

NAT'L INST. OF STAND & TECH R.I.C.



A11105 038668

NIST
PUBLICATIONS

NISTIR 5932

Design, Integration, and Evaluation of Form-Based Handprint and OCR Systems

**Charles L. Wilson
Jon Geist
Michael D. Garris
Rama Chellappa**

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

December 1996

QC
100
.U56
NO.5932
1996

NIST

Design, Integration, and Evaluation of Form-Based Handprint and OCR Systems

**Charles L. Wilson
Jon Geist
Michael D. Garris
Rama Chellappa**

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

December 1996



U.S. DEPARTMENT OF COMMERCE
Michael Kantor, Secretary

TECHNOLOGY ADMINISTRATION
Mary L. Good, Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS
AND TECHNOLOGY
Arati Prabhakar, Director

Design, Integration, and Evaluation of Form-Based Handprint and OCR Systems

C.L. Wilson[†], J. Geist*, M. Garris[†] and R. Chellappa[‡]

Contents

1	Introduction	6
1.1	Human/OCR Interface	8
1.2	Economic Factors	9
1.2.1	The Computer Museum Problem	9
1.2.2	Form-Based OCR	9
1.2.3	Character Versus Phrase-Based Recognition	10
1.2.4	Dictionary-Based Correction	11
1.2.5	Systems Configurations	12
1.3	Discussion	12
2	System Integration	13
2.1	Optimum System Configuration	14
2.2	System Training	15
2.3	Confidence Threshold	15
2.4	Human Correction	16
3	Evaluation of System Performance	18
3.1	Error Rate Versus Rejection Rate	19
3.2	Accuracy Measures	20
3.2.1	Character-Error Rate	20
3.2.2	Field-Error Rate	22
3.2.3	Other Error Measures	23
3.3	Evaluating the Test Sample	23

3.3.1	Absolute Versus Relative Accuracy	23
3.3.2	Importance of Context	24
3.3.3	Extension to OWC and OWR	25
4	System Components & Systems	25
4.1	The NIST Form-based Recognition System	26
4.1.1	The Application	26
4.1.2	System Components	26
4.1.3	Performance Evaluation	37
4.1.4	Accuracies and Error Rates	37
4.1.5	Rejection versus Error Rate	40
4.2	Other Systems	42
4.2.1	The ERIM System	43
4.2.2	The MCC System	48
4.2.3	The KODAK System	49
5	Evaluation Studies	50
5.1	First Census OCR Systems Conference	50
5.2	Evaluation of Two OCR Systems	58
5.3	Second Census OCR Systems Conference	66
6	Summary and Conclusions	70
7	Appendix: Literature Review	72
7.1	Preprocessing	72
7.2	Character Segmentation	73
7.3	Feature Extraction	75
7.4	Recognition	76
7.4.1	Statistical Techniques	76
7.4.2	Image Analysis Techniques	77
7.4.3	Neural Network Techniques	79

List of Figures

1	Comparison of a typical error versus rejection curve with HIGH, MEDIUM, and LOW error-rate requirements.	21
2	Example HSF form from <i>NIST Special Database 19</i>	27
3	Organization of functional components within the NIST system.	28
4	The results of 500 HSF forms registered and ORed together.	30
5	Results of form box removal.	31
6	Line-bands computed from the paragraph image above.	32
7	Segmentor results of merging components together.	33
8	Segmentor results of splitting components apart.	34
9	Results from processing the top paragraph image.	36
10	Rejection versus error rates for digit, upper, and lowercase recognition between HS-FSYS1 and HSFSYS2.	43
11	Segmentation of the word “medical” by the ERIM system.	45
12	Formation of unions from segments by the ERIM system	46
13	Selection of the best combination of unions of segments to represent a word from a dictionary of expected words.	47
14	Error rate versus rejection rate for isolated digits for systems in the First OCR Systems Conference.	52
15	Error rate versus rejection rate for isolated upper-case letters for systems in the First OCR Systems Conference.	53
16	Error rate versus rejection rate for isolated lower-case letters for systems in the First OCR Systems Conference.	54
17	Comparison of human error rate versus rejection rate for isolated digits with that for some system results obtained following the First OCR Systems Conference.	55
18	Comparison of human error rate versus rejection rate for isolated upper case letters with that for some system results obtained following the First OCR Systems Conference.	56
19	Comparison of human error rate versus rejection rate for isolated lower case letters with that for some system results obtained following the First OCR Systems Conference.	57
20	A reduced copy of a FAX of the form used to collect the handprint sample for the comparison of two OCR engines.	60

21	Error rate versus rejection rate for two OCR engines in recognizing a set of 6,422 isolated images, almost all of which contained digits.	61
22	Error rate versus rejection rate for two OCR engines in recognizing a set of 700 images, most of which contained the ten digits, one digit to a box, but about 8 type of non-character information typically found in OCR applications as described in the text.	62
23	Field error rate versus rejection rate for systems in the Second OCR Systems Conference. See [3] for a detailed explanation	68
24	Field distance rate versus rejection rate for systems in the Second OCR Systems Conference. See [3] for a detailed explanation	69

List of Tables

1	HSFSYS1 accuracies and error rates for digit fields across the first part of SD19. Part (A) reports character-level statistics and (B) reports field-level statistics.	38
2	HSFSYS2 accuracies and error rates for digit fields across SD19. Part (A) reports character-level statistics and (B) reports field-level statistics. (<i>†Segmented character images from the writers in this partition were used to train the neural network classifiers.</i>)	39
3	HSFSYS1 accuracy and error rates for uppercase fields across the first part of SD19.	39
4	HSFSYS2 accuracy and error rates for uppercase fields across SD19. (<i>†Segmented character images from the writers in this partition were used to train the neural network classifiers.</i>)	40
5	HSFSYS1 accuracy and error rates for lowercase fields across the first part of SD19.	40
6	HSFSYS2 accuracy and error rates for lowercase fields across SD19. (<i>†Segmented character images from the writers in this partition were used to train the neural network classifiers.</i>)	41
7	HSFSYS1 accuracy and error rates for Preamble fields across SD19.	41
8	HSFSYS2 accuracy and error rates for Preamble fields across SD19.	42

Abstract

This paper addresses several key issues in the design, integration, and evaluation of end-to-end systems for recognizing form-based handprint and cursive handwriting that arise in data entry applications. The applications include recognition of alphanumerals from tax, census, and insurance forms, bank checks, etc. For a system to succeed in these applications, it should be as accurate as possible (recognition accuracies in the upper 90%'s) and at the same time should minimize human intervention to correct that part of the text that machines fail to recognize. An end-to-end system for recognizing handprint and cursive writing is typically made up of the following key modules: preprocessing, character segmentation, feature extraction, recognition and word or phrase correction. These modules are linked to each other in a simple feed-forward manner but proper design of earlier modules depends on later ones. Complex, nonlinear correlations characterize the interdependence of the modules; for example, many of the errors in segmenting cursive writing can be corrected by passing the results of segmentation to a lexicon-based recognizer. Such strong interdependence between the modules argues for an end-to-end system design and evaluation.

The main themes of this paper are the integration of components to design end-to-end systems and their evaluation in the context of human data entry operations. We first discuss design issues such as system training, confidence measures and the role of correction mechanisms employed by humans. Based on these considerations, we present a fairly general recipe for system integration. The successes and failures of many of the existing systems designed for data entry operations can be traced to how well the steps presented in this recipe have been followed. As the components that make up Optical Character Recognition (OCR) systems interact in a highly non-linear fashion, it is very difficult to generalize component-level evaluation metrics to system-level metrics. We argue in favor of system-level evaluation and discuss several useful performance metrics. We present several case studies involving OCR systems that have undergone large-scale evaluations. These include the NIST form-based recognition system, the ERIM, MCC, and KODAK systems, and an evaluation of two commercially available OCR engines.

We hope that the discussions presented here will guide OCR end-to-end system designers and potential users in all aspects of the problem.

†C. L. Wilson and M. Garris are with the Visual Image Processing Group, Division of Information Access and User Interfaces, National Institute of Standards and Technology, Gaithersburg, MD 20899.

* J. Geist is with Sequoyah Technology LLC, Olney, MD 20832.

‡R. Chellappa is with the Department of Electrical Engineering, Center for Automation Research and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742.

1 Introduction

Communication with machines through voice (speech), machine or handprinted text (document image recognition and understanding), and possibly through other means such as gestures and facial expressions is an active research field with numerous applications. In these applications the challenges span technical and economic issues. In this paper we consider a specific case: constrained form-based handprinted and cursive digit and text recognition through OCR technology. In addressing the relevance of OCR technology as a medium of human/machine communication we consider the following issues: the design, integration and evaluation of the capabilities of state-of-the-art OCR technology for form-based recognition of hand printed/cursive writing, the economic and commercial factors that sustain this technology, and the human/machine interface, particularly as applied to data correction operations.

OCR research using electronic and electro-mechanical methods dates back to the 1950's. This work was initially directed toward machine print and was restricted to fixed-format applications. In the late 1980's, decreasing costs for collection and transmission of electronic images and increasing computer power generated renewed interest in OCR for the unrestricted machine print problem and for handprint on forms. Given the forty years of active research on OCR, it is impossible even to enumerate all the relevant OCR technologies for form-based machine, handprinted, and cursive recognition. The key modules that make up an end-to-end system for form-based recognition are preprocessing (form identification, field and line isolation), character segmentation, segment reconstruction, character recognition, and word and phrase construction without human intervention or human correction. For many of these modules, methods based on two primary approaches have been taken: rule-based and image (pattern) recognition. The rule-based approaches have applied existing image analysis techniques in a straightforward manner for field isolation, character segmentation, segment reconstruction, and word and phrase construction. The image recognition approaches originated from the application of classical pattern recognition techniques but are now more or less dominated by neural network (NN) approaches.

Most published research results are for specific components of OCR systems. The NIST OCR Systems Conferences [1], showed that although each component is important, the errors generated by a component can only be corrected in later components by including additional tools designed to work with the original components and to correct specific types of error. As an example, dictionary correction, recognition, and segmentation can be effectively coupled if the segmentor is designed to over-segment, the recognizer is trained to reject small character fragments, and the correction process is capable of merging fragments to reduce errors. Therefore system integration of the modules described above is the most critical, but the least understood topic.

In 1989 the Image Recognition Group at NIST began developing methods for testing large scale OCR systems for handprint on forms for Census and Internal Revenue Service (IRS) applications. In 1990 NIST published Special Database 1 [2] for handprint OCR. In 1992 the University of Nevada at Las Vegas began publishing annual reports [3] on commercial machine-print OCR systems. In May of 1992 [1] and February of 1994 [4] NIST conducted two OCR Systems Conferences which were designed to provide base-line information on OCR technology and specific application-related information on Census industry and occupation data from the 1990 Census. (These Conferences will be referred to collectively as the NIST OCR Systems Conferences, and separately as the First OCR Systems Conference and the Second OCR Systems Conference.) All of these efforts were designed to provide a common framework for the evaluation of research results in machine, handprinted and

cursive OCR that would allow application feasibility to be tested.

The First NIST OCR Systems Conference test [1] was concerned with isolated handprinted characters so that correlation with research results could be made. In the first conference more than half the vendors reported comparable scores with less than 5% error for digit recognition and less than 20% error for lower-case letters. The errors made by most of the low error-rate systems were also highly correlated. In the Second Conference, participants were asked to recognize data from 1990 Census forms. Both microfilm (low quality) and scanned paper (higher quality) images were used. In this conference, the field error rates for the systems ran from 40% to 80%. At 50% rejection, see section 2.2, the error rate for the best system was reduced to 6.3%. Many of the same organizations participated in both conferences, but no correlation was found between isolated character error rates from the First Conference and field error rates from the Second Conference.

The results of the First OCR Systems Conference [1] demonstrated the applicability of a wide variety of pattern recognition methods for solving the isolated character recognition problem. The most accurate of these are comparable to human recognition of isolated characters, and most methods provide estimates of recognition confidence which, with proper rejection, allow results to be produced which are more accurate than can be produced by humans on the part of the data that remains after rejection. As OCR error rates fall, test sample sizes must increase so that a statistically significant number of errors remain for further analysis. In this sense, a perfect OCR result tells the researcher nearly nothing about the recognition process except that the test sample is "easy" for a particular recognition device. The accuracy of method reported in [1] is such that further testing is only of interest on difficult problems where large data samples are available so that adequate error statistics can be computed and failure mechanisms analyzed. The basic machine-printed character recognition problem has therefore been solved with sufficient accuracy so that only large tests, involving 10,000 to 1,000,000 characters, are informative in either basic research or commercial applications.

In the test cited above, strictly rule-based methods were uniformly less accurate than methods based on machine learning or Bayesian statistical models. The general procedure used in the most successful recognition systems was to develop large, rich feature sets and use a large number of examples to train a recognition device. If the training and test sets used are sufficiently similar, these methods produce results comparable to the best previously published recognition rates. The remaining research challenge in this area is dealing with the last few percentage points of error. These cases are hard for humans and usually include both ambiguous and rare forms of characters. From an understanding of the relative strengths and weaknesses of machines and humans described later in this section, the logical way of proceeding with document research is to increase the contextual information available to solve the hard problems. This means converting letters into words and checking the words against some vocabulary using dictionaries. Once words are available, more complex language models can be applied to provide additional context.

An end-to-end system such as a form-based handprint recognition system has correlated modules. The results presented at the two Census Systems Conferences demonstrate the difficulty of deducing end-to-end system performance from component characteristics even when the same organization designed and constructed the components, when interactions between components are complex, highly nonlinear, non-local, non-commutative, and non-additive.

This paper addresses a specific type of handprint recognition problem which meets three constraints. First, the data is on forms, which implies that the data of interest is found in predetermined,

predictable locations. Second, the image quality is sufficient to provide legible images, which implies that the forms have adequate space to enter the required information and that scanning resolution is sufficient to resolve the text. Third, the material to be read from the form has specific, known content, which restricts the lexicon of expected answers. The success or failure of form-based OCR is strongly influenced by the degree to which the application adheres to these constraints. Good form design is essential to economical OCR-based data input but is very application-specific. High-quality scanning and good image quality are necessary to lower processing cost and allow the use of electronic images without extensive cleanup processing. Prior knowledge of field content is important both for recognition correction and for the detection of human errors in placing information on forms. On the otherhand the visual and linguistic content of forms, as well as scanning and input image quality, are outside the scope of this paper.

1.1 Human/OCR Interface

Commercial form-based OCR applications are usually designed to replace data entry operations. This considerably simplifies both the application and the cost of evaluation since OCR is not usually expected to replace all human data entry, but is only required to replace the initial input pass, with human correction taking place in later steps. This allows the application to be designed so that machine and human capabilities complement each other.

It is important to understand that human and machine recognition have complementary strengths and weaknesses. This means that great care must be exercised when comparing their performance, but it also allows them to be combined into a system in such a way that the strengths of one compensate for the weaknesses of the other. When properly designed, such hybrid systems can often outperform purely human or purely machine systems in commercial applications while reducing costs. Based on the published results in connection with the NIST OCR Systems Conferences some of the major conclusions that can be drawn regarding the relative strengths and weaknesses of machines versus human recognition of character can be summarized as follows.

The most important strengths of machine recognition are 1) the ability to perform routine tasks without fatigue, and 2) the ability to produce a result quickly and at the same time to produce a number that characterizes the reliability of that result. A typical reliability number is the confidence, which attempts to approximate the probability that a recognition result is actually correct. The major strength of human recognition is the ability to utilize a hierarchy of ever more powerful skills to recognize a character or word. At the bottom of the hierarchy these skills tend to be relatively fast and quite reflexive. At the other end of the hierarchy, the skills tend to be relatively slow and cognitive.

The major weakness of machine recognition is that it does not work well on the most difficult samples in any given application. On the other hand, the major weaknesses of human recognition are that 1) it quickly fatigues when utilized at its maximum rate, 2) there is a certain time overhead associated with moving up and down the hierarchy of recognition skills, 3) the higher skills in the hierarchy are much slower than the lower skills, and 4) it takes a very long time relative to the average recognition time for a human to assign a useful measure of reliability to his or her recognition results.

Based on these observations, a general set of guidelines is presented in Section 2.4 for building on the complementary strengths of human and machine recognition to produce a hybrid system

that produces low error rates at lower costs than either type of pure system. This model is based on a careful analysis of machine and human recognition results on a large sample, relevant to the application at hand. If this model is followed, it is possible, as demonstrated in systems that are currently functioning in many commercial applications such as remittance processing, to obtain both lower cost and higher accuracy than with a system based only on human entry. However, not every system that has been implemented has been successful. Failures can be traced to the fact that one or more guidelines were not even approximately satisfied, as well as to the failure of one or more of the OCR component modules.

1.2 Economic Factors

1.2.1 The Computer Museum Problem

For many years the decline of paper as the dominant document storage medium has been predicted but has not taken place. In most organizations, computers have been used to automate the creation of paper rather than to replace paper. This seems to be related to both the low initial cost and the anarchical properties of paper storage and to the rapid technological evolution of computer technology. It seems likely that the cost factors will shift to favor computer storage over paper even in areas where this is not now the case. The obsolescence created by rapid technological change of computer technology poses a much less tractable problem.

The problem of maintaining electronic archives through many generations of computer systems is sometimes referred to as the Computer Museum Problem. Simply stated, unless data is kept on paper or microfilm, a computer museum must be maintained to read old data. Try to read a paper tape or an old 107K eight inch hard-sectored floppy disk today! This means that unless a user is prepared to migrate all data forward to the latest "standard" media, all archives must be paper-based or a computer museum must be set up. The Computer Museum is charged with maintaining one or more working copies of each computer system, both hardware and software, required to read each kind of data in the archive. The only other alternative to maintaining the museum is to stay in the document conversion business for an indefinite period!

More rapid improvement in storage and processor technology increases the problem by lowering the expected life of each generation of system. The only present plan of attack is to try to make conversion so low-cost (and automatic) that this problem doesn't matter. This will require accuracy such that no human intervention is needed. This does not mean that the conversion process must be perfect but only that it must be as good as human performance. We therefore expect OCR to remain viable for a lot longer than predicted by the oracles of the paperless office.

1.2.2 Form-Based OCR

For form-based OCR to be economically successful several requirements must be met. 1) The form design must be compatible with both OCR and human data entry. 2) The work flow in the data entry process must allow a gradual conversion from all-hand data entry to OCR. 3) The volume of work must be sufficiently large to cover the substantial capital expenditure required to install high-speed paper-handling equipment, large image databases, and high-speed OCR hardware. The large volume requirement reduces the risk in applications where a large number of transactions are

processed by a single set of common or centralized system, because setup cost can be amortized over a larger volume of work.

Typical applications which meet these requirements are credit card forms, tax forms, census forms, insurance forms, and some types of checks. In the credit card application, the form is relatively simple, redesign to improve segmentation is not too costly, and volume is high. The volume of forms in both the Census and IRS applications is high, form redesign is possible, and the periodic nature of the data collection process makes running a prototype test between the peaks of the flow practical. For insurance applications, the forms are often complex and the data flow is continuous rather than periodic. This makes inserting new technology into the work-flow much more difficult and increases the economic risk involved in the introduction of the technology. Checks have a more complex profile. Commercial checks are usually machine-printed with standard backgrounds but do not adhere to any standard field locations. Personal checks are usually in a standard field format but have complex backgrounds. The data volume exceeds 10^9 items per year but maximum clearing time is regulated by Federal Reserve regulations. This makes the application of form processing technology to checks much more complex than to more centralized applications.

In large-scale commercial applications of OCR, the goal is to provide lower-cost data entry with a minimum of change in existing work-flow. An analysis of the cost of data entry shows that the cost of using OCR for data entry is critically dependent on the cost of correcting the part of the text that machines fail to recognize. When dealing with hand-written or poor-quality machine-printed material, there is always some level of illegibility in the sample which is too difficult for humans to read in isolated form. Machine failures on this material are expected in the data entry process as they have been with human data entry. Since some failures are expected in the OCR, the material must be checked by humans. The cost of achieving a specified level of accuracy depends on how efficiently the material that has errors can be separated from all of the material processed. An OCR system that requires 100% human checking and correction is economically valueless. This means that commercial use of contextual material must be directed toward increasing accuracy and reducing the amount of material that needs to be checked.

1.2.3 Character Versus Phrase-Based Recognition

When considering form-based OCR, it is important to distinguish between forms that contain a separate box for each character in a word or phrase answer and forms that contain only one box for an entire answer. The former will be called character-based forms and the latter will be called phrase-based forms.

The distinction between character-based and phrase-based forms is important because current OCR systems are much less accurate on phrase-based forms than on character-based forms. The reason is that segmenting handprinted words or phrases into isolated letters is very difficult at the current state of the art. Instead of producing a string of handprinted letters, current segmentation algorithms often produce letter fragments, combinations consisting of a letter and a fragment of another letter, and combinations of two or more letters.

OCR systems designed for phrase-based forms use dictionaries of expected words or phrases to compensate for segmentation errors. A continuum of different dictionary-based approaches can be distinguished ranging from what could be called Optical Word Correction (OWC) through what could be called Optical Word Recognition (OWR).

A pure OWC system is characterized by 1) a segmentation algorithm that is optimized to maximize the fraction of isolated letters occurring among the segments, 2) a recognition algorithm that outputs a string of letters, and 3) a powerful spell-check algorithm that attempts to correct misspellings caused by the contamination of the string with both letter fragments and combinations of letters that were recognized as single letters.

A pure OWR system is characterized by 1) a segmentation algorithm that is optimized to produce no combinations of letters, but only isolated letters and letter fragments, 2) a fragment merging algorithm that combines letters and letter fragments into candidate letters, 3) a recognition algorithm that attempts to select a subset of the candidate letters in such a way that all fragments occur in at least one letter, but no fragment occurs in more than one letter.

It is uncommon to distinguish explicitly between OCR of character-based forms and either OWC or OWR of phrase-based forms, or to distinguish explicitly between OWC and OWR, and we will continue this tradition by using OCR to refer to all types of systems except where the distinctions among these systems are being discussed.

1.2.4 Dictionary-Based Correction

For both OWC and OWR, words or phrases from a dictionary of expected answers are compared with the recognition results. The dictionary is created by selection of answers from a representative sample of the intended application. Note that there is both an optimum size and an optimum content for the dictionary. Ideally, the dictionary should contain all of the answers, and only those answers, that will occur in some given application. This, however, is not possible for most applications.

Dictionary coverage describes the fraction of the answers obtained in a given application that actually occur in the dictionary of expected answers. As possible answers are added to the dictionary, the coverage over any given application either increases or stays the same. If this were the only consideration, then larger dictionaries would be better than smaller dictionaries under all circumstances. Unfortunately, as possible answers are added to a dictionary, coverage is not the only thing that increases. The probability that incorrect dictionary entries will be selected as answers also increases. This is called dictionary confusion.

An efficient way to build a dictionary is to start with the most commonly occurring answers from a sample of the intended application and to use it to construct words for another representative sample of the same application. As the size of the dictionary is increased, the error rate will initially decrease because the dictionary coverage will grow rapidly while the dictionary confusion will grow at a more moderate rate. As the dictionary grows larger through the addition of fewer and fewer common answers, the coverage will grow more and more slowly, but confusion will grow. Eventually, the rate of growth of coverage will no longer compensate for the rate of growth of confusion, and the error rate will stop decreasing and start increasing. This yields an estimate of the optimum dictionary size for the particular application.

1.2.5 Systems Configurations

The most common large-scale application of form-based OCR technology is data entry from forms such as IRS tax forms, Census forms, or insurance forms. These applications are usually custom-designed systems. Document input is usually through high-speed scanners which are fed by high-speed paper-handling equipment. The scanners are part of a network environment which includes large-scale storage for images and special high-speed servers to perform OCR. Imaging workstations are used for data correction. The general structure of these systems is similar whether tax forms, Census forms, or bank checks are being processed. It is expected that data throughput will be in the page per second range for each input scanner and total page throughputs of 25,000 pages per operating shift are possible. A relatively small system of this type can process 5,000,000 pages per year and large systems can process 100,000,000 pages per year. Total system cost scales with size and begins in the \$1,000,000 range.

Commercial systems usually are directed toward a specific type of application. Most systems do either machine print or handprint and are designed to run in desktop applications or in large-scale batch data processing applications. The market for desktop applications has been oriented toward providing an input device for word processing software which uses scanned images of machine-printed documents. These systems are not assumed to be dedicated document input stations but are assumed to be general-purpose machines where only a few pages are input each day. The desktop system is expected to have a high-end word processor, a scanner with scanning software, OCR software, and sufficient memory and processor power to run all three applications. This takes a workstation or high-end personal computer. A relatively small fraction of all desktop systems meet these requirements. Additional system costs include the price of the scanner and word processing and OCR software.

1.3 Discussion

Although form-based OCR systems capable of solving many economically important problems can be constructed with state-of-the-art technology, NO general off-the-shelf solutions exist for large-scale, centralized form processing applications. The economically useful systems are successful because they are constructed from modules customized to capitalize on ALL constraints afforded by a particular application. The more constraints available and incorporated, the higher the probability of success. The more complex an application, the fewer constraints are available. the lower the probability of success.

The only meaningful way to compare the performance of two different OCR modules in these systems is by comparing the overall system performance for the system in a particular application using one module with that of the same system using the other module in the same application. Much literature exists at the module level, but little at the systems level. Therefore, this paper treats OCR carried out at NIST from the systems point of view, and provides a summary of the other literature with references to other reviews where possible.

The organization of this paper is as follows: We begin with a detailed discussion of issues relevant to system integration with an emphasis on data entry applications. Critical issues such as the appropriate error measures and confidence thresholds to be used as well as the role of human correction mechanisms in post processing OCR results are addressed in detail. Based on these

discussions a general recipe is presented for integrating the different modules that make up an OCR system. The implications of not following some of the steps in the recipe are unexpectedly large error or rejection rates when the poorly integrated OCR system is applied to real data. As mentioned earlier, the components that make up the OCR system interact in complex, non-linear ways. Therefore it is nearly impossible to generate system-level performance characterization from component-level performances. Section 3 presents several system-level accuracy measures at the character and field levels. Important considerations for evaluation of the test sample and the role of context in OCR system performance are discussed.

A paper that emphasizes system integration and evaluation would not be complete if supporting evidence were not presented based on OCR systems that have gone through large-scale evaluations. We present brief descriptions of several end-to-end systems designed by NIST, ERIM, MCC, and KODAK. This is followed by an analysis of the evaluations conducted as part of the OCR Conferences and a more detailed evaluation of two commercially available OCR engines. Finally, a summary and conclusions are presented in Section 6.

The designers of commercially available OCR systems have benefitted from over thirty years of basic and applied research done in Universities. Aside from the CEDAR group at SUNY, Buffalo, most of the research performed in academia has focused on a subset of the modules of the OCR system. For the sake of completeness, we present a brief survey of relevant work, largely performed at United States Universities in the Appendix. Due to space limitations we could not cover all the published work.

2 System Integration

Depending on the application, system integration can play a crucial role in determining failure or success. Nevertheless, at the current state of the art, it would probably be a serious mistake to think of an OCR application as nothing more than stringing a set of modules together. This section will explore some of the system integration issues, especially the role of human correction of machine recognition results in difficult applications.

The major system integration issues include the following:

1. What is the optimum system configuration?
2. What error measures should be used?
3. What type of system training should be used?
4. What type of confidence threshold selection should be used?
5. Should human correction be used?
6. How should human correction mechanisms be exploited for system integration?

All of these issues, except for 2) which is addressed in section 4, are addressed here.

2.1 Optimum System Configuration

Since the authors know of no successful commercial systems based on OWC or OWR, it is not clear that forms with a single box for each multiple-character answer are a viable option at the current state of the art. Nevertheless, since this situation may soon change, the trade-off between using forms with a box for each character and forms with a box for each multi-character answer is the first topic that will be discussed here.

Many potential users of commercial OCR systems reject OCR once they discover that they will have to redesign their forms to provide a box for every character in a response. Usually, they assume that their potential users will not agree to fill out a form with these constraints or will fill it out less carefully or less accurately. This is probably more true for forms that request a great deal of information, particularly if they are currently filled out with cursive rather than handprint. However, before rejecting OCR, a potential user might want to attempt to redesign the form, and run a small test on potential users of the form to determine whether or not the desired information can be obtained from a form suitable for OCR.

Even with OWC or OWR, assuming that it becomes commercially viable in the near future, some form design will probably be needed in order to obtain sufficiently accurate performance. There is no question that form design can have a major impact on accuracy for all of these types of systems. Also, OWC and OWR systems optimized for any given application will probably cost more and run more slowly than an OCR system optimized for the same application but including isolated character boxes.

The type of data to be extracted from the forms will also have a bearing on the trade-offs among OCR, OWC, and OWR. OCR will probably retain a cost advantage wherever a simple transcription of the strings of characters on a form is the desired end product. On the other hand, if further human effort is needed to convert the transcriptions into a small number of numeric codes, OWC or OWR may have the edge. This will depend on the complexity of the rules that map the transcribed answers into the numeric codes, and on how many codes there are. Small numbers of codes and complex rules favor OWC and OWR. For instance, the US Bureau of the Census converts the three industry and occupation answers on the long Census form into two codes, an industry code and an occupation code. For each industry or occupation code, a large number of possible multi-character answers are entered on the Census forms. It is possible, but to the authors knowledge has not yet been demonstrated, that OWC or OWR can provide a significant advantage in a case like this.

After deciding among OCR, OWC, and OWR, there remain other optimization trade-off issues. For any given subsystem, there will be a trade-off between speed and accuracy. If a subsystem does not limit overall system accuracy, it should be possible to replace it with a faster but less accurate subsystem without reducing overall system accuracy. This is true of any system, but it is particularly important for OCR systems because the difficulty of practical applications covers a very large range, from typed characters having a fixed and known inter-character distance, to a full word or phrase of handprinted or handwritten characters in a single answer box. Unfortunately, it is often difficult to determine appropriate accuracy measures to study these trade-offs. Even though the question of what accuracy measures should be used is the next systems issue that needs to be addressed, its discussion will be postponed until Section 4.

2.2 System Training

Most OCR systems have variable parameters that can be adjusted to produce better or worse performance in any given application. In the simplest systems, only a few values of a small number of parameters are allowed, and the adjustments can usually be made by hand. In the most complex systems, a very large number of parameters that must be adjusted by some sort of automated or machine-learning algorithm are involved.

The adjustment of system parameters to optimize the performance of an OCR system for any given application, or for some expected mix of applications, is called system training. It can be done by the vendor before delivering the system, or by the user during installation, or application-independent generic settings may be used.

The simplest systems provide useful results only for the simplest applications, but are generally not very sensitive to the details of these applications. Consequently, there is very little variation in performance from one of these applications to another. Thus training can be carried out on a single generic data set for almost any of these simple applications. However, as already mentioned, these systems are generally not accurate enough for the more difficult recognition tasks.

At the other end of the complexity spectrum are neural networks and other complex systems. In order to provide peak performance, these often require a significant amount of automated training on data that is very similar to the intended application—that is, a small, but statistically representative sample of the actual application data.

In cases where application-specific training is needed, failure to use a truly representative sample can cause unexpectedly large error rates or unexpectedly large rejection rates when the system is applied to real data. The net result can be failure to achieve the desired accuracy or cost goals on the application data.

2.3 Confidence Threshold

Most OCR systems assign a confidence value to each character, number, word, or phrase produced by the system. Confidence values, which are defined to be real numbers in the range from 0 to 1, are used by OCR systems to combine machine recognition with human keying of unreliable recognition results.

The confidence should correlate strongly with the probability that the system output is correct according to the error measure being used. The more strongly the confidence correlates with the probability, the more useful it will be. If the confidence does correlate with the probability, then it is always possible to transform the former into the latter with some residual error. Some OWC and OWR systems make use of this property to calculate system-level confidences from module-level confidences. Other systems use heuristic rules instead.

The confidences associated with the recognition results from an OCR system are used to order the results from least reliable to most reliable. This allows some fraction of the less reliable results to be rejected for human keying. The confidence threshold is the confidence value that is used as a reliability cut-off for rejection. Any recognition result with a system-level confidence less than the confidence threshold is rejected, and the corresponding image is sent to a human for keying before

being included in the system output. Any recognition result with a system-level confidence greater than the confidence threshold is accepted and included in the system output.

The same trade-offs associated with system training are associated with selection of the confidence threshold. Simple systems can often use a generic value of the confidence threshold just as they can be trained on generic data. On the other hand, complex systems and highly optimized systems will require selection of application-specific confidence thresholds just as they require application-specific training.

Ideally, the selection process is carried out on a small but statistically representative sample of the actual application data. However, the same sample cannot be used for confidence selection as for system training without risk. With insufficient training sample size, the error rate and the confidence threshold for the training sample will be much lower than the error rate and confidence threshold for the threshold selection sample. As the size of the training sample is increased, the error rate for the training sample increases towards an asymptotic value, and the error rate for the threshold selection sample decreases toward the same asymptotic value, again assuming that both samples are statistically representative of the same application.

Thus, if the training sample is not large enough to produce asymptotic behavior, a confidence threshold that gives a higher than expected error rate on the real application data will be obtained from the selection process. It is generally easier to use one sample for training and another for confidence selection rather than to study the asymptotic behavior of the confidence threshold as the size of the training sample is increased.

2.4 Human Correction

In some applications, OCR systems are accurate enough that their output can be used without rejecting any recognition results. More often than not, the error rate measured over the entire application is too large to be acceptable. In these cases, it is often possible to obtain a satisfactory error rate by replacing a small but unreliable fraction of the total machine recognitions with key entry. In fact, this spells the difference between failure and success in many applications.

If key entry is required to obtain the desired level of accuracy, the cost of human keying of the rejected fraction of the application must be added to the cost of running the entire application through the OCR system in order to determine the overall cost. If this total cost exceeds the cost of key entry for the entire application, there is no cost advantage to be gained from OCR. However, there still may be accuracy advantages. OCR combined with key entry is usually cost-effective when the obtainable rejection rate is small.

A fairly general recipe for system integration of an OCR application is given below based on the points raised above:

1. Obtain access to a prototype of a candidate OCR system for the application of interest.
2. Have humans key two different, small, and statistically representative samples of the application. One is called the training sample, the other the threshold-selection sample.
3. Select or develop an answer-matching algorithm that properly distinguishes correct from incorrect answers from the point of view of the application of interest.

4. Train the system on the training sample: Machine recognition is applied to all of the characters or phrases in the training sample, the results are compared with the human-keyed results using the answer-matching algorithm, and the system parameters are adjusted to minimize the recognition errors.
5. Order the recognition results for the threshold-selection sample according to their associated confidence values.
6. Calculate the error rate versus rejection rate for the threshold-selection sample: Machine recognition is applied to all of the characters or phrases in the threshold-selection sample, the results are compared with the human-keyed results using the answer matching algorithm, any differences considered significant by the algorithm are flagged, the flagged answers produced by the OCR system are compared with the answers entered on the original forms by the respondent, and scored as either correct or incorrect, and a table of error rate versus rejection rate and confidence value is prepared.
7. Select acceptable, application-dependent error and rejection rates for the machine-produced answers in the original data set. Assuming that a line in the table of error rates and rejection rates versus confidence values has an error rate that is less than or equal to the chosen error rate combined with a rejection rate that is less than or equal to the chosen rejection rate, then the confidence value on that same line may be selected as the confidence threshold, and step 8) below may be carried out.
8. Use the run times and costs for machine recognition of the training and threshold selection samples to estimate the run time and cost for the entire application. If these are favorable in comparison with key entry, then proceed with step 9) below.
9. Run the entire applications through the OCR system.
10. Accept all of the recognition results that have a confidence greater than the confidence threshold.
11. Reject all of the recognition results that have a confidence less than the confidence threshold.
12. Have humans key the rejected results.
13. Combine the accepted recognition results and the answers keyed by humans to form the final answer set for the application.
14. During normal operation periodically recalibrate the system using steps 6 and 7 to insure the best match between system parameters and current work.

If everything can be done as described above, it is possible, as demonstrated in systems that are currently functioning in many commercial applications such as remittance processing, to simultaneously obtain lower costs and higher accuracy than with a system based only on human entry. However, not every system that has been implemented has been successful.

Many failures have been caused by a failure to assure that each step in the above model is approximately satisfied. Failures have also been caused by use of rather different models as the basis for implementation. The major stumbling blocks in this model are Steps 1), 2), 3) and 7).

As far as step 1) is concerned, some potential users of OCR think that they can reduce the implementation cost by avoiding prototyping the system. Of course, it may not cost much more to implement a very small system than to prototype it, but this is definitely not the case for large systems. At the current state of the art, it is probably a major mistake to put a lot of money into a system that has not been proven with a prototype.

As far as step 2) is concerned, it is not really necessary to draw the training and threshold-selection samples from the actual application. Any representative samples will do, but they must be representative of the application from the point of view of both the OCR system and the answer-matching algorithm.

As mentioned above, failure to use truly representative and different samples can cause unexpectedly large error rates or unexpectedly large rejection rates when the OCR system is applied to real data. The net result can be failure to achieve the desired accuracy or cost goals when running the system on the real application data, although predicting success in the threshold-selection run.

As far as step 3) is concerned, much of Section 4 of this paper is concerned with the selection of the answer-matching algorithm, so it will not be considered further here. This leaves step 7) as the last major problem area. If acceptable error and rejection rates cannot be found on the same line in the table of error rates versus rejections rates and confidences values, then there are only a few alternatives:

1. Use a more accurate or sophisticated OCR system.
2. Improve the training of the existing system to produce more accurate results. A larger training sample may be needed, or a training sample with more carefully checked and corrected human-keyed results may be needed, depending on the particular system being used.
3. Relax the requirements on the error rate or the rejection rate. If an overly conservative answer-matching algorithm is being used, it may be possible to modify the algorithm in a less conservative way to obtain satisfactory error rates.
4. Reject OCR as a viable option at this time for the application of interest.

Of course, none of these things can be done until suitable accuracy measures have been chosen for the application of interest. This and related topics are the subject of the next section.

3 Evaluation of System Performance

It is useful to distinguish between system-level and module-level evaluation of OCR systems. System-level evaluation can be used to choose among candidate systems for some particular application or to determine whether or not a particular system can produce the required combination of cost and accuracy for some particular application. System-level evaluation can also be used to observe the effect of module-level changes on overall system performance.

Module-level evaluation is sometimes used during attempts to improve the performances of individual modules. However, this requires a module-level accuracy measure that is strongly correlated

with the system-level accuracy measure appropriate to the intended application. Except at the earliest stages of system development (when the system-level error measures may not be available), it is often more difficult to develop suitable module-level error measures than to run the whole system and observe the changes in a system-level accuracy measure when changes are made in the individual modules.

When suitable module-level accuracy measures are available, it is possible in principle to calculate the system-level accuracy measure from the module-level accuracy measures, but in practice this is very difficult. The problem is that the interactions among the modules are non-linear, non-local, and non-additive. This effect is more apparent with OWC and OWR systems than with OCR systems.

For instance, it has already been mentioned in the introduction that no correlation was apparent between the performance of the OCR systems in the First Census OCR Conference and the OWC and OWR systems in the Second Census OCR Conference, even though the same character recognition subsystems were used in both conferences by many participants. In fact, the relatively small differences in performance among the different character recognition modules that were apparent in the first conference were washed out by the relatively larger differences in the ability of the integrated systems to process the full words and phrases entered in a single answer box.

As a general rule, the effect of any given type of error on the next module in the processing chain is very non-linear. For example, small misalignments in field isolation usually cause no errors in later modules. At some misalignment threshold, which is difficult to quantify because misalignment is not a one-parameter phenomenon, recognition errors suddenly appear. As the misalignment is further increased, the recognition errors rapidly multiply. The same is true for segmentation errors.

Moreover, misalignment errors and segmentation errors are not additive. Some recognition errors are caused by segmentation errors that arise in the segmentation module, while others are caused by segmentation errors that result from misalignment errors. Consider a segmentation error that was caused by an alignment error in the field-isolation module, but that would have occurred in the segmentation module even with perfect field isolation. This error occurs only once in the final recognition result, yet it is counted twice in any module-level algorithms that treat misalignment and segmentation errors as additive.

In light of the above, the remainder of this section deals only with system-level error measures. Even with this restriction, there remain important problems. The first is how to choose an accuracy measure that is suitable for a particular application. The second is how to obtain a set of test data for which the correct results are known so that the OCR system results can be compared with the correct results for scoring. These topics will now be discussed.

3.1 Error Rate Versus Rejection Rate

The measurement of error rate versus rejection rate was introduced in the preceding section. It will be discussed in more detail here. This quantity can be calculated only for systems that assign a measure of recognition reliability, such as the confidence defined in the preceding section, to each output result, whether it be a character, a word, or a phrase.

For a system that provides confidence values, the error rate versus rejection rate for a sample of forms is calculated as follows. The system output results for the sample are scored as correct or

incorrect according to an appropriate accuracy measure. The scored results are then sorted into a list according to the magnitude of the confidences, starting with the lowest confidence. Finally, the fraction of the results that were scored correct is tabulated as a function of the size of the list as the list is truncated starting from the low-confidence end of the list.

The resulting tabulation is the error rate versus rejection rate for the recognition results, starting at a rejection rate of zero and ending at a rejection rate near 100%. Figure 1 shows a generic graph of error rate versus rejection rate. Three error rates, HIGH, MEDIUM, and LOW, are shown on the graph. The error rates for all values of the rejection rate fall below the HIGH value and above the LOW value, while the error rate curve crosses the MEDIUM value at some value between 0% and 100% rejection rate. The confidence threshold to obtain the MEDIUM error rate is defined as the confidence value that corresponds to the rejection rate where the error rate curve crosses the MEDIUM value.

If the HIGH value of the error rate shown in Fig. 1 is good enough for the particular application of which the sample is representative, then the OCR system can be used for that application without rejecting any results for human correction. If no error rate above the LOW value in Fig. 1 is good enough, then the system cannot be used for the application. If the MEDIUM error rate is good enough, then the system can be used provided that only those recognition results having confidences greater than the confidence threshold are used as answers. The rest must be keyed by humans.

The slope of the error rate as a function of rejection rate is a measure of the efficiency of the rejection process. A flat error-rate versus rejection-rate curve indicates that the answers are being rejected at random. This means that there is no correlation between the confidence assigned to a result and the probability that the result is actually correct.

3.2 Accuracy Measures

3.2.1 Character-Error Rate

The OCR accuracy measure most commonly cited is the character-error rate. It is defined as the ratio of the number of characters correctly recognized to the number accepted. The character-error rate can be either a system-level error measure or a recognition-module accuracy measure. When used at the module level, correctly isolated images of characters are the input to the module, and character hypotheses are the output. When used at the system level, the output of the segmentation module, which may include character fragments as well as multiple characters, is the input to the recognition module.

From the system point of view, the character-level error rate is fairly well suited for evaluating OCR system performance when segmentation is not an accuracy-limiting step. This is just a reflection of the fact that OCR systems work well in these applications.

The results of the First Census Systems Conference were noteworthy in that most systems had character-error versus rejection rate curves that could be fit to a simple model. Furthermore, the parameters determining the shape of the model curve varied much less about their average values than did the error rate at zero rejection rate. Reference [5] showed that this unexpected result was explained by the fact that most of the systems were behaving as Bayesian classifiers that

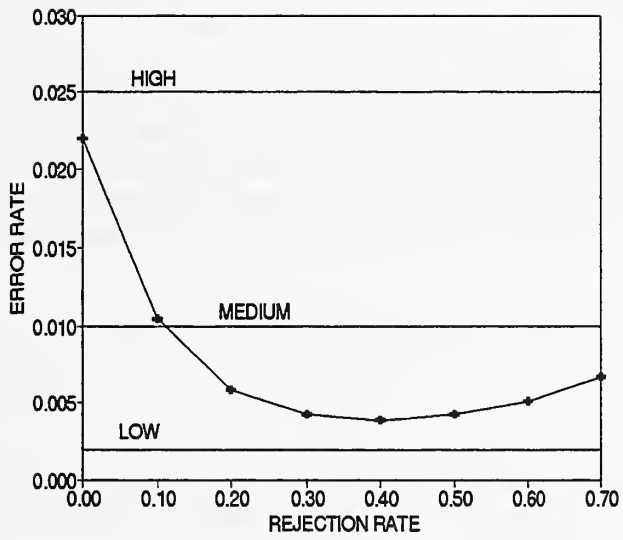


Figure 1: Comparison of a typical error versus rejection curve with HIGH, MEDIUM, and LOW error-rate requirements.

were choosing between only two possibilities. Thus for well-segmented characters, the recognition of hand-printed characters (even with fairly difficult handprint samples) usually involves a choice between no more than two plausible alternatives.

Unfortunately, even with negligible segmentation errors, the character-error rate may not adequately distinguish small differences in performance among OCR systems for some particular application. Suppose two systems have the same character-error rate, but the first system has a much greater probability of producing errors in adjacent characters compared to the second system. In this case, the first system would tend to concentrate its errors in a smaller number of answer hypotheses than the second system, but these hypotheses would be more difficult to decipher because they would contain a larger fraction of multiple errors.

Either of these systems might be more suitable for a given application, depending on the details of the application. Many applications can tolerate single errors in occasional words because the meanings of the words are still easily determined. On the other hand, other applications may require the errors to be concentrated in a small number of words, which can then be rejected on the basis that they do not match any entry in a dictionary of expected answers very well. This illustrates how important it is to choose an error measure that accurately reflects how the results will be used.

In applications where segmentation is a major source of error, character deletions and insertions, as well as substitutions, must be accounted for at the system level, but not at the module level. Therefore, even though the character-level error rate can still be useful at the recognition-module level, it cannot be used at the system level. At the current state of the art, such applications require OWC and OWR systems, and they also require a more sophisticated accuracy measure than the character-level error rate.

3.2.2 Field-Error Rate

The field-error rate is another common error measure. It is defined as the ratio of the number of answer fields with at least one error (character substitution, insertion, or deletion) to the number of answer fields accepted. The field-error rate cannot be used to determine the error rate versus the rejection rate unless a field confidence is available. With an OCR system, the field confidence can be calculated from the product of the individual character confidences, provided that the latter have been transformed to an approximation to the probabilities that the individual character hypotheses are correct. Without this transformation, the resulting confidences will correlate very poorly with the probabilities that the field hypotheses was correct.

There are two problems with the field-error rate. First, it makes no distinction between a ten-letter word with one error and a ten-letter word with ten errors. Second, longer answers have lower probabilities of having zero field-error rate (being completely correct). This does not usually reflect the usefulness of the answer. The number of errors that can be tolerated in an answer before it becomes difficult to decipher increases with the number of characters in the answer.

3.2.3 Other Error Measures

Commercially successful OCR is a fairly recent phenomenon. As a result, the development and testing of more sophisticated error measures than the character-error and field-error rates is rather immature. Nevertheless, some general conclusions can be drawn.

Over the years, the U.S. Bureau of the Census developed a very simple error measure for distinguishing between usable and useless key-entry results for alphabetic data hand-entered on Census forms. A small fraction of the returns are keyed by two different keyers. For each answer field, the characters having descenders (in lower case), the characters having ascenders, and the remainder of the character counts are separately summed for each answer field result. The sum of the absolute value of the differences between the sums for each category are then calculated. No differences are allowed for words having less than six letters, and increasing numbers of differences are allowed with increasing word size. This procedure has apparently worked well for comparing key-entry results, but it has not been shown to be suitable for comparing OCR results with other OCR results or with key-entry results.

The Second Census Systems Conference used an accuracy measure called the field-distance rate. This was defined as the ratio of the sum of the numbers of insertions, deletions, and substitutions required to convert the answer hypothesis to the correct answer after alignment by an algorithm that minimized a particular function of three variables; the number of insertions, the number of deletions, and the number of substitutions. In the case of no insertion or deletion errors, the field-distance rate was identical to the character-error rate defined earlier, and it has a number of desirable properties as an error measure for words and phrases. Nevertheless, it produces counter-intuitive results in certain (apparently rare) cases.

A more detailed discussion of the pros and cons of the field-error rate is given in [4]. That reference also proposes a generalized field-distance rate that can, in principle at least, be optimized for any given application. Unfortunately, implementation of the generalized field-distance rate appears to be very complex.

3.3 Evaluating the Test Sample

3.3.1 Absolute Versus Relative Accuracy

So far it has been assumed that there is a unique correct answer against which any given answer hypothesis can be compared to determine correctness. This section addresses the extent to which this is actually the case, and how the correct answers can be determined.

Assume that a small but representative sample of handprint answers has been drawn from some particular application to be used as a training sample, a threshold-selection sample, or some other type of test sample. First consider the case of OCR of characters printed one character per character box, as opposed to OWC or OWR of multiple characters in a single answer box. In this case, it is necessary only to determine the correct character to assign to each character box in the sample.

We must next decide whether it is absolute error or relative error that is of importance in the particular application. It is almost always the latter, which is a good thing because the former is very difficult to measure. The former requires distinguishing between printing and spelling errors

made by the person filling out a form and recognition errors made by the OCR system reading the form.

Consider the case where the person filling out a form interchanges the letters “o” and “e” in the word “people”. How should a system (including human key entry) be scored that produces the answer hypothesis “poeple” and how should a system be scored that produces the answer hypothesis “people”? What if the “o” and “e” have been printed over as if to correct the misspelling, or to introduce it? What if one of the two letters is completely illegible, and the other is clearly legible, but in the wrong location? What if neither letter is legible? What if a letter in some word is clearly not the letter that it is supposed to be, but it is not clear what letter it actually is supposed to be? These can be the subject of much debate if absolute error is the goal, but they are only minor problems if relative error is the goal.

If an absolute error measure is the goal, it is necessary to assign a correct classification to every character. If relative error is the goal, it is necessary to assign a correct classification only to those characters for which at least two systems provide different character hypotheses and allow “ambiguous” to be a permissible class. If all systems assign the same incorrect classification to some particular character, they will all be scored correct, which may be wrong in an absolute sense, but this will not affect the relative ranking of the systems on the test sample.

References [1] and [6] describe machine-assisted techniques for assigning correct classifications to most characters, and rejecting a few characters as too ambiguous to classify. This technique can be used if absolute error rates are desired. Note that it requires every character to be viewed and classified by at least one person, and every character and classification to be viewed and accepted or rejected by at least one different person. On the other hand, 1024 or more characters can be viewed at one time in a context that makes incorrect classifications easy to detect, so this is a fast process.

Section 5.2 in this report describes a procedure suitable for determining relative error rates. The character hypotheses produced by two or more different systems are compared character by character. Those characters for which the two systems disagree (produce different character hypotheses) are marked and presented to at least one human for resolution. There are some trade-offs to be considered when presenting the flagged handprinted characters for human classification. Ideally, a large number of humans would independently classify each flagged character, and the character would be rejected from the test or assigned a correct classification based on the results of the human classifications. In practice, this is quite complex, and classification by a single human is usually sufficient.

3.3.2 Importance of Context

In the procedure of Section 5.2, it is important that the flagged characters be presented to the human classifier with no clue as to the character hypotheses produced by the different systems. It is also important that they be presented along with sufficient context to allow the human to surmise what character was intended. Showing the entire answer field to the human is one way to do this. An alternate technique is described in [6].

Without such context information, a human classifier will reject most of the ambiguous characters. This may seem like a good idea at first glance, but actually it is not. It is just these characters,

ambiguous but still readable in context, that differentiate among the performances of humans and different OCR systems. Just as the characters that are so difficult that no OCR system gets them correct do not distinguish among competing systems, neither do the characters that are so easy that all systems get them correct. It is how well an OCR system performs on the characters that are ambiguous out of context, but can be read by humans in context, that determine whether or not the system can produce human-level error rates.

3.3.3 Extension to OWC and OWR

The above discussion can be applied to evaluating a test sample for OWC and OWR with minor modifications, all of which are obvious with one exception. In most applications more than one word or phrase will be as acceptable as another, and blindly accepting only one of a number of essentially equivalent answer hypotheses as correct may prevent the acceptance of OWC or OWR in cases where it could save a substantial amount of money. Examples include “DRIVING THE TRUCK”, DRIVING THE TRUCKS”, and “DRIVING TRUCKS” for “DRIVING A TRUCK” as the answer to an occupation question on a census form. All of these will be classified with the same occupation code during the analysis of the answers, so none is actually any more correct or incorrect than any other.

One way to handle this problem is to use error measures that assign small or even zero error to the three answer hypotheses listed above. The difficulty of doing this in a way that neither occasionally penalizes answer hypotheses that are essentially correct, nor rewards answer hypotheses that are obviously wrong, has been mentioned earlier.

A better way would be to incorporate the final use of the answers into the OWC or OWR process. For instance, in the example given above, the output would be an occupation code rather than an ASCII word or phrase. In this case, the assignment of correct answer codes would be much more straightforward than the assignment of correct answer words or phrases, and the error measures would also be very simple and straightforward.

Exploitation of this possibility for OWC and OWR, which is a natural extension of the type of processing that is already being carried out, may be the key to the commercial viability of OWC and OWR. Indeed, many current OCR users were sure that their application could not tolerate individual character boxes within each answer. However, when they discovered exactly how difficult OWC and OWR are at the current state of the art, they decided otherwise, and settled for OCR. Unless some further advantage can be found for OWC and OWR, these users are unlikely to see any reason to switch from OCR to something more complex.

4 System Components & Systems

In describing the complexities of integrating various components into a successful OCR system, it is useful to study one or more operational systems. Very little can be found in the published literature on the internal workings of complete systems. Granted, many of the technologies required for successful OCR have been researched and results have been published, (see the Appendix for a brief survey) but these components are typically tested in isolation and their impact on overall recognition is not measured. Also, many of the algorithms implemented in an end-to-end system

are proprietary. Companies disclose research results on pieces of their recognition systems, but no current publications can be found that disclose the details of a completely operational system, except for the work conducted at NIST.

In this section we first describe in detail the NIST form-based recognition system. This is followed by brief presentations of ERIM , MCC, and KODAK systems. It should be noted that the descriptions of the latter three systems are based on our reading of the published papers and numerous interactions that the first three authors of the paper have had with the principals responsible for the integration and evaluation of these systems.

4.1 The NIST Form-based Recognition System

NIST has been actively developing performance evaluation methods for OCR systems since 1989 [7]. In order to understand how these systems should be tested, NIST has conducted extensive research on OCR technology itself with a focus on processing handprinted responses on forms. As a result, a public domain handprint recognition system was developed whose components are fully disclosed, and in fact, the source code is available free of charge on CD-ROM [8].

The NIST Form-Based Handprint Recognition System (referred to from here on as the NIST system) provides a baseline of performance with which new and promising technologies can be compared and evaluated. As a case study, this section describes the application for which the system was designed, provides a high-level technical description of each of the components in the system, and presents performance statistics. At the time this paper was written, a second release of the NIST system was nearing completion. This section describes the components comprising the new release.

4.1.1 The Application

It has already been noted that the successful application of OCR technology requires more than system integration. State-of-the-art solutions still require customization and tuning to the problem at hand. This being true, an operational system is largely defined by the details of the application.

The NIST system is designed to read the handprinted characters written on Handwriting Sample Forms (HSF). An example image of a completed HSF form is shown in figure 2. This form was designed to collect a large sample of handwriting to support handprint recognition research. A CD-ROM named *NIST Special Database 19* (SD19), containing 3669 completed forms scanned in binary at 11.8 pixels per millimeter (300 pixels per inch), is publicly available [9]. This data set also contains over 800,000 segmented and labeled character images from these forms.

The NIST system is designed to read all but the top line of fields on the form. The system processes the 28-digit fields and the randomly ordered lowercase and uppercase alphabet fields along with the handprinted paragraph of the Preamble to the U.S. Constitution at the bottom of the form.

4.1.2 System Components

Figure 3 contains a diagram that illustrates the organization of the components of the NIST system. Generally speaking, each of these components has many possible algorithmic solutions. Therefore,

HANDWRITING SAMPLE FORM

NAME [REDACTED] DATE 08-03-89 CITY Holland STATE MI ZIP 49424

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.

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9

0123456789 0123456789 0123456789

97 420 5290 15880 932784

459 6104 53943 420501 69

3291 60118 047763 56 607

35424 183567 52 067 1258

193828 83 768 7146 79293

ixaviksbuhtpwoyqgfmdrcas

IXNVIKSD buht PWOY98efmdrcas

EDOSMZLTUHGRIXWKAFNVJYQIPCB

EDOSMZLTUHGRIXWKAFNVJYQIPCB

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 2: Example HSF form from NIST Special Database 19.

the NIST system is designed in a modular fashion so that different methods can be evaluated and compared within the context of an end-to-end system. This section provides a brief description of the most recent techniques developed by NIST for each of these tasks.

Batch Initialization: The NIST system is a noninteractive batch processing system designed to process one or more images of completed HSF forms with each invocation. This first step loads all the precomputed items required to process a particular type of form (in this case HSF forms). These items include a list of the image files to be processed in the batch, prototypical coordinate locations of dominant form structures used for form registration, a spatial template containing the prototypical coordinate location of each field on the form, basis functions used for feature extraction, neural network (NN) weights for classification, and dictionaries for spelling-correction. There are four types of fields on the HSF form: numeric, lowercase, uppercase, and the Preamble paragraph. Each type of field requires a separate set of basis functions and NN weights. Only the Preamble paragraph has a dictionary available. The use of these items will be explained in more detail later.

Because the NIST system only processes HSF forms, form identification is not utilized. Form identification can be avoided in any application where it is economically feasible to sort forms

NIST Form-Based Recognition System

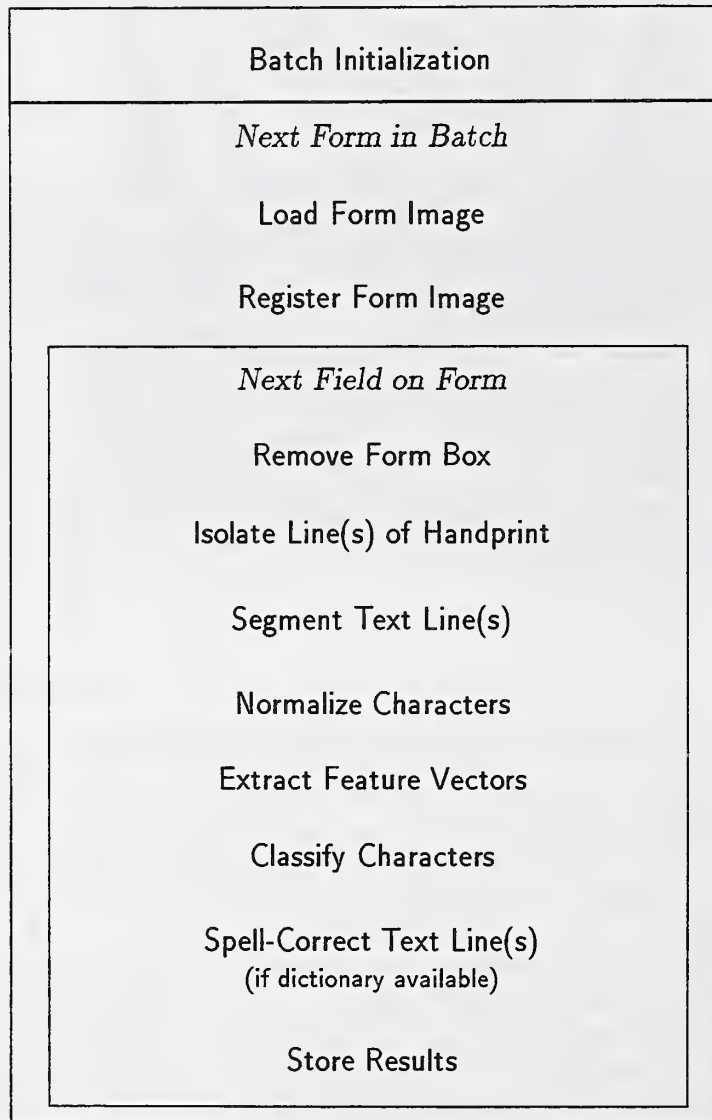


Figure 3: Organization of functional components within the NIST system.

(whether automatically or manually) into homogeneous batches. Unfortunately, this is not practical for all applications.

Load Form Image: The NIST system is strictly an off-line recognition system, meaning that the time at which images are scanned is independent of when recognition takes place. This is typical of large-scale OCR applications where operators work in shifts running high-speed scanners that archive images of forms on mass-storage devices for later batch conversion. For each form in the batch, the NIST system reads a CCITT Group 4 compressed binary raster image from a file on disk, decompresses the image in software, and passes the bitmap along with its attributes on to subsequent components.

Register Form Image: A considerable amount of image processing must take place in order to reliably isolate the handprint on a form. The form must be registered or aligned so that the fields in the image correspond to the prototypical template of fields. The NIST system uses a generalized method of form registration that automatically estimates the amount of rotation and translation in the image without any detailed knowledge of the form [10].

To measure rotational distortion, a technique similar to that invented by Postl is used [11]. The technique traces parallel rays across the image accumulating the number of black pixels along each ray using a non-linear function. This is done for a range of ray angles and the angle producing the maximum response is used to estimate the rotational skew. The image is rotated based on this estimate, and it is then analyzed to detect any translational distortion. This step capitalizes on the fact that most forms contain a fixed configuration of vertical and horizontal lines. Once the rotational skew is removed, these lines correspond well with the raster grid of the image. A run-based histogram is computed to detect the top and bottom and left and right, dominant lines in the image.

For example, to locate the top and bottom dominant lines, the horizontal runs in the image are computed. The n longest runs (in the NIST system, $n=3$) on each scanline of the image are accumulated into a histogram bin. These bins are then analyzed for relative maxima as described in [10]. The accumulation of the n longest runs effectively suppresses regions of the form containing handwriting and noise, and accentuates the lines on the form. The same analysis is conducted on vertically-oriented runs to locate the left-and right-most dominant lines. Given the locations of these lines, translation estimates in x and y are computed with respect to the coordinates of prototypical lines, and the image is translated accordingly. At this point, the fields in the image correspond to the coordinates of the prototypical spatial template.

By using this general registration technique, new form types are trained automatically. A prototypical form is scanned, its rotational distortion is automatically measured and removed, and the positions of the detected dominant lines are stored for future registrations. The result of registering 500 HSF forms is shown in figure 4. The image displayed in this figure is the result of logically ORing the corresponding pixels across the set of 500 registered images. Notice the tight correspondence of the boxes and the printed instructions.

Remove Form Box: After registration, the spatial template is used to extract a subimage of each field on the form. Fields are extracted and processed one at a time. Given a field subimage, the black pixels corresponding to the handwriting must be separated from the black pixels corresponding to the form. This is a difficult task as a black pixel can represent handwriting, the form, or the overlap of both. As all the fields on the HSF form are represented by boxes, the NIST system uses a general algorithm that locates the box within the field subimage, and intelligently removes the sides so as to

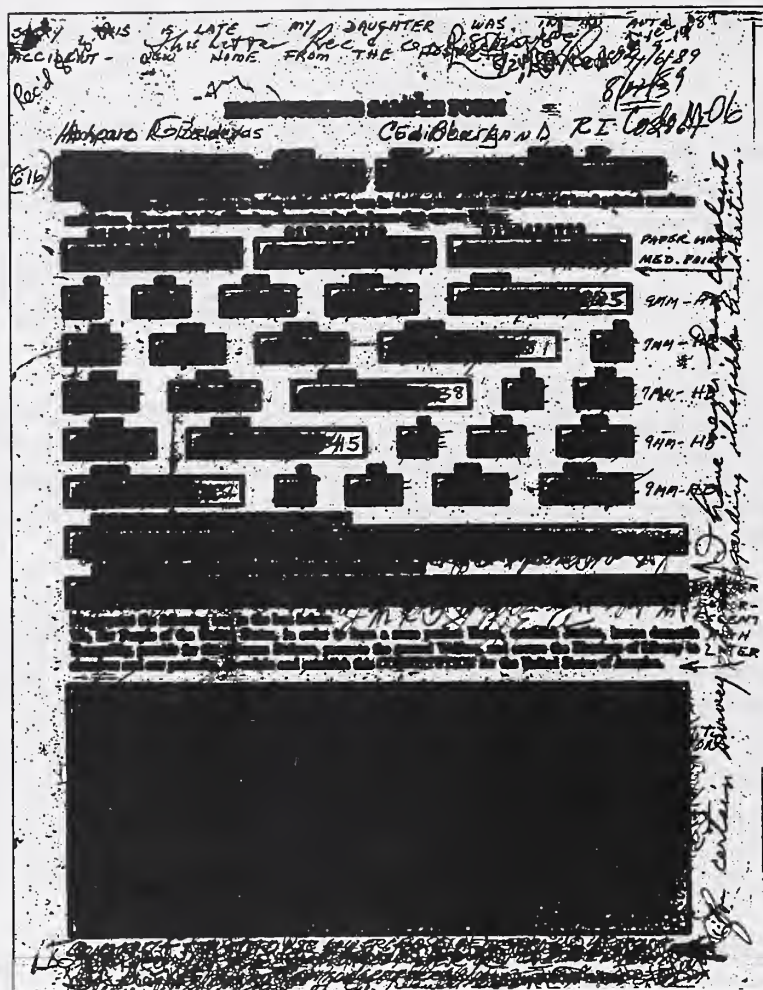


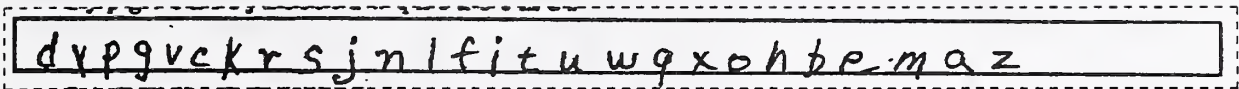
Figure 4: The results of 500 HSF forms registered and ORed together.

preserve overlapping characters [12]. The sides of the box are detected using a run-based technique that tracks the longest runs across the subimage. Then, by carefully analyzing the width of the sides of the box, overlapping character strokes are identified using spatial cues, and only pixels that are certainly part of the form's box are removed. In this way, descenders of lowercase characters, for example, are not unnecessarily truncated. Figure 5 shows several fields before and after form removal.

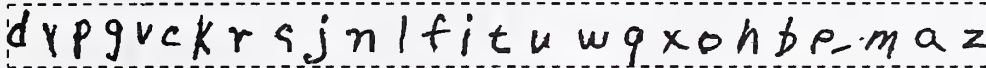
Isolate Line(s) of Handprint: The numeric and alphabetic fields on an HSF form are written as single-line responses. After the box is removed, the handprint in these fields is isolated (or lifted out) by simply extracting all the connected components (of black pixels) that overlap the interior region of the detected box.

Line isolation is much more difficult for multiple-line responses such as the handprinted paragraph of the Preamble at the bottom of the HSF form. There are no lines provided in this paragraph box to guide the writer, nor are there any instructions as to how many words should be written on a line. The handwriting is relatively unconstrained, and as a result, the baselines of the writing significantly fluctuate at times. This, along with the fact that the paragraph contains handprinted punctuation marks, makes tracking the lines of handprint difficult. Histogram projections (used

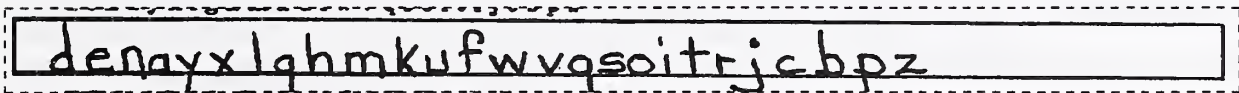
Field Subimage (A)



Isolated Handprint



Field Subimage (B)



Isolated Handprint

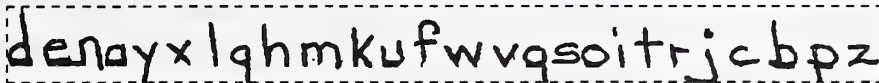


Figure 5: Results of form box removal.

extensively for isolating lines of machine printed characters) are useless in this case.

The NIST system uses a bottom-up approach to isolating the lines of handprint within a paragraph. The technique starts by decomposing the paragraph into a set of connected components. Each component is represented by a point at its geometric center. To reconstruct the handprinted lines of text, a nearest neighbor search is performed left-to-right and top-to-bottom through the two-dimensional patterns points [13]. The search is horizontally biased and links sequences of points into piecewise-linear segments. Simple statistics are used to sort components into categories of too small, too tall, problematic, and normal. Only those components determined to be normal are linked together by the search.

Given these piecewise-linear trajectories, the tops and bottoms of linked components are interpolated and smoothed forming line bands. Examples of these bands are shown in figure 6. They were derived from the paragraph image displayed directly above. These bands form a spatial map, and all the components in the image are sorted into their respective lines in correct reading order according to their overlap and/or proximity to these bands [14]. At this point, the handwriting in the paragraph has been decomposed into individual text lines.

Segment Text Lines: The segmentation method used in the NIST system is image-based and does not use the oversegmentation techniques described earlier. Connected components are used as first-order approximations to single and complete characters. Connected components frequently

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

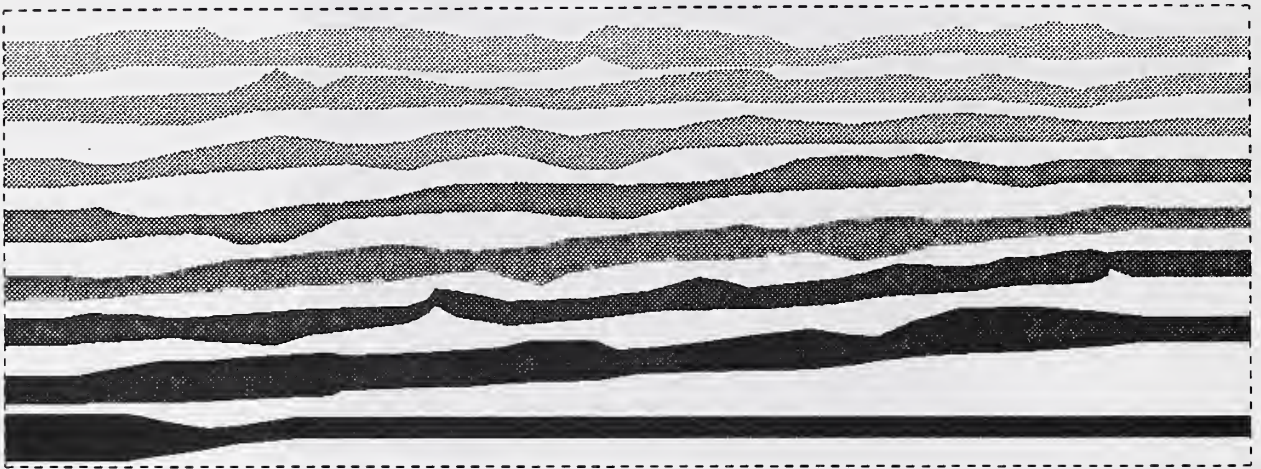


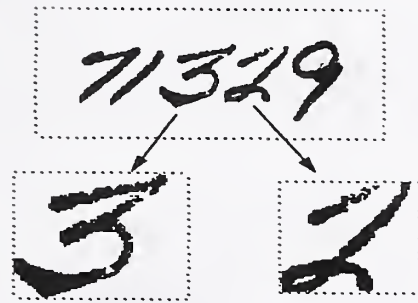
Figure 6: Line-bands computed from the paragraph image above.

represent single characters and they can be computed very quickly. On the other hand, their direct use as character segments is prone to error. Errors occur when characters touch one other and when characters are written with disconnected strokes (naturally occurring with dotted letters). The NIST system was initially designed to process the numeric fields on HSF forms. Numeric fields typically do not have any linguistic context; therefore oversegmentation schemes are of little or no use in this case.

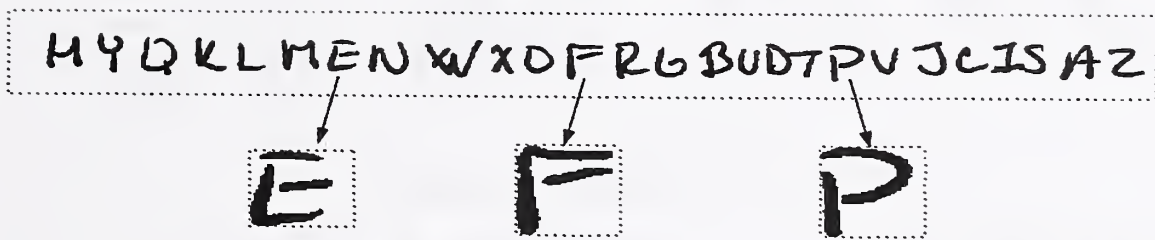
Building on the utility of connected components, the NIST system utilizes a method of handprint character segmentation that uses a simple adaptive model of writing style [15]. Using this model, fragmented characters are reconstructed, multiple characters are split, and noise components are identified and discarded. Visual features are measured (the width of the pen stroke and the heights of the characters) and used by fuzzy rules, making the method robust. Examples of segmentation results are illustrated in figures 7 and 8. The segmentor performs best when applied to single-line responses, and then even better when the fields are numeric.

With minor modification, the same method is used to segment the isolated lines extracted from paragraphs of handprinted text as described in [14].

Broken Characters



Detached Strokes



Dotted Characters

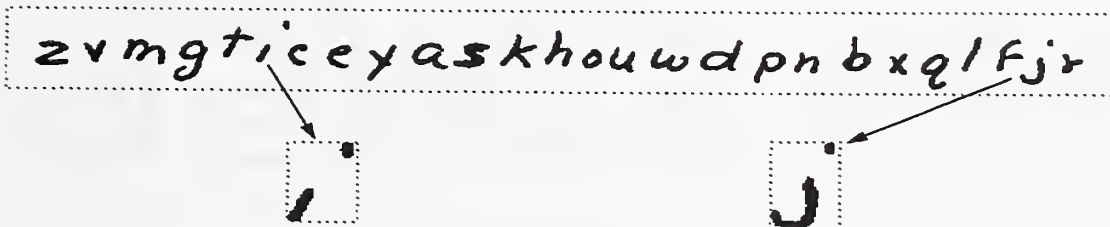


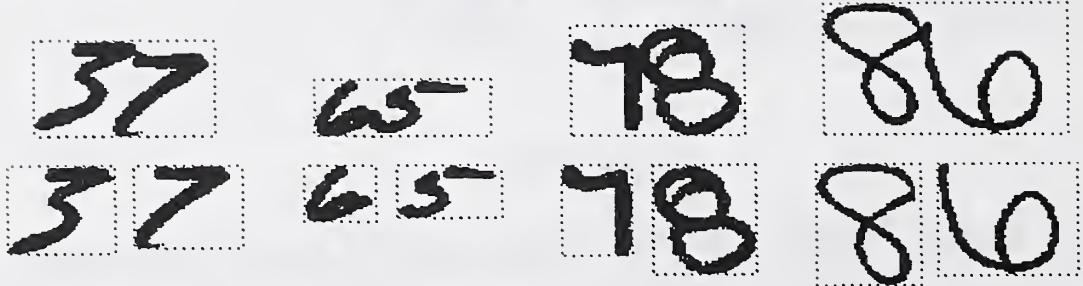
Figure 7: Segmentor results of merging components together.

Normalize Characters: The recognition technique used by the NIST system falls under the category of feature-based pattern classification. The segmented character images vary greatly in size, slant, and shape. Image normalization is performed to deal with the size and slant of writing, leaving to the recognition process primarily the task of differentiating characters by variation in shape.

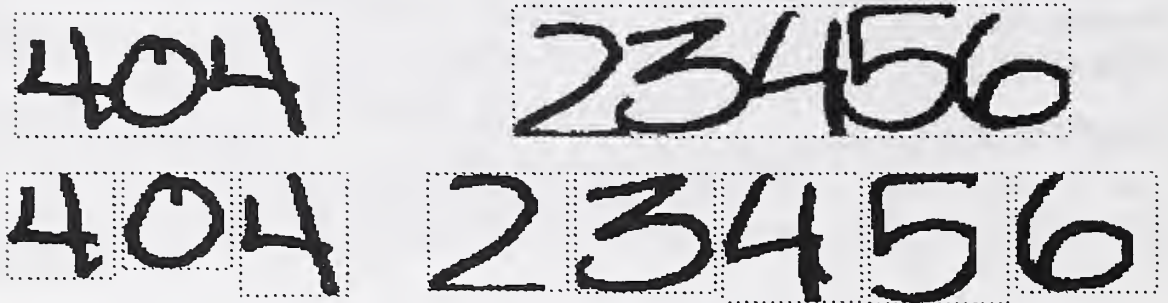
The segmented character images are size-normalized by scaling the image either up or down so that the character tightly fits within a 20×32 pixel region. The stroke width is also normalized using simple morphology. If the pixel content of the character image is too high, it is eroded (strokes are thinned), and if too low, it is dilated (strokes are widened).

Slant is removed by interpolating a line between the top left-most black pixel and the bottom left-most black pixel in the scaled image. The line (centered on the image) is used as a horizontal shear function. The slant of the character is removed as horizontal rows of pixels in the image are

Two Characters Touching



More Than Two Characters Touching



Uppercase Characters Touching

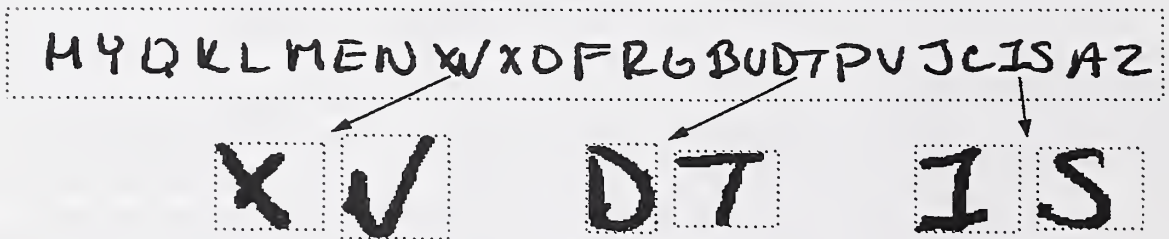


Figure 8: Segmentor results of splitting components apart.

increasingly shifted left or right from the center of the image. Upon normalization, each character is centered in a 32×32 pixel image.

Extract Feature Vectors: At this point, each character is represented by 1024 binary pixel values. The Karhunen Loève (KL) transform is applied to these binary pixel vectors in order to reduce dimensionality, suppress noise, and produce optimally separable features (in terms of variance) for classification [16].

A training set of normalized character images is used to compute a covariance matrix which is diagonalized using standard linear algebra routines, producing eigenvalues and corresponding eigenvectors. This computation is expensive, but is done once off-line, and the top n ranked eigenvectors are stored as basis functions and used subsequently for feature extraction. Feature vectors of length 128 are used in the NIST system; each coefficient in the vector is the dot product of an eigenvector with the 1024 pixel vector of the character being classified.

Classify Characters: Once segmented characters are represented by feature vectors, a host of pattern classification techniques can be applied. NIST has conducted extensive research on classification methods that utilize machine learning; most of these have been various types of neural networks [17]. In previous work, the Probabilistic Neural Network (PNN) [18] was shown to provide better zero-reject error performance on character classification problems than the Radial Basis Function (RBF) and Multi-Layer Perceptron (MLP) NN methods. Later work [19] demonstrated that various combinations of NN's could provide performance equal to PNN and substantially better error-reject performance, but these systems were very expensive to train and were much slower and less memory-efficient than MLP-based systems.

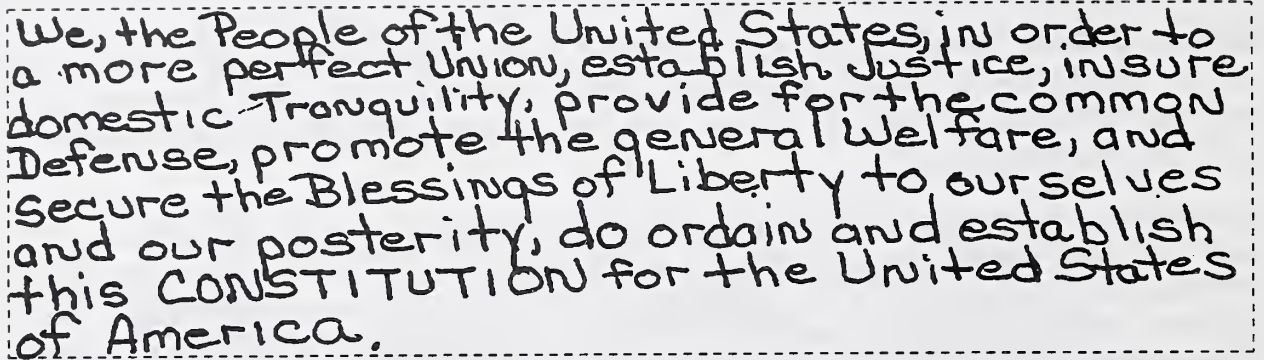
NIST has developed a training method that produces MLP networks with performance equal to or better than PNN for character recognition [20]. This is achieved with a single three-layer network by making fundamental changes in the network optimization strategy. These changes are: 1) Neuron activation functions are used which reduce the probability of singular Jacobians; 2) Successive regularization is used to constrain the volume of the weight space; 3) Boltzmann pruning [21] is used to constrain the dimension of the weight space. All three of these changes are made in the inner loop of a conjugate gradient optimization iteration [22] and are intended to simplify the training dynamics of the optimization. On handprinted digit classification problems, these modifications improve error-reject performance by factors between 2 and 4 and reduce network size by 40% to 60%.

To classify a character, the appropriate eigenvectors (or basis functions) and MLP weight matrices must be loaded into memory. As mentioned earlier, this is accomplished during batch initialization. Using the eigenvectors, the normalized image is transformed into a feature vector. The feature vector is then presented to the MLP network. The result is an assigned classification along with a confidence value.

Spell-Correct Text Line(s): The only field on the HSF form that has any linguistic context is the Preamble field. The Preamble contains 38 distinct words all of which are used in the dictionary.

The dictionary-based processing performed by the NIST system is somewhat different from other correction techniques described in this paper. Up to this point, segmented character images have been extracted from the handprinted paragraph, sorted into reading order line by line, and classified. This results in one contiguous character stream for each line in the paragraph. The MLP weights used to process the Preamble paragraph were trained to map lowercase and uppercase instances of

Field Subimage



We, the People of the United States, in order to 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.

Raw Classifications

WEJTPEOPIEOPTHEUNITEASTATFSJLNORDERTO
AMOREPQRFKTUNIONJEBTAEIBHJUSTICEJINSURE
DOMDLCITRONGUIIJTYIPROVIDEFPRTHFCOMMQN
DEFENBELPROMOTETHEGENEMIWELNRELAND
SECURETHEBLCSSINPOFLIBBHYTOOURSELUES
ANDOURPOSTERLTYIDOOBINANDQDADLISH
THISCONETITUTIBNFORTHEUNIFEDSBTES
OFAMERICA

Spell-Corrected Words

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

Figure 9: Results from processing the top paragraph image.

the same letter into the same class, making the output of the classifier case-invariant. No interword gaps are identified by the system at this point. Examples of the raw classifications are shown in the middle of figure 9 for the paragraph image displayed directly above then.

Words are parsed from each line of raw classifications by applying the preloaded dictionary as described in reference [13]. This process identifies words within the character stream while simultaneously compensating for errors due to wrong segmentations and classifications. The limited size of the dictionary helps offset the burden placed on this process.

Hypothesized words are constructed from sequences of the classifier outputs and then matched to the dictionary. When there is a sufficiently good match, the dictionary word is output, the process resynchronizes itself in the character stream, and parsing resumes. The matching criterion takes into account the number of errors in the word relative to the length of the word. In this way longer words are permitted to tolerate more errors.

Store Results: When processing the Preamble paragraph, the system produces a sequence of spell-corrected words as output. Results of spell-correcting the paragraph image in figure 9 are listed at the bottom of the figure. Shorter words such as articles and prepositions tend to be frequently deleted and in other places inserted, while the system does a reasonable job of recognizing longer words. This type of dictionary processing is better suited to word-spotting than to full OCR transcription.

For the numeric and randomly ordered alphabet fields, the NIST system outputs for each segmented character an assigned class and its associated confidence as determined by the MLP classifier.

4.1.3 Performance Evaluation

The NIST system has been designed to follow the general guidelines presented in section 3. This section evaluates and compares the performance of the new NIST system (HSFYSYS2) described above to its older counterpart (HSFYSYS1) [8]. The older recognition system uses less robust methods and a PNN-based character classifier. Both of these systems were designed to process the HSF forms distributed in SD19. HSFYSYS1 is capable of processing the forms in SD19 partitions hsf_0, hsf_1, hsf_2, & hsf_3, whereas HSFYSYS2 is capable of processing every one of the 3669 forms in the database.

4.1.4 Accuracies and Error Rates

In order to compile statistics on accuracy and error rates, both systems were run across the forms in SD19 and recognition results were stored in files. Recognition system classifications were stored in *hypothesis* files, and their associated confidence values were stored in *confidence files*. Once generated, these files were processed using the NIST Scoring Package [23], and performance statistics were compiled at the character and field levels. Table 1 lists the digit recognition results of running HSFYSYS1 on the first 2,100 forms (partitions hsf_0 to hsf_3) in SD19. The forms in the remaining partitions differ enough that the method of form registration used in HSFYSYS1 fails. The top portion of the table reports character-level statistics, and the bottom reports field-level accuracies. 33 of the 2,100 forms were rejected due to form registration failures and their characters are not included in the table. It was determined that a majority of these failures occurred due to writing outside the provided boxes with continued responses or annotations. A small number (about 5) failed registration due to scanner noise in critical areas on the form.

The performance statistics in table 1 can be compared to those listed in table 2. The second table reports the digit recognition results from the new MLP-based system, HSFYSYS2. These two systems use significantly different algorithms for more than just classification. A complete system description of HSFYSYS1 can be found in [8] and [24].

As can be seen from the two tables, HSFYSYS2 performs significantly better than HSFYSYS1. In terms of digit accuracy, HSFYSYS2 is 3.2% more accurate at 96.3% and it recognizes 86% of the digit fields entirely correct (6% more than HSFYSYS1). This difference in accuracy is primarily attributed to the different segmentation methods used in both systems, not to the different classifiers. Studies have shown that at zero-rejection, PNN and the new MLP classifier have similar accuracy [20]. Looking at deletion errors, HSFYSYS2 cuts them by 80% which testifies to the improved performance of the system's statistically adaptive segmentor. In addition, HSFYSYS2 is capable of registering every

HSFSYS1 DIGIT RECOGNITION

(A) Characters					
	hsf_0	hsf_1	hsf_2	hsf_3	Total
Correct	92.9% 59288	93.0% 59868	92.6% 58258	93.9% 72872	93.1% 250286
Substituted	3.9% 2481	3.9% 2531	4.1% 2578	3.6% 2782	3.9% 10372
Inserted	0.6% 398	0.7% 475	0.6% 385	0.7% 539	0.7% 1797
Deleted	3.2% 2061	3.0% 1951	3.3% 2084	2.5% 1956	3.0% 8052
Total	63830	64350	62920	77610	268710

(B) Fields					
	hsf_0	hsf_1	hsf_2	hsf_3	Total
Correct	79.0% 10865	79.5% 11012	79.1% 10717	81.7% 13662	79.9% 46256
Total	13748	13860	13552	16716	57876

Table 1: HSFSYS1 accuracies and error rates for digit fields across the first part of SD19. Part (A) reports character-level statistics and (B) reports field-level statistics.

form in SD19, with only 10 fields (6 digit fields, 1 lowercase field, and 3 Preamble fields) rejected due to poor image quality. The characters from these 10 fields have been tallied into the reported statistics as deletions.

The results of uppercase recognition can be compared between tables 3 and 4. HSFSYS2 recognizes uppercase characters at nearly 90% (4.6% higher than HSFSYS1). Again, the difference in performance can be primarily attributed to the segmentation methods used. With HSFSYS2, insertion errors are reduced by 46% and deletion errors by 58%.

Lowercase statistics are listed in tables 5 and 6. HSFSYS2 correctly recognizes not quite 80% of the lowercase characters in SD19. Not only does the new system employ a new segmentor, it also conducts intelligent line removal that preserves character stroke data that overlaps with the form and extends beyond the immediate limits of the field. In an independent study, it was shown that one can expect up to a 3% improvement in lowercase accuracy when using this method of line removal [12]. The difference between HSFSYS1 and HSFSYS2 is 2.8%, some of which can be directly attributed to the line removal. Adaptive character segmentation is also contributing, as insertion errors are reduced by 70%. This demonstrates the segmentor's ability to compose characters from multiple components, as unattached fragments contribute to insertion errors. On the other hand, the number of deletion errors increases with HSFSYS2. This leads one to conclude that the adaptive segmentor may be over-aggressive in merging components, and not aggressive enough when it comes to splitting touching characters. Independent studies have shown that the general segmentation method used in HSFSYS2 can benefit from further refinement for lowercase characters [15].

The last pair of tables (tables 7 and 8) list the results of recognizing words across SD19's Preamble fields. SD19 has completed Preamble paragraphs only in its first 4 partitions. These word-level statistics were computed by tokenizing each word in the system output. The NIST Scoring Package was then used to align the word tokens with the known Preamble text and statistics were accumulated. Much effort was spent in improving the line isolation algorithm used in HSFSYS2 [14]. Even so, overall word accuracy only improved 2.3% (61.6% to 63.9%). Considerable work still remains

HSFSYS2 DIGIT RECOGNITION

(A) Characters									
	hsf_0	hsf_1	hsf_2	hsf_3	hsf_4	hsf_6 [†]	hsf_7	hsf_8	Total
Correct	96.6% 62772	96.5% 62731	96.1% 62486	97.2% 75804	93.7% 60933	97.3% 63144	96.3% 62615	96.9% 8817	96.3% 459302
Substituted	2.8% 1816	2.9% 1871	3.0% 1972	2.4% 1891	5.5% 3575	2.1% 1367	3.0% 1920	2.5% 229	3.1% 14641
Inserted	0.7% 454	0.7% 487	0.7% 425	0.7% 571	0.8% 489	0.5% 318	0.6% 422	0.3% 25	0.7% 3191
Deleted	0.6% 412	0.6% 398	0.8% 542	0.4% 305	0.8% 492	0.6% 359	0.7% 465	0.6% 54	0.6% 3027
Total	65000	65000	65000	78000	65000	64870	65000	9100	476970

(B) Fields									
	hsf_0	hsf_1	hsf_2	hsf_3	hsf_4	hsf_6 [†]	hsf_7	hsf_8	Total
Correct	86.3% 12084	86.7% 12139	86.2% 12068	88.6% 14881	77.5% 10855	89.6% 12517	86.5% 12105	88.2% 1728	86.0% 88377
Total	14000	14000	14000	16800	14000	13972	14000	1960	102732

Table 2: HSFSYS2 accuracies and error rates for digit fields across SD19. Part (A) reports character-level statistics and (B) reports field-level statistics. ([†]Segmented character images from the writers in this partition were used to train the neural network classifiers.)

HSFSYS1 UPPERCASE RECOGNITION

Characters					
	hsf_0	hsf_1	hsf_2	hsf_3	Total
Correct	84.7% 10808	85.5% 11004	84.7% 10661	86.2% 13377	85.3% 45850
Substituted	12.8% 1636	11.8% 1524	12.8% 1609	12.0% 1858	12.3% 6627
Inserted	4.7% 599	4.3% 556	5.5% 687	4.6% 717	4.8% 2559
Deleted	2.5% 322	2.7% 342	2.5% 314	1.8% 287	2.4% 1265
Total	12766	12870	12584	15522	53742

Table 3: HSFSYS1 accuracy and error rates for uppercase fields across the first part of SD19.

in improving the segmentation of vertically and horizontally touching characters, the detection of punctuation marks, and dictionary-based are spell correction.

As a final note on these accuracy statistics, note that results are reported with HSFSYS2 run across the entire set of forms in SD19. This is one of the largest published experiments of its kind, and it is reproducible by purchasing the SD19 database from NIST. In all, sample handwriting from 3669 writers was tested and a total of 109,200 words and 667,758 characters were recognized and scored. As SD19 is our only handprint database, training samples were extracted from specific writer partitions and used to train the PNN and MLP character classifiers off-line. From the 667,758 characters, 109,719 were used in training. In the case of digits, the writers in hsf_6 (6,1094 characters) were used in the training set, and in the case of upper and lowercase, writers in both hsf_4 and hsf_6 (totalling 24,420 uppercase characters and 24,205 lowercase characters) were used.

Comparing the HSFSYS2 results on hsf_6 to other partitions, it is interesting to see that the

HSFSYS2 UPPERCASE RECOGNITION

Characters									
	hsf_0	hsf_1	hsf_2	hsf_3	hsf_4 [†]	hsf_6 [†]	hsf_7	hsf_8	Total
Correct	89.1%	89.3%	89.2%	89.9%	90.3%	93.0%	88.9%	89.5%	89.9%
	11587	11603	11591	14029	11740	12062	11563	1629	85804
Substituted	9.7%	9.3%	9.4%	9.1%	9.1%	6.5%	10.4%	8.8%	9.1%
	1256	1210	1223	1425	1187	847	1347	161	8656
Inserted	2.3%	2.3%	2.5%	2.7%	3.4%	2.4%	2.6%	1.4%	2.6%
	294	300	320	426	436	315	336	25	2452
Deleted	1.2%	1.4%	1.4%	0.9%	0.6%	0.5%	0.7%	1.6%	1.0%
	157	187	186	146	73	65	90	30	934
Total	13000	13000	13000	15600	13000	12974	13000	1820	95394

Table 4: HSFSYS2 accuracy and error rates for uppercase fields across SD19. ([†]Segmented character images from the writers in this partition were used to train the neural network classifiers.)

HSFSYS1 LOWERCASE RECOGNITION

Characters					
	hsf_0	hsf_1	hsf_2	hsf_3	Total
Correct	75.1%	76.8%	76.5%	78.7%	76.9%
	9593	9890	9625	12217	41325
Substituted	22.9%	21.4%	22.3%	19.9%	21.5%
	2927	2748	2804	3092	11571
Inserted	3.2%	2.6%	3.0%	3.2%	3.0%
	405	336	381	500	1622
Deleted	1.9%	1.8%	1.2%	1.4%	1.6%
	246	232	155	213	846
Total	12766	12870	12584	15522	53742

Table 5: HSFSYS1 accuracy and error rates for lowercase fields across the first part of SD19.

inclusion of hsf_6 in the classifier training does have a small influence. With digits, HSFSYS2 is 97.3% correct on hsf_6 whereas the results on hsf_3 are almost as good at 97.2%, and the other partitions (with the exception of hsf_4) range between 96% to 97%. The writers in hsf_4 are from a different population and are known to be statistically more difficult to recognize [25]. The influence of training is a bit more pronounced for the results on uppercase and lowercase fields. On uppercase, HSFSYS2 is 93% correct on hsf_6, and the other partitions range between 89% to 90%. For lowercase, HSFSYS2 is 83% correct on hsf_6, and other partitions range between 77% to 80%. These small differences (particularly for the digits) demonstrate that the MLP character classifier is doing a reasonably good job at generalizing on writers it hasn't seen during its off-line training. The MLP-based system doesn't generalize as well on uppercase and lowercase recognition in part because fewer training samples were used than for digits.

4.1.5 Rejection versus Error Rate

Using a machine for OCR in many ways complements the performance of humans. Machines are very efficient in doing tasks that are primarily repetitive and reflexive, whereas humans quickly fatigue under these conditions. Humans, on the other hand, are very adept at performing tasks

HSFSYS2 LOWERCASE RECOGNITION

Characters									
	hsf_0	hsf_1	hsf_2	hsf_3	hsf_4 [†]	hsf_6 [†]	hsf_7	hsf_8	Total
Correct	77.2%	78.6%	77.8%	80.4%	82.6%	82.9%	79.1%	75.5%	79.7%
	10042	10218	10109	12541	10739	10756	10280	1374	76059
Substituted	20.4%	19.2%	20.3%	17.8%	15.1%	15.2%	18.4%	20.5%	18.1%
	2654	2495	2645	2781	1967	1966	2397	373	17278
Inserted	0.9%	0.8%	0.8%	1.1%	0.8%	0.6%	0.9%	1.0%	0.9%
	119	102	110	168	110	75	114	18	816
Deleted	2.3%	2.2%	1.9%	1.8%	2.3%	1.9%	2.5%	4.0%	2.2%
	304	287	246	278	294	252	323	73	2057
Total	13000	13000	13000	15600	13000	12974	13000	1820	95394

Table 6: HSFSYS2 accuracy and error rates for lowercase fields across SD19. ([†]Segmented character images from the writers in this partition were used to train the neural network classifiers.)

HSFSYS1 PREAMBLE RECOGNITION

Words					
	hsf_0	hsf_1	hsf_2	hsf_3	Total
Correct	60.3%	59.8%	60.5%	65.0%	61.6%
	15403	15387	15237	20166	66193
Substituted	15.2%	14.3%	14.0%	12.7%	14.0%
	3871	3676	3525	3943	15015
Inserted	1.1%	0.9%	1.1%	1.1%	1.0%
	283	224	270	335	1112
Deleted	24.5%	25.9%	25.5%	22.3%	24.4%
	6258	6677	6406	6935	26276
Total	25532	25740	25168	31044	107484

Table 7: HSFSYS1 accuracy and error rates for Preamble fields across SD19.

requiring higher-level reasoning, and as a result they provide more robust but much slower solutions to complex problems. Accounting for these differences, successful recognition systems allow the machine to perform the bulk of the work, and on an exception basis, humans are used to resolve ambiguities and potential errors. This is accomplished through rejection mechanisms that automatically route low-confidence machine decisions to humans for verification. This section compares the ability of the two NIST recognition systems to effectively reject low-confidence character classifications.

The graph in figure 10 plots rejection versus error rates with error plotted on a logarithmic scale. Results are shown for both HSFSYS1 and the new system HSFSYS2, and the results are broken out by digit, upper, and lowercase recognition. In general, as the number of rejected character classifications increases, the error rate on the remaining accepted (or non-rejected) classifications decreases, and accuracy improves. Also, the impact of rejection on accuracy tapers off as more and more characters are rejected. In the figure, the bottom two curves represent the performance of the new and old systems on recognizing characters in the numeric fields on the HSF forms. With no rejection, HSFSYS2 has an error rate near 4%, and HSFSYS1 has an error rate over 7.5%. As the number of rejected digit classifications is increased, the error rate proceeds to drop, but HSFSYS2 falls at a significantly faster rate than does HSFSYS1. The difference in the slope of the two digit curves testifies to the robustness of the MLP classifier used in HSFYS2 over the PNN

HSFSYS2 PREAMBLE RECOGNITION

Words					
	hsf_0	hsf_1	hsf_2	hsf_3	Total
Correct	62.6%	62.4%	62.7%	67.2%	63.9%
	16276	16231	16304	20961	69772
Substituted	15.4%	15.2%	14.7%	12.6%	14.4%
	4012	3958	3833	3940	15743
Inserted	1.6%	1.0%	1.1%	1.3%	1.3%
	405	260	287	418	1370
Deleted	22.0%	22.4%	22.6%	20.2%	21.7%
	5712	5811	5863	6299	23685
Total	26000	26000	26000	31200	109200

Table 8: HSFSYS2 accuracy and error rates for Preamble fields across SD19.

classifier used in HSFSYS1. The digit error rate of HSFSYS2 continues to drop to nearly 1.2% at 15% rejection. One concludes from these results, that in terms of recognizing numeric fields, the new NIST recognition system has hale the error-rate of the original syatem for a moderate rejection level.

The differences between the two systems, is less dramatic with uppercase and lowercase recognition. The middle two curves in figure 10 correspond to the results of recognizing the uppercase alphabet fields on the HSF forms. The HSFSYS2 curve does fall off slightly faster than does HSFSYS1's, but the distance between the curves is not as large as that of the digit curves. With no rejection, HSFSYS2 has an error rate of almost 13% and HSFSYS1 just over 19%. The two lowercase curves are even closer to each other, and they maintain pretty much the same relative distance across the range of rejections plotted. This emphasizes that lowercase recognition is still the most difficult for the NIST systems. The small increase in separation between these two curves as rejection increases can be attributed to a combination of two factors. First, the decision surfaces trained within the MLP classifier for lowercase are much more complex than those of uppercase, and the decision surfaces for uppercase are more complex than those of digits [20]. Second, the challenges remaining in the system that are effecting accuracy lie primarily in components other than the classifier. Otherwise, the relative slopes in the upper and lowercase curves would more closely resemble those of the digit classifications.

4.2 Other Systems

The NIST OCR System Conferences evaluated a number of OCR systems developed by University researchers and companies in the U.S. and abroad. A general discussion of the evaluations is presented in Section 5. Due to proprietary considerations, only a high level understanding of the system configurations and module functionalities has been possible. For the sake of completeness we present brief descriptions of three of the systems evaluated at the NIST OCR System Conferences. These are the systems designed at ERIM [26, 27], MCC [28, 29] and KODAK [30]. The ERIM system is based on an Image Analysis approach, while the MCC and KODAK systems have taken an integrated segmentation and recognition approach using NN's. The KODAK system can be viewed as an enhanced and improved MCC system.

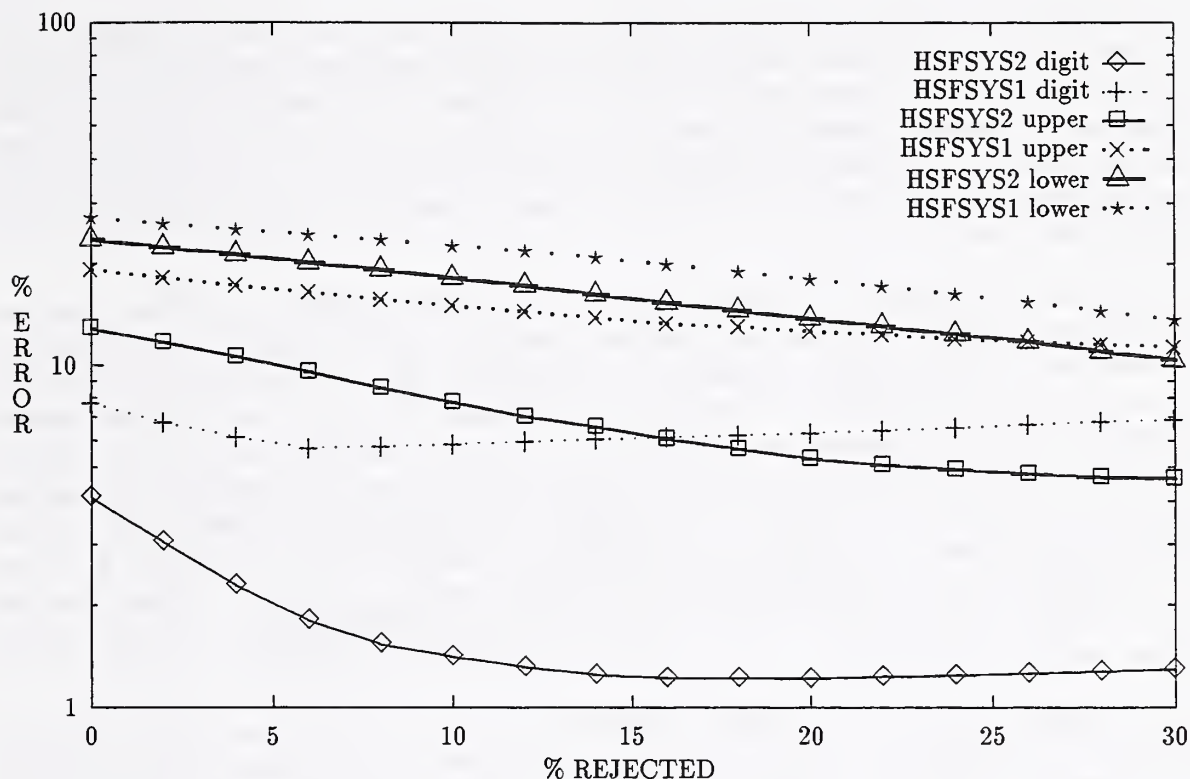


Figure 10: Rejection versus error rates for digit, upper, and lowercase recognition between HSFSYS1 and HSFSYS2.

4.2.1 The ERIM System

Application of Ullman's minimal mapping theory [31] of minimal correspondence has resulted in a system that gave the lowest field-error rates and lowest field-distance rates at all rejection rates in the Second OCR Systems Conference [4]. This system, designed by ERIM, uses an effective combination of techniques such as feature-based matching, NN's for learning feature attributes, and unsupervised clustering schemes for generating character prototypes. After the mundane preprocessing steps such as noise removal, slant correction, and scaling to a fixed height, feature extraction driven by morphological operations is performed on multi-letter images. By identifying valleys in the core stroke image, possible segmentation points (PSPs) are identified and features are extracted between pairs of PSPs. The features used are core area (where the center stroke information falls) and discrete features extracted from six feature images (stroke above and below the core, holes and concavities, lower and upper limbs in the core). Finding the discrete features amounts to extracting connected components for holes, concavities and, lower and upper strokes, whereas for lower and upper limbs, the extracted connected components are broken into local maxima and minima. To generate the matches between character samples and character models, a random graph structure, in which vertices represent character features and edges the spatial relationships between feature pairs, is used. Features extracted as described above are represented using the attributes of position and shape. The shape attribute is extracted using an NN training scheme.

For each PSP pair, a correspondence matching algorithm formulated as an assignment problem is invoked. For each string in the given lexicon, an optimal match between the word image features

and the model of the string is determined using a dynamic programming (DP) technique. Since hand-printing can produce many variations, several character templates for each of the 26 lower-case, printed, 26 upper-case, and 26 lower-case cursive characters are generated from the training set using an unsupervised clustering procedure. A total of 457 character model prototypes is used in the system derived from 996 word images. This system judged to be the best performing OCR system at the Second OCR System Conference, is a good example of a commercial application of computer vision research.

One of the important features of the ERIM system is the ability to generate confidence measures along with the matched word. The word confidence measure and the ranking is derived from the probability that a word model for the given string generated the set of features extracted from the word image. The distributions for the position and shape attributes of the extracted features are derived as part of the feature extraction process. Using assumptions such as the conditional independence of the probability distribution of features, that the spatial relationships between the features are independent of any feature other than those connected to the given ones and that the conditional distributions of the edges are independent, one can generate the probability of a word from probabilities of its component features. The ERIM system provides an excellent example of a pure OWR (as distinguished from OCR followed by OWC, illustrated by the NIST system described earlier in this report).

Figure 11 illustrates aspects of the ERIM segmentation process for the word *medical* appearing in an answer box. The final ten segments are shown at the bottom of the figure. Figure 12 shows the combinations of segments produced by the ERIM segment-combining process, including some that are rejected as very likely multiple characters. A total of 34 unions were created, and a total of 24 were accepted as likely character candidates. Each union was recognized by the ERIM isolated-character recognizer, but the output of the recognition process was not the character with the highest recognition confidence value. Instead, the output was a vector of confidence values, one for each letter of the alphabet.

The ERIM word-construction algorithm tests each word from the dictionary of allowed words against the set of 24 confidence vectors in a search for the best mapping of the segments and unions onto the letters in each word. Figure 13 illustrates the ERIM algorithm for the case where the dictionary entry is identical to the unknown answer. (Note that this is possible only when the unknown answer is contained in the dictionary as described above.) Each of the original segments of the word *medical* corresponds to a column in the matrix. Each row of the matrix corresponds to a letter in *medical*. A dynamic programming algorithm is used to map the segments along the top of the matrix onto the letters along the side of the matrix in such a way as to maximize a product that is described below.

The path through the matrix in Figure 13 illustrates the mapping of the segments and unions of *medical* onto the letters of *medical* by the dynamic programming algorithm. Only diagonal steps down and to the right, and horizontal steps to the right, are allowed on this path.

A diagonal step in Figure 13 signals the start of a new letter, while a horizontal step signals a union of segments. The path starts with a diagonal step down from the upper left hand box to signal the start of the first letter. The path takes two horizontal steps before taking another diagonal step down. This indicates that the first three segments along the top of the matrix are to be treated as a union and mapped onto the first letter *m* on the left-hand side of the matrix. It is convenient to suppose that the dynamic programming algorithm extracts the confidence with which that union

ERIM	Field Segmentation	2nd Central OCR Systems Conference
<ul style="list-style-type: none"> <input type="checkbox"/> Goal is oversegmentation of characters <input type="checkbox"/> Based on outer contours of word images <input type="checkbox"/> Use local minima and maxima as segmentation points <input type="checkbox"/> Result is primitive segments 		




ERIM	Field Segmentation	2nd Central OCR Systems Conference
Contours of Field Image		
Segmentation Points		
Resulting Primitive Segments		

Figure 11: Segmentation of the word “medical” by the ERIM system.

ERIM	Unions				2nd Census OCR Systems Conference
	Union Size (Number of Primitive Segments)				
Primitive Segments	1	2	3	4	
I	A	M	ML		
I	A	AL	AE		
I	IL	IE	IED		
L	E	ED	EDL		
E	E	ED	EDL		
D	DI	DIC	DIEA		
I	IC	ICA	ICAL		
C	CA	CAL			
A	AL				
L					

Figure 12: Formation of unions from segments by the ERIM system

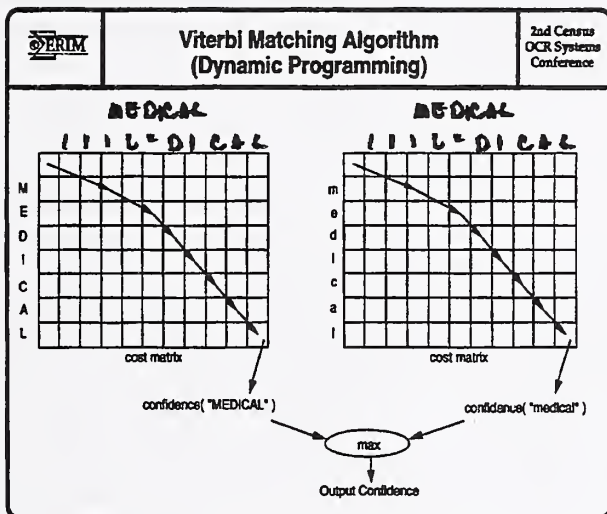


Figure 13: Selection of the best combination of unions of segments to represent a word from a dictionary of expected words.

was recognized as an m and puts this confidence into the last box before the diagonal step.

The diagonal step signals the start of the second letter. This time the union of two segments is mapped onto the letter E , and the confidence of the recognition of this union is put into the last box before the diagonal step that signals the start of the next letter. This time a single segment that looks like a D is mapped onto the letter D , and the confidence of the recognition of this segment as a D is put into the box under that segment. This process proceeds with diagonal steps through the letters I , C , A , and L .

After this process is completed for every combination of letters and segments, there is a confidence value in the last box associated with each letter (the box that precedes a diagonal step in the matrix). The dynamic programming algorithm chooses the path that maximizes the product of these confidence values for any given combination of letters and segments. Thus a separate word-level confidence value is produced as a measure of how consistent each dictionary word is with the segments produced from the answer in some box. Finally, the word from the dictionary that has the greatest word-level confidence value is chosen as the best match.

In a subsequent paper [27] ERIM presented the design details of a phrase recognition system, where a phrase can include multiple words, abbreviations, etc. Such applications arise in matching a database record such as business, street name, etc. The key components of this system are a word segmentor, a word recognizer, a lexicon generator and a dynamic phrase recognizer. As the intent is to handle both printed and cursive words, several techniques are used in the system. In the case of the word segmentor module, machine-printed words can be separated by using simple projection-based techniques. Since projection-based techniques do not work for cursive words, both over- and under-segmentation of the line image is done to mark possible locations of word breaks. Under-segmentation is done using breaks at large gaps between successive components, while over-segmentation is done at locations such as capital letters, punctuation, etc. to produce multiple hypotheses to be evaluated by the word recognizer. To keep the system's capabilities as general as possible, evaluation of word hypotheses is done using four kinds of word recognizers. Two of the word recognizers use hidden Markov models trained on printed and cursive words, respectively. The third is the word recognizer designed using the feature correspondence matching approach described earlier. The fourth recognizer is based on segmentation followed by evaluation using a dynamic programming technique. Since all of these modules return their top few choices and the associated confidence measures, a voting scheme and reject curve are used for deciding on the top choice and its new confidence measure.

The recognized words are structured into phrases using a lexicon-directed dynamic programming approach. For the case of matching of street names, the phrase lexicon is generated using aliases, abbreviations, suffixes, and prefixes of street names. The results of experiments with 1200 hand-written mail-pieces containing street address information indicated acceptable performance.

4.2.2 The MCC System

The MCC system is based on a novel approach to Integrated Segmentation and Recognition (ISR) using NN's [32, 29]. The network uses a sliding window and attempts to recognize a character only if the window is centered on it. If the window center is between two characters a NOT-CENTERED condition is declared. The training set is prepared by a human who clicks at the horizontal center of each digit in sequence using a mouse button, and associates a digit label with each such center.

The target output is generated during training as follows: When the center position of the window is within ± 2 pixels of the center of a character, the target value of the character's output node is set at the maximum with the target values of the NOT-CENTERED mode and all other characters set at the minimum, When the center position of a window is within ± 2 pixels of the halfway point between two character centers, the reverse situation holds. The NN is a two-hidden-layer MLP network, with local shared connections in the first hidden layer, and local connections in the second hidden layer. The weight-sharing techniques used to reduce memory requirements often yielded poor results.

This approach has been extended to handle hand-printed digits and characters [29]. An added feature to the approach described above is the introduction of a saccadic scan. This is accomplished by training a MLP net to estimate the distance to the next character on the right so that at run-time, the system can navigate along a character field by effectively jumping over the spaces between characters. This effectively reduces the number of forward passes required compared to the exhaustive scanning scheme [32]. In addition to the saccade system, when the recognition of handwritten characters is considered, post-processing techniques that use dictionaries result in significant reductions in the field error rate.

4.2.3 The KODAK System

It has been observed [30] that the MCC system using saccades worked well on nicely spaced characters, but if the localization was on inter-character space, then the saccade could move to the previous character and classify it the second time. The reason given for this is the discontinuities in weight values in the first processing layer lead to significant changes in input that get fed to the NN when even small shifts are present in the input image. The saccade network in the MCC system has the burdensome task of making saccades as well as classification of characters using a single complicated network. The KODAK system is an improvement over the MCC system using the following modifications: a) projections of Gabor-basis sets are used as inputs instead of raw input image windows, and the task of positioning and classification are done by two separate NN's.

The positioning network accepts as inputs size-normalized subfields of disconnected groups of characters with a spacing of six pixels and with two black rows appended at the top and bottom. The resulting image is processed by a feature extraction layer that basically performs projection onto Gabor basis sets. The input layer is followed by three hidden layers and an output layer with 12 outputs. As the input window slides to each pixel position in the input field, the output layer produces 12 activations corresponding to the chosen pixel position in 12 consecutive windows. Using a weighted average of these activations and some post processing, a waveform with peaks characterizing the positions of characters is produced. Based on extensive testing with the NIST database, the author reports that 91% of all characters were positioned to within a pixel. Comparisons with the performance of the MCC saccade system show the superior performance of the positioning network.

The classifier network is similar to the positioning network, but has only two hidden layers, and an output layer with 10 outputs for digits. To account for the errors in positioning the characters, the classifier network was trained and tested on five copies of input fields corresponding to shifts of up to and including two pixels on either side of the center pixel. Comparison to the saccade system shows improved performance.

An interesting part of this work [30] is the system-level comparison of performance with the SACCADE, NESTOR, AEG, and IBM systems. Using a small test data set (240 fields with 1492 characters) from IRS forms the author concludes that the KODAK system performed better than the others.

5 Evaluation Studies

This section summarizes the results of three different system evaluations in which the authors were involved. Two of these, the First and Second Census OCR Systems Conferences, were run by NIST for the U.S. Bureau of the Census to ascertain the state of the OCR art with respect to Census-like applications. The third was a comparison of two OCR engines that was commissioned by a company that had recently developed a new OCR engine.

For the purposes this paper, these comparisons can be viewed as tests requiring very different levels of segmentation capability. The First Census OCR Systems Conference required no segmentation capability. The test data for this comparison were isolated handprinted characters that had been segmented from weakly constrained handprint entered onto questionnaires by members of the general population. The only constraint on the handprint was a box into which all of the characters associated with a particular question were to be printed. This was the level of constraint imposed by the type of questionnaire generally used by the Census Bureau at the time of these conferences, so this was the type of task of interest for benchmarking.

The comparison of two OCR engines involved a relatively simple segmentation task. The test data for this task were short strings of handprinted characters, each letter in a separate box. The spacing between boxes was uniform, and was available to the OCR systems to help in the segmentation process. This task is typical of what is being done in commercial applications today.

The Second Census OCR Systems Conference involved a much more difficult segmentation task. The test data for this task were single rectangular boxes containing all of the handprint characters that constituted full answers (words or phrases) for the questions on the questionnaire. The only constraint was the height and width of the rectangle into which the word or phrase answer was to be entered. Since there were no boxes for individual characters, the spacings between the characters, the number of touching and overlapping characters, and the slant of the characters were largely unconstrained. Related discussions on performance metrics for an objective evaluation of automated data-entry system is presented in [33].

5.1 First Census OCR Systems Conference

As mentioned above, the test materials for the First Conference were separate images of single handprinted characters that had been isolated from a sample of (largely) unconstrained answer phrases. The resulting images were checked to assure that they contained single characters that could be identified in context before distribution to the conference participants. A small fraction of the images failed this test and were removed from the test sample. Thus the First Conference was concerned only with the central OCR task, the assignment of the correct ASCII character to the image of an isolated character. At the time of the First Conference, it was widely believed that this problem was essentially solved, and it was desired to verify or refute this belief. The test materials

for the First Conference consisted of about 60,000 digital images of digits, 12,000 lower-case letters, and 12,000 upper-case letters. The digit images were approximately uniformly distributed among the ten digit classes, and the letter images were approximately uniformly distributed among the 26 letter classes. The digits and letters were produced by about 500 different people.

Training materials were provided before the actual test so that Conference participants could adjust their systems to the data formats adopted for the Conference, and train their systems on a large data set, if they did not already have one available. The training materials were similar to the test materials with three exceptions: 1) approximately four times as many images of digits and letters from approximately 2100 writers were provided, in an attempt to provide sufficiently diverse examples for training, 2) different segmentation algorithms were used to produce isolated images of single characters for the training materials than were used for the test materials, and 3) the populations who created the training and test materials were not identical.

The training materials were obtained from 2100 filled-out questionnaires that were returned, out of 3000 questionnaires that were mailed to permanent Census Bureau field employees. The resulting images clearly came from highly motivated people, as demonstrated by the fact that they filled out and returned the questionnaires. The test materials were obtained from questionnaires filled out by students at a local school as a supervised class exercise. The students were probably not as motivated as the Census employees. In fact, a cross-validation study detected a statistically significant difference between the digits and letters printed by the Census employees and those printed by the students. In particular, the training digits turned out to be a subset of the test digits even though there were more of the former. On the other hand, neither the training letters nor the test letters were a subset of each other. Rather, each contained unique letters not contained in the other.

Figures 14, 15, and 16 summarize the results of the First Conference test in terms of error versus rejection curves. These results consist of 118 submissions from the 26 (of 29) participants who provided submissions for scoring. Note that the shapes of these curves are very similar even for systems having very different performances at zero rejection rate. In fact, it is possible to fit all of the error-rate versus rejection-rate curves to a simple model that shows that most of the systems are behaving as if they were Bayesian classifiers deciding between no more than two plausible alternatives [5].

About half of the systems correctly recognized over 95% of the digits, 90% of the upper case letters, and 80% of the lower-case letters. For comparison, a human correctly recognized about 98.5% of the digits. Not all participants used the NIST training materials, but many did. This included some participants who had larger or more general data sets available for training, but chose to use the NIST set to remove training-material quality as a variable in the test. Many of the participants in this category felt that the NIST training materials limited the performance of their systems in the test. Following the test, some of these participants carried out further tests to demonstrate this point. In the meantime, NIST carried out further, but still very limited, studies of human classification of the test images.

The results of some of these studies are summarized in Figures 17, 18, and 19. These figures suggest that human performance is still better on digits and lower-case letters at zero rejection rate, but machine performance can produce significantly lower error rates at commercially viable rejection rates than can human performance, which is not at all suited to sophisticated rejection strategies. In other words, human and machine recognition capabilities have complementary strengths and

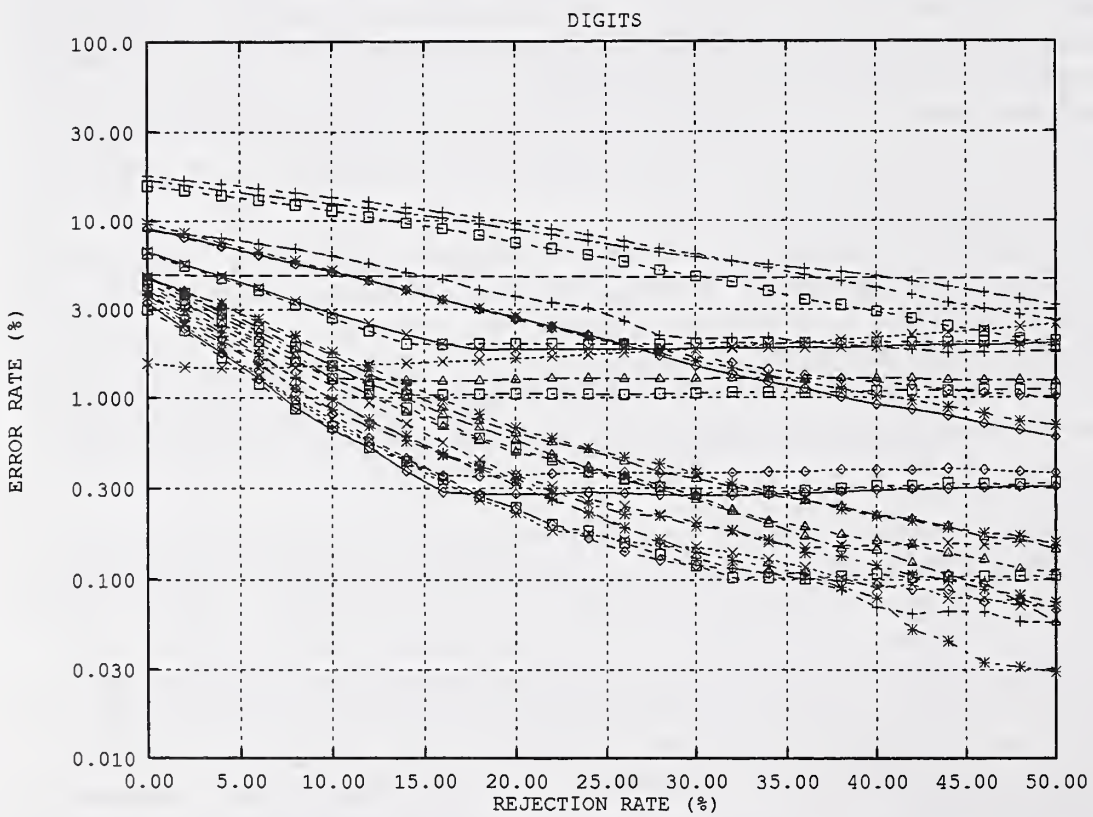


Figure 14: Error rate versus rejection rate for isolated digits for systems in the First OCR Systems Conference.

UPPERS

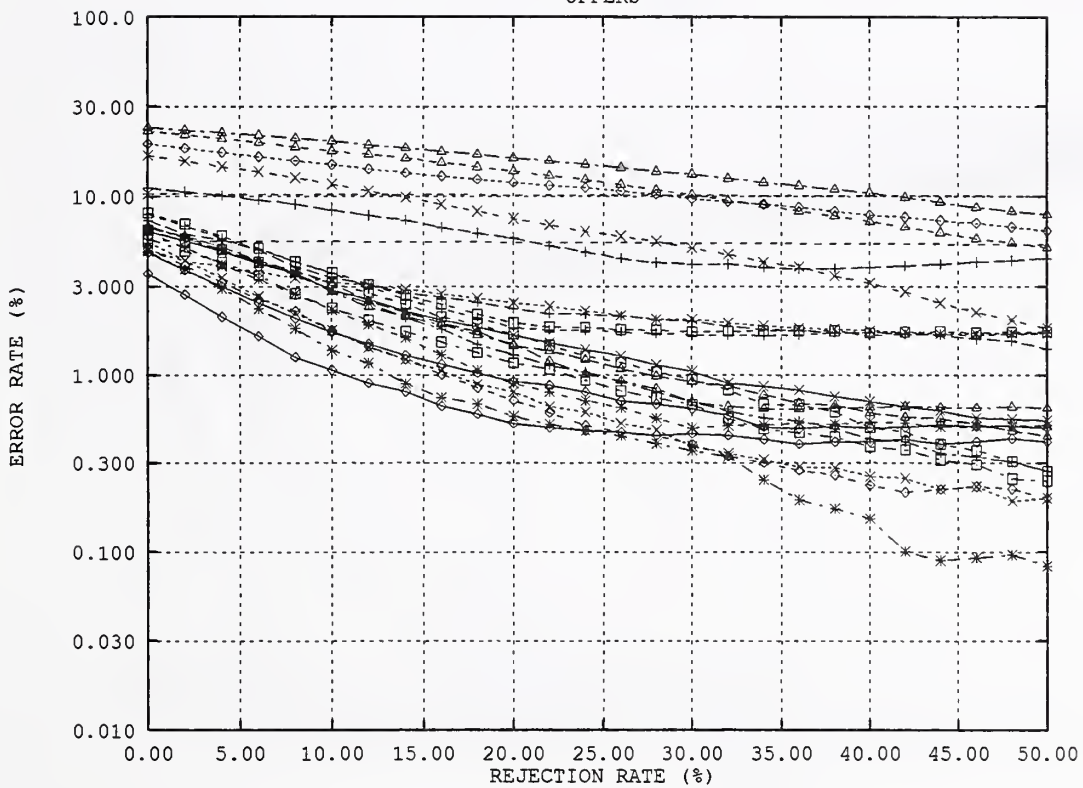


Figure 15: Error rate versus rejection rate for isolated upper-case letters for systems in the First OCR Systems Conference.

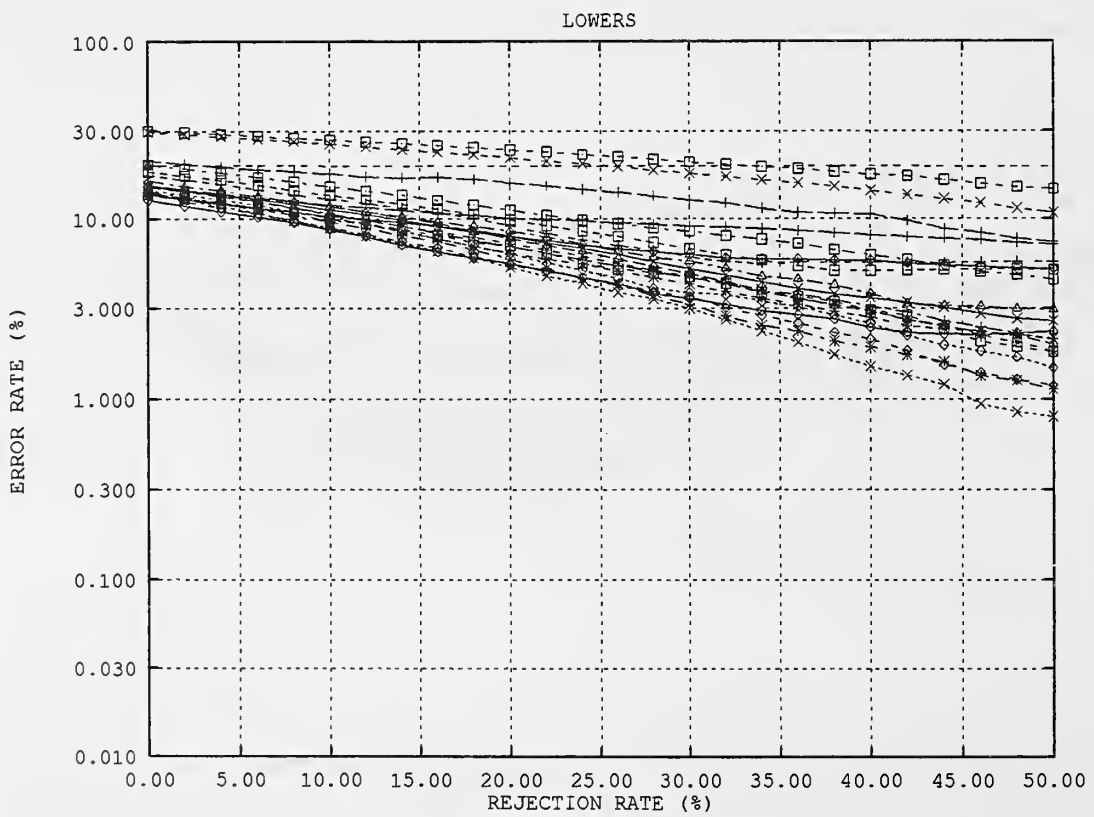


Figure 16: Error rate versus rejection rate for isolated lower-case letters for systems in the First OCR Systems Conference.

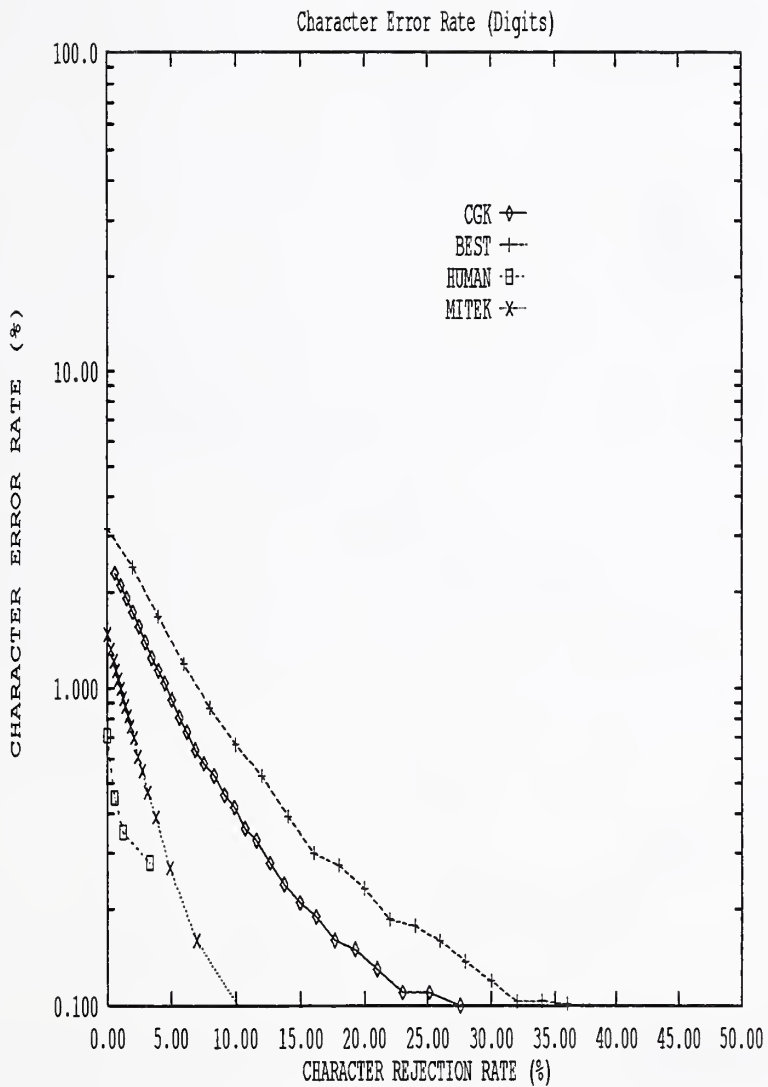


Figure 17: Comparison of human error rate versus rejection rate for isolated digits with that for some system results obtained following the First OCR Systems Conference.

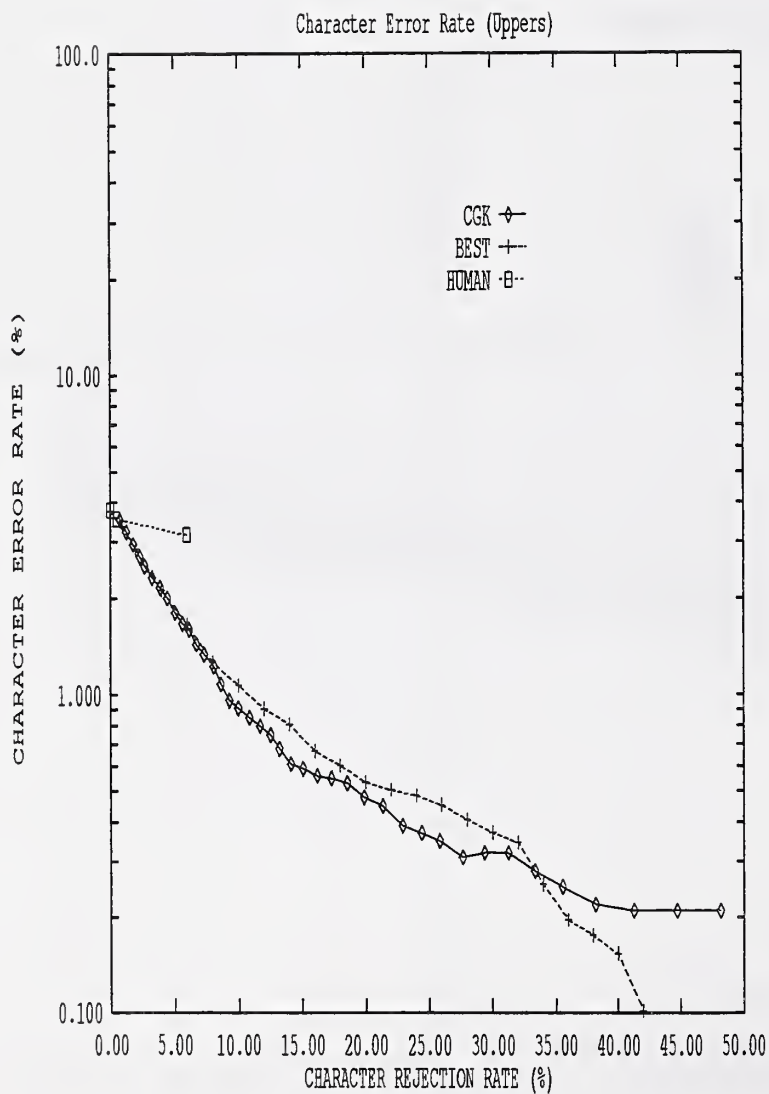


Figure 18: Comparison of human error rate versus rejection rate for isolated upper case letters with that for some system results obtained following the First OCR Systems Conference.

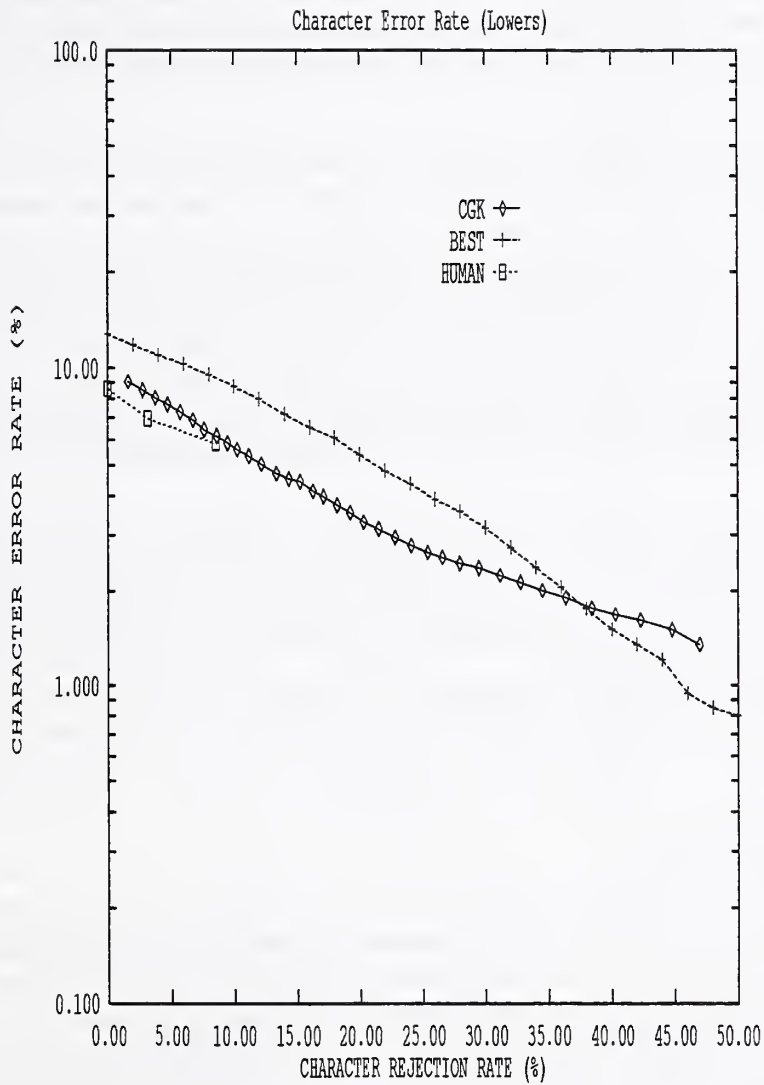


Figure 19: Comparison of human error rate versus rejection rate for isolated lower case letters with that for some system results obtained following the First OCR Systems Conference.

weaknesses, and a system that uses the strengths of one type of recognition to compensate for the weaknesses of the other type can outperform either one alone.

The major conclusions of the first conference were:

- At small rejection rates, many implementations of OCR algorithms behave like Bayesian classifiers that need to choose between no more than two different plausible characters when recognizing images of isolated characters. Therefore, the differences in performance among different OCR algorithms occur primarily in two areas: 1) the fraction of images for which only one character is plausible, and 2) the largest rejection rate for which the choice remains between only two plausible characters. The first difference determines the error rate at zero rejection rate. The second determines the minimum error rate achievable with rejection of some fraction of the images.
- Human and machine recognition of isolated characters have complementary strengths and weaknesses. Human performance, which has access to a hierarchy of capabilities ranging from highly reflexive to highly cognitive, 1) is capable of superior accuracy on the hardest cases and, 2) requires much more time for these cases than for the easiest cases, and 3) will not improve much over the next ten years. Machine performance, which provides a measure of the classification accuracy along with the classification, 1) is capable of higher accuracy with sufficiently high rejection rates, 2) sacrifices very little speed to achieve reliable rejection, 3) is capable of higher sustained speeds over long periods of time, and 4) will improve substantially in all areas of performance during the next ten years due to continually increasing computing power.
- While it is true that machine performance is currently roughly comparable to human performance in the recognition of images of isolated characters, this result misses the point. The key conclusion is that it is possible (in both principle and practice) to combine machine and human OCR capabilities in a hybrid system to achieve both lower error rates and lower cost than can be obtained from a purely human key-entry or a purely machine OCR system. However, a significant re-engineering effort, starting with form redesign, may be necessary to obtain these improvements.

5.2 Evaluation of Two OCR Systems

This section summarizes a comparison of two OCR engines that was commissioned by a company that had recently developed a new OCR engine. Internal studies suggested that the new engine was better than another commercial engine that was well accepted as an industry standard. The company wanted to verify independently that the new product was indeed significantly better in a typical application than the industry standard product. The two products will be referred to as System 1 and System 2 in this section.

Both systems were OCR rather than OWR systems—that is, they produced characters rather than words or phrases as output. The systems were separately compared in the recognition of handprinted digits, uppercase letters, and mixed uppercase and lowercase letters referred to as alpha. As mentioned in the introduction to this section, the test materials for this comparison were typical of the types of applications for which commercial OCR systems are being used, whereas the

other comparisons discussed in this section are not. Furthermore, the comparisons were conducted in a manner as similar as practical to the sorts of tests that might be carried out by a potential customer. Therefore, this comparison illustrates how a potential user might conduct a comparison of two or more OCR software products before replacing a key-entry system with one of them.

Figure 20 shows the form that was used to collect a handprint sample for the comparison. Only the characters above the boxes were used in the comparison. The samples of the other characters were collected in anticipation of future research and comparisons. All future references to the form or form design in this section are restricted to the characters above the boxes.

The particular form design shown in Fig. 20 was chosen to minimize the cost of classification and verification of the handprinted characters entered on the form. Different form designs will be better suited to comparing the performances of OCR engines in different applications. Since application-specific and generic tests have slightly different goals, they impose somewhat different constraints on form design. However, it is very unlikely that the choice of the form adopted for the comparison reported here had a significant effect on the outcome of the comparison since all system were subject to the same form limitations.

Except for the use of a separate box under each character, the form shown in Fig. 20 is not well optimized to improve OCR accuracy. The use of separated boxes and drop-out ink, as well as other measures, could provide substantial improvements in OCR accuracy.

Alphabetical and numerical order were chosen to reduce the likelihood that a person would interchange two characters when filling out the form. This type of handprint error is fairly common, and it is probably more likely with random groups of characters (nonsense words) than with the alphabet, due to alphabet training in school. The alphabet was broken into small groups to separate either the uppercase or the lowercase letters that are the most difficult to tell apart without context clues. It was thought that separating these characters would reduce the chance that a person would interchange them when filling out the form. This grouping should also reduce offset errors, but these are already fairly unlikely on a form with a separate box for each character.

The low probability of offset error means that a fairly good estimate of the classification of each character in each box can be obtained from the location of the box on the form. Since errors are still possible, some sort of verification of the classification assigned by location is still needed. It was decided to carry out the verification as an operator-assisted step during the OCR process.

Volunteers were encouraged to fill out the forms by an incentive. (Enrollment in a drawing with a small but not worthless prize is an example of a good incentive.) A total of 1200 forms were collected. These forms were scanned with a Bell and Howell 2137A scanner to produce 200 by 200 dots per inch (dpi) binary, full-form images.

A subset of the resulting images were removed from the collection by a person who was naive about how OCR works, using software written specifically for this task. The first 700 images remaining after this step were stored in seven directories, 100 images per directory on a personal computer system. These images will be referred to as the test images in the remainder of this section. To restrict the test to segmentation and recognition performance, only one form-removal software package was used, and the same image snippets were sent to both OCR engines for further processing.

During these processes, some characters were lost due to unsatisfactory form removal, and some others due to writer errors and local form-removal errors. Furthermore, some form-removal errors

28368

Hand Print Collection Form

Thank you for participating in our hand print test program. We appreciate your time

Please print each indicated letter in the box below that letter using your normal upper case printing. Please avoid touching the sides of the box.

A B C D E	F G H I	J K L M
<input type="text"/>	<input type="text"/>	<input type="text"/>
N O P Q R	S T U	W X Y Z
<input type="text"/>	<input type="text"/>	<input type="text"/>

Please print each indicated letter in the box below that letter using your normal lower case printing. Please avoid touching the sides of the box.

a b c d e	f g h i	j k l m
<input type="text"/>	<input type="text"/>	<input type="text"/>
n o p q r	s t u	v w x y z
<input type="text"/>	<input type="text"/>	<input type="text"/>

Please print each indicated number, or special character in the box below that character using your normal printed numerals and symbols. Please avoid touching the sides of the box.

. , - () 0 1 2 3 4 5 6 7 8 9 @ \ / ; : ' " ! ? | =

The quick brown fox jumps over the lazy dog @ (1,234.00 - 5,678.90) = 1/2?

Please print the text into the box below it. You may write in more than two lines if necessary. Use your normal writing, with normal word spacing.

Figure 20: A reduced copy of a FAX of the form used to collect the handprint sample for the comparison of two OCR engines.

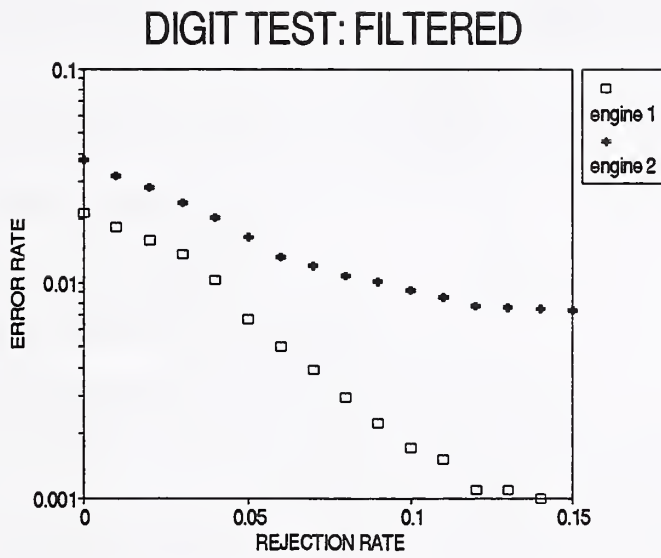


Figure 21: Error rate versus rejection rate for two OCR engines in recognizing a set of 6,422 isolated images, almost all of which contained digits.

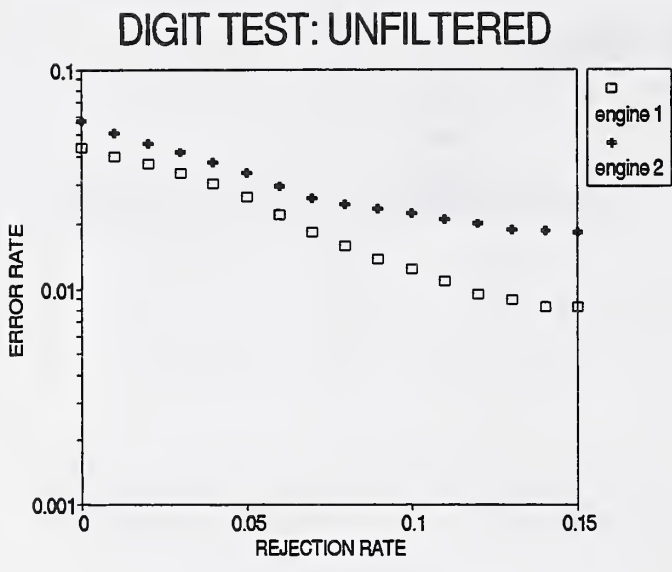


Figure 22: Error rate versus rejection rate for two OCR engines in recognizing a set of 700 images, most of which contained the ten digits, one digit to a box, but about 8 type of non-character information typically found in OCR applications as described in the text.

and some phase errors (see below) in the printing of the digits were discovered during the character validation procedure. In the end, the comparison was run on about 18,000 upper-case letters, 18,000 lower-case letters, and 7,000 handprinted digits printed by 700 different people, and collected specifically for this comparison.

A total of six tests were run. Two tests used both the upper- and lower-case letters, two tests used just the upper-case letters, and two tests used just the digits. One of the digit tests was run with the entire data set, which had a number of non-characters among the images as well as incorrectly classified and ambiguous characters. The other digit test was run with a very clean subset (92%) of the characters used in the first test. The same was the case for the letters tests, but the clean subset contained about 95% of the characters used in the other test.

The Forms Processing (FP) system used in the comparison was configured specifically for these comparisons. It was designed to read hand-print entered in the boxes on the form shown in Fig. 20 to support recognition by both OCR engines, and to support verification of the classification of the character images on the test images. The OCR results were in the form of 700 files. The FP system was run to recognize the characters in the test images with both OCR engines in two separate passes, one for each system. During the second pass, the validation/rejection of any character classified differently by either engine than expected on the basis of location on the form was carried out.

In summary, the FP system was set up to perform the following tasks:

1. select one OCR engine,
2. retrieve a test image,
3. isolate a line from the image,
4. isolate a group of touching boxes from the line,
5. remove the outlines of the boxes, leaving a group of character images,
6. send the group of character images to the appropriate OCR engine for segmentation and recognition, along with the scanner resolution in dots per inch (dpi) and the estimated uniform character spacing in characters per inch (cpi), based on the form design.
7. store the classifications assigned to the character images (hypotheses) by the selected OCR engine,
8. repeat steps 2) through 7) until all 700 test images have been processed,
9. select the second OCR engine
10. repeat steps 2) through 8).

In addition, it was possible to configure the system to carry out the following steps between steps 7) and 8):

- compare the hypotheses from both OCR engines for the test image with the expected classification based on location on the form,

- if either OCR engine produces an unexpected hypothesis for any character image in a group of characters, display the entire group and mark the first discrepancy for the user to resolve either by accepting the original classification based on the location on the form or by flagging the original classification as problematic. If more than one discrepancy is found in a group, continue display of the group after the first discrepancy is resolved, and mark the second discrepancy for resolution. This process continues until all discrepancies have been resolved by the user.

It took about 20 minutes for the system to process 100 forms in TIF format without validation. It took about 1 hour and 15 minutes to process 100 forms with validation. Part of the increase in time was associated with displaying the groups of characters on the screen. The rest was associated with using a hierarchy of human cognitive skills to answer the following questions:

Is the character immediately identifiable in context? If so, is it identified as the expected character? Does it remain so out of context? With what characters might it be confused?

On the basis of this thought process, each marked character was either flagged as problematic or accepted. At least half of the marked letters were accepted, but probably less than three quarters. Very few digits were marked, but a large fraction of those marked were flagged as problematic. This reflects the fact that digits have to be much less ambiguous than letters in order to be useful, due to the different way they are used. Thus different criteria were used in flagging problematic digits and problematic letters. In any case, only a small fraction of the characters were removed in this way as described below, and the tests were scored both with all characters and without the flagged characters.

The verification process described above is important as a check on the validity of the results of the test. It is tempting to assume that whenever one OCR system produces a hypothesis that agrees with the classification based on position on the form, then this hypothesis is correct, even if the other OCR system produces a different hypothesis. This assumption substantially reduces the total number of discrepancies that must be resolved by the evaluator. However, the discrepancies that would be ignored by this procedure are exactly those discrepancies that cause a difference in score between the two systems. Therefore, it was important to verify the classification based on location on the form in just these cases, and all discrepancies were displayed for resolution in the FP system used in this comparison.

On the other hand, over-zealous removal of problematic images may mask real differences in performance between the OCR systems. This would be the case if one system were as accurate as the other in recognizing well-formed characters, but substantially more accurate than the other in recognizing ambiguous characters. Therefore, it is a good idea to flag only obvious errors (such as an O where an I is expected) when resolving discrepancies between the location-based classifications and the hypotheses generated by the OCR systems. As a final check, it is a good idea to score the results with and without the problematic images. Both types of scoring were adopted for this comparison.

The validation process also serves as a check on overall system performance. During the validation process, thirty forms were encountered with major form registration and removal problems that were so severe that the zones sent to the OCR engines for segmentation and removal had no character images at all or a mishmash of box lines and partial character images. Similarly, four instances of phase errors in the number boxes were discovered during validation. In each instance

the person filling out the form entered the digits 1 2 3 4 5 6 7 8 9 in the boxes under the digits 0 1 2 3 4 5 6 7 8. Sometimes the digit 0 (zero) was entered in the last box under the 9; other times the number 10 was entered.

Lower-case letters were accepted during validation when an upper-case letter was requested on the form, provided that the lower case letter was recognizable as the letter requested. The same is true for upper-case letters entered in boxes where lower-case letters were requested. The fact is that many people print this way. Therefore, an OCR engine must be able to handle this situation.

For each image in the test, the FP system produced three files. The first was a key file, the second was a classification file for one of the engines, and the third was a classification file for the other engine. The key file contained 88 lines of data, 52 lines for the 26 upper-case and 26 lower-case letters in the alpha test, 26 lines for the 26 upper-case letters in the upper-case letter test, and 10 lines for the 10 digits in the digit test. (Note that the same 26 upper-case letters from each form were used in the both the upper-case letter test and the alpha test, but that different OCR-engine classification options were used. Thus the 62 character boxes on the form give 88 different classifications for the three tests.) The n th line of the key file contained either the ASCII character expected in the n th box on the form, or a question mark to flag the classification as problematic based on the validation procedure.

Each of the classification files had 88 lines corresponding to the 88 lines in the key file. The n th line of these files had the hypothetical classification (hypothesis) assigned by the OCR engine to the image in the n th box on the form, followed by a confidence value that ranged from 0.00 to 1.00 (two digits to the right of the decimal point).

Each test (alpha, upper-case letter, and digit) was scored for each OCR engine with the unfiltered results and with the filtered results. The scoring was carried out in three operations. In the first operation, four histograms were computed for each test and each engine. The histogram bins were the 101 confidence values 0.00, 0.01, 0.02, ..., 1.00. The first histogram sorted the unfiltered hypotheses by confidence value. The second sorted by confidence value the unfiltