224	1,461
TOTAL	3,163	6,777

Fig. 2. The left table lists the lines of source code supporting the recognition system's modules. The right table lists the lines of source code supporting the off-line training of the neural networks.

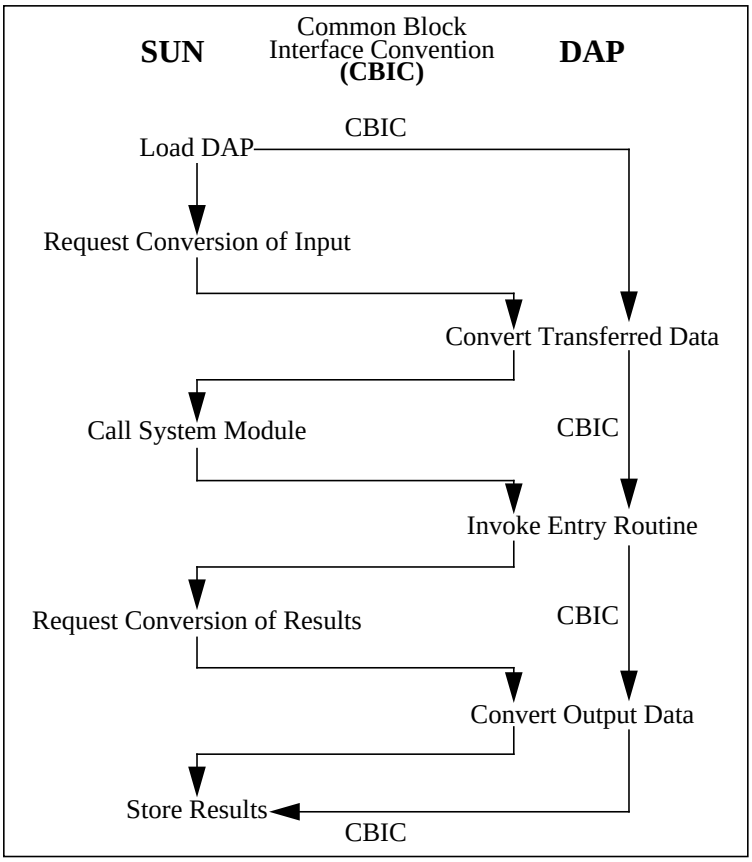

Fig. 3. The modular design used to build and test the recognition system.

Figure 3 illustrates the modular design used to test and integrate recognition system components. The figure shows the interface enveloped around a single system module whose executable code is resident on the parallel computer. This diagram doc-

uments the communication control and data flow between the serial server and the parallel computer. The right and outermost path charts the flow of data through the module, and the left and innermost path charts the control flow between the two computers. The serial server is represented on the left portion of the figure and the parallel computer is represented on the right portion of the figure.

The data interface on the parallel computer is implemented in parallel FORTRAN as global common blocks represented as the Common Block Interface Convention (CBIC) in Figure 3. Basically each system module's inputs and outputs, whether scalar, vector, or matrix, are organized into a set of common block definitions. These global memories remain resident on the parallel machine and are accessible across successive invocations of modules on the parallel machine as well as accessible by the serial server. Figure 3 illustrates a test harness for a single system module. Following the data path, right and outermost, the serial server downloads the parameters and data required as input to the system module onto the parallel machine into a set of predetermined common blocks. The data once loaded onto the parallel computer is then converted to the binary representation used by the parallel computer. The system module is executed and the results are stored into a predetermined set of output common blocks. The results are converted to the binary representation of the serial server. The results are then transmitted from the parallel computer to the serial server for storage and analysis.

This entire process is controlled by the serial server. Following the control path, left and innermost, the serial server transfers the system module's parameters and input data to the parallel computer. The server requests the parallel computer to convert the transferred information. The server invokes the system module resident on the parallel computer. Upon completion, the server requests the module's results on the parallel computer be converted to the serial server's binary representation. Then the server receives and stores the converted results.

This control design and data flow enables easy test harnessing of the system modules resident on the parallel computer. This has been extremely useful in developing and testing different system modules in isolation from the rest of the system. Countless hours of tedious debugging have been preempted with this system design. In addition to test harnesses, this software design enables modular configuration of entire recognition systems. Systems can be compiled by simply stacking multiple modules of the type shown in Figure 3 and connecting the data flows so that the output common blocks of the previous module become the input common blocks to the next module. This is exactly how the three versions of the recognition system studied in this paper were made. In order to optimize the system, system module parameters such as the pre-trained weights for the recognition module are loaded onto the parallel computer once as an initial system setup stage. The CBIC allows different system modules to be developed and tested independently and yet ensures that they compile directly into the system. This technique has been used to develop and test different algorithms for the same module including segmentation, recognition, and rejection. In this way, new techniques can be explored and determined to be superior. It may also be found that certain techniques are superior only for certain applications under certain conditions. In either case the modules are simply *plugged* into the system.

5. RESULTS ON *NIST SPECIAL DATABASE 1*

5.1 System demo

Figure 4 displays one of the forms from *NIST Special Database 1*. There are 2,100 forms in the database completed by 2,100 different writers. Each form is similar to the one shown. These forms collectively contain 58,800 numeric fields, 28 per page, and potentially 273,000 digits. The recognition system was designed to locate and process these fields.

Figure 5 is a screendump of a recognition system demo after processing a single form. In this demo, the KL system configuration is being used. The display is split into two windows. The portion of the form containing the numeric fields is displayed in each. The top window shows the results of global de-skewing and field isolation. Each form in this database contains some amount of rotation, translation, and scale noise. The results of de-skewing can be seen by examining how the left margin along the boxes lines up with the edge of the figure. The results of field isolation are also shown in the top window. Each entry field has a bounding rectangle drawn around the handprint contained therein. The image processing to load and isolate 34 fields on a scientific workstation takes 900 seconds. The model recognition system takes only 11 seconds, 9 seconds for the serial computer to decompress and load the image onto the parallel computer, and 2 seconds for the parallel computer to isolate the text in all 34 entry fields.

The bottom window displays the results of segmentation and recognition. The machine printed characters displayed in reverse video represent the recognition system's classifications. The box containing "920" in the middle of the last row of entries fields contains no reverse video characters whatsoever. This is a field which did not segment successfully. Notice how the "2" and "0" touch. This is a known problem with the current segment module. Now examine the last box in the second row of entry fields. Where the recognition module should have classified a segmented "4", there is an empty black space. This represents a confusable recognition which had sufficiently low confidence so that it was rejected and classified as unknown. By examining the corresponding character in the top window, one can see that the "4" is malformed with significant tilt and strange curvature along the horizontal stroke. The same is true for the "2" in the third box in the fourth row of entry fields.

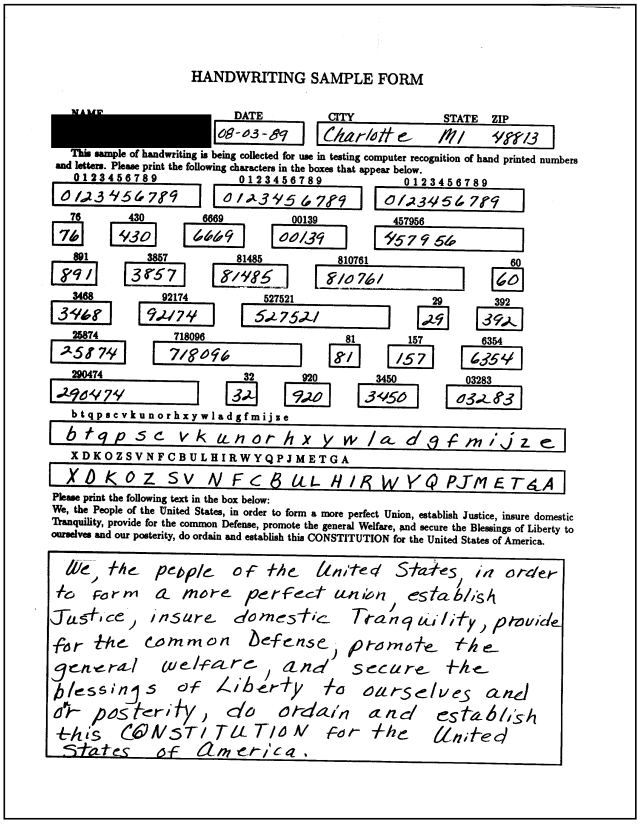

Fig. 4. A completed Handwriting Sample Form from *NIST Special Database 1*.

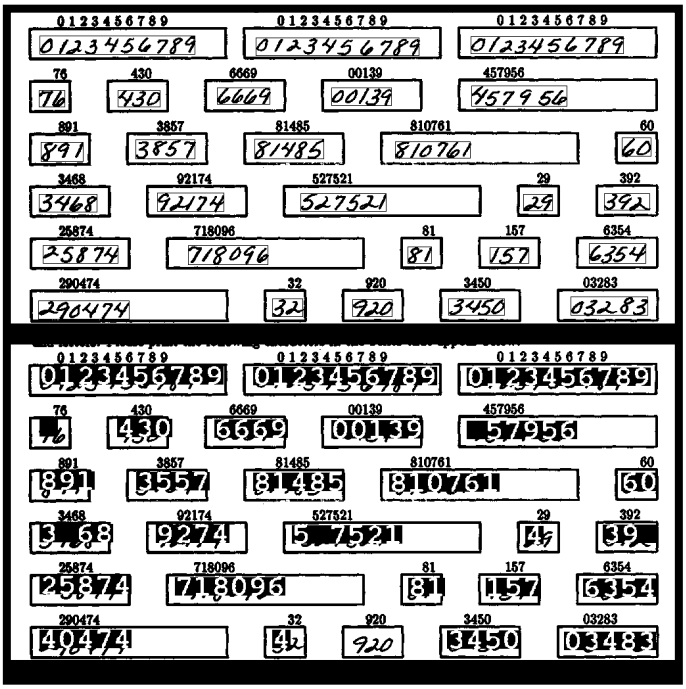

Fig. 5. A screendump of the KL system having processed the form in Figure 4.

There are two other examples of segmentation errors in concert with recognition errors. The first box in the second row, "76", has a single black space entered in the bottom window. A single black space implies that the segment module was unable to separate the two characters. Correct segmentation would have produced two reverse video characters or spaces. The two-character image containing both "7" and "6" is still of a reasonable size for a single character image. As a result, the incorrectly segmented image was sent to the recognition module. The recognition module classified the two digit image with low confidence, therefore it was rejected. However, this is not the case with the fourth box in the fourth row, "29", and the first box in the last row, "290474". Here, the segment module was unable to separate the two digits "29". The composite image was therefore sent to the recognition module and in both cases the two-digit image was classified as a "4". If one examines the handprint with squinted eyes, the shape of a four can be seen. The tail of the "2" serves as the horizontal stroke of a normal "4" when extended to the "9". This is an example of naturally occurring ambiguities in handprinted characters.

The most intolerable error for most automated recognition applications are substitution errors. An example of this type of error is found in Figure 5 in the second box of the third row of entry fields. The second character, a handprinted "8", was incorrectly classified as a "5". Examining the handprint in the top window helps explain why the system has problems with this character. The recognition module's confidence was unfortunately too high to be caught by the reject module and so the confusion

went through undetected. This type of undetectable error will corrupt automatically generated computer databases. Setting the reject thresholds higher reduces the risk of these types of errors at the expense of rejecting more and more correct classifications thereby reducing the system's effective throughput.

5.2 Accuracy analysis

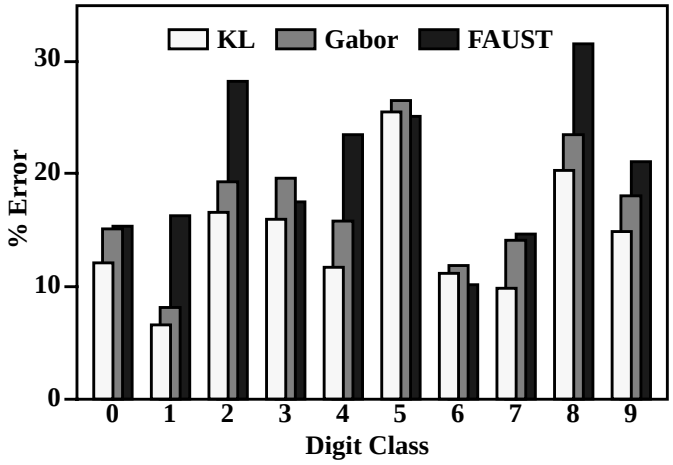

Fig. 6. The accuracy of the 3 system configurations by digit across the entire testing sample.

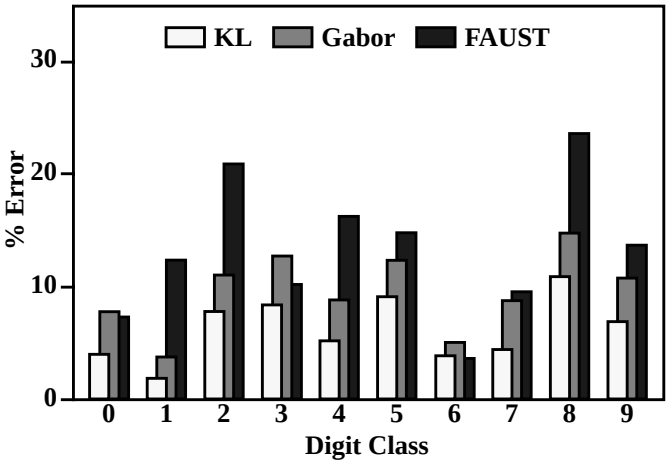

Fig. 7. The accuracy of the 3 system configurations without segmentation errors.

Figures 6 and 7 plot the performance of the three different system configurations. The performance is plotted by digit class. The horizontal axis plots the 10 digit classes. The vertical axis plots percentage error of the system. The lightest bars correspond to the KL system configuration, the medium gray bars correspond to the Gabor system configuration, and the darkest bars correspond to the FAUST system configuration. This data has been compiled by running each system configuration across the entire 2,100 forms in *NIST Special Database 1*. Each unique system processed 58,800 fields totalling 273,000 characters. Figure 6 plots the performance of the system across the entire testing sample, representing the classification of 819,000 digits. Figure 7 plots the performance of the system on just the entry fields which had the correct number of images segmented as there are characters in each field's reference string. In this way, the effect of segmentation accuracy on the system's performance can be studied. Figure 7 represents the system performance with the majority of segmentation errors removed from the analysis.

In Figure 6, the FAUST system's recognition decision accuracy across all 10 digit classes was 79%, the Gabor system's accuracy was 82%, and the KL system's accuracy was 85%. These percentages reflect the accuracy of the entire end-to-end system shown in Figure 1 across the 273,000 character testing sample. Each error occurring in any one of the system's modules propagates forward through the system. For example, if the system's segmentation module is 90% accurate and its recognition module is 90% accurate then one can expect the end-to-end system's accuracy to be no better than (0.9 * 0.9) = 81%. That is precisely the behavior of the working recognition system. The segment module has been analyzed and found to be 89% accurate on handprinted digits and alphas. The KL system, when tested in isolation on cleanly segmented characters is 96% accurate. Multiplying the two accuracies together, one gets 85% which is precisely what our end-to-end system analysis produces. This shows there are no other significant error sources in the recognition system implying that all system modules except for the segment and recognition modules are virtually 100% accurate.

The performances shown in Figure 6 are contrasted to the performances shown in Figure 7 where segmentation errors have been removed from the analysis. In the second figure, the FAUST system is 87% accurate, the Gabor system is 91% accurate, and the KL system is 94% accurate. Note that accuracies, based on entry fields having the proper number of images segmented, only approximates results free of segmentation errors. Ambiguous cases still exist where the segment module creates both character splits and merges in the same entry field. Therefore, the performances plotted in Figure 7 are not based purely on the recognition of well formed, cleanly segmented character images, but they are close.

Examining both performance figures shows the KL system to be consistently superior to the other two system configurations. However, the KL system's neural network was trained on a larger training set. Another thing to notice is that the performance based on class varies dramatically from digit to digit within each system. For example the KL system is 98% accurate when recognizing 1's and only 89% accurate when recognizing 8's. An interesting observation is that all three systems did equally poorly on recognizing 5's in Figure 6. Figure 7 clearly shows that the poor recognition of 5's in the previous figure was due to a high frequency of segmentation errors. Frequently people print the top horizontal stroke of a 5 disconnected from the rest of the character's body, and this event causes problems for the current segment module.

5.3 Timing analysis

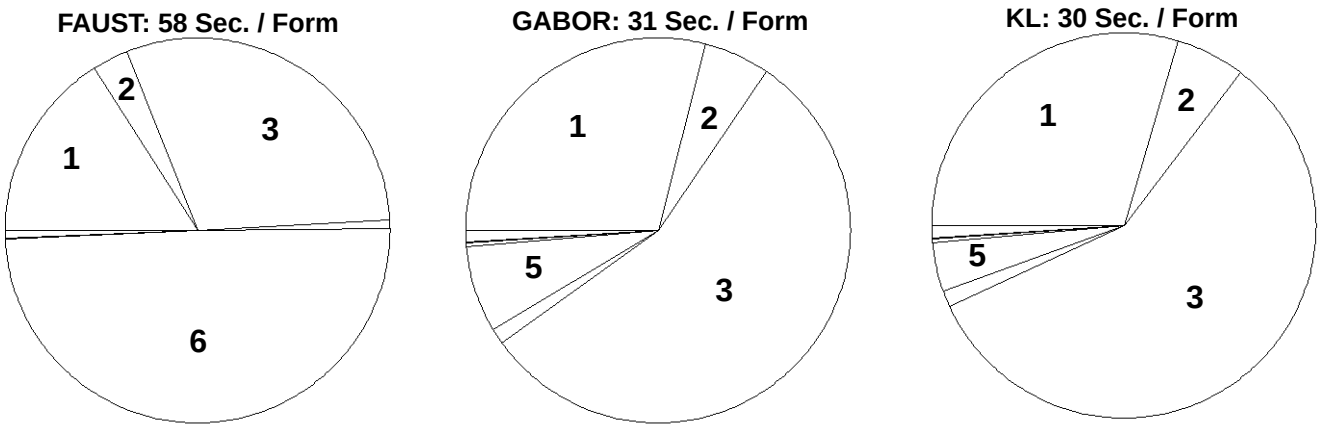

Fig. 8. The time required for the 3 system configurations to process a single form, broken down by system component. The modules are: 1) Load, 2) Isolate, 3) Segment, 4) Normalize, 5) Filter, 6) Recognize, 7) Reject, and 8) Store.

The percentage of time consumed by the individual modules for each of the three system configurations is shown in Figure 8. The FAUST system requires almost one minute to process and recognize the digits from a form like the one displayed in Figure 4. The MLP-based Gabor and KL systems require approximately 30 seconds to process the same form. The pie charts in this figure are separated into wedges representing the time required by each of the system modules listed in Figure 1. The order of the modules from top to bottom in Figure 1 are represented as wedges starting at 9 o'clock and proceeding clockwise in Figure 8.

The three dominant modules in the FAUST system are the load module, requiring 16% of the system time, the segment module, requiring 30% of the system time, and the recognition module, requiring 50% of the system. Ways to optimize the self-organized recognition module are currently being explored. Two dominant modules exist in the Gabor and KL systems. They are the load module, requiring about 30% of the system time, and the segment module, requiring about 55% of the system time. With the use of the MLP network architecture on the massively parallel machine, the recognition module in these two system configurations requires only 0.3% of the entire system time. Given these timing statistics, the MLP networks achieved an effective throughput of 4.3 characters/second. The MLP recognition modules, by themselves, realized a throughput of 1,300 characters/second.

The recognition module throughput is an order of magnitude less than the theoretical throughput reported of 10,100 characters per second. This difference can be accounted for in two different ways. First, the theoretical throughput was achieved by presenting thousands of character feature vectors concurrently resident on the parallel computer to the MLP network in an extremely tight loop. This maximized the use of the parallel processors on the system. In the recognition system processing forms from *NIST Special Database 1*, there are only 130 digits on each image available for recognition at any one time. The load on the parallel system's processors is dramatically reduced, which in turn reduces the effective pipelining of the computer. A second reason for the reduced performance is due to the emphasis on software modularity. A production version of this system, in which algorithms are fixed and interactions between functional components are known, can be optimized by removing the interface overhead supporting modularity and reorganizing the code so that it is entirely *in-line* on the parallel computer.

Three important observations can be made from these timing results. First, by using neural techniques on a massively parallel computer, the time limitations of the recognition module are all but removed. Second, conventional segmentation using traditional image processing techniques poses a large bottleneck on overall system time. This is true even after effectively reducing the segmentation time by a factor of 100 by mapping algorithms from a serial computer onto the massively parallel machine. The final statement Figure 8 makes is to point out the bottleneck imposed when loading the recognition system. With special purpose hardware and VLSI chips capable of 10^{10} connections per second, what image retrieval and transport technology will be able to keep these hardware implementations optimally busy?

6. CONCLUSIONS

A massively parallel character recognition system using neural recognition has been described. The system has been designed to read handprinted information from structured forms. Results from processing the NIST handprint database, *NIST Special Database 1*, were reported. From Figure 1, it was shown that the recognition module is one of many components necessary for implementing an end-to-end system. The integration of these components using a modular software design has been documented and the use of a massively parallel computer has been emphasized. Results from three different system configurations on a 273,000 character testing sample have been studied, the FAUST system, Gabor system, and KL system. An accuracy analysis demonstrated the KL system to be superior to the other two systems. A timing analysis pointed out the bottlenecks inhibiting system throughput. It was shown that determining factors in system throughput are the load and segment modules. Using the parallel computer, the speed of the various system components is approximately proportional to the volume of data handled by each module. There is an 8,000 to 1 data compression from the system's image input to the system's text output. The most likely speed improvements are in the segment module.

7. REFERENCES

1. M. D. Garris, R. A. Wilkinson, and C. L. Wilson, "Methods for Enhancing Neural Network Handwritten Character Recognition," *International Joint Conference on Neural Networks*, Vol. I, pp. 695-700, Seattle, 1991.

2. G. E. Hinton, and C. K. I. Williams, "Adaptive Elastic Models for Character Recognition," *Advances in Neural Information Processing Systems*, R. Lippmann, Vol. IV, to be published, Denver, 1991.

3. 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.

4. 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.

5. 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.

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

7. M. D. Garris, "Design and Collection of a Handwriting Sample Image Database," *Social Science Computing Journal*, Vol. 10, to be published, 1992.

8. R. A. Wilkinson, "Segmenting Text Images with Massively Parallel Machines," SPIE Intelligent Robots and computer vision X, to be published, Boston, 1991.

9. 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.

10. M. D. Garris and C. L. Wilson, "A Neural Approach to Concurrent Character Segmentation and Recognition," to be published in the proceedings of Southcon, Orlando, 1992.

11. H. P. Graf, C. Nohl, and J. Ben, "Image Segmentation with Networks of Variable Scale," *Advances in Neural Information Processing Systems*, R. Lippmann, Vol. IV, to be published, Denver, 1991.

12. J. D. Keeler and D. E. Rumelhart, "Self-Organizing Segmentation and Recognition Neural Network," *Advances in Neural Information Processing Systems*, R. Lippmann, Vol. IV, to be published, Denver, 1991.

13. G. L. Martin, "Centered-Object Integrated Segmentation and Recognition for Visual Character Recognition," *Advances in Neural Information Processing Systems*, R. Lippmann, Vol. IV, to be published, Denver, 1991.

14. M. D. Garris, R. A. Wilkinson, and C. L. Wilson, "*Analysis of a Biologically Motivated Neural Network for Character Recognition*," Proceedings: Analysis of Neural Network Applications, ACM Press, New York, 1991.

15. A. K. Jain, *Fundamentals of Digital Image Processing*, pp. 163-174, Prentice Hall Inc., Englewood Cliffs, 1989.

16. J. G. Daugman, “Complete Discrete 2-D Gabor Transform by Neural Networks for Image Analysis and Compression,” *IEEE Trans. on ASSP*, Vol. ASSP-36, pp. 1169-1179, 1988.

17. 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, to be published, Denver, 1991.

18. C. L. Wilson, “A New Self-Organizing Neural Network Architecture for Parallel Multi-map Pattern Recognition - FAUST,” *Progress in Neural Networks*, Vol. 4, to be published, 1992.

19. A. Rojer and E. Schwatz, “Multi-map Model for Pattern Classification,” *Neural Computation*, 1:104-115, 1989.

20. 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., Volume 1: Foundations, pp. 318-362, MIT Press, Cambridge, 1986.

21. M. F. Moller, “A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning,” Neural Networks, 1991.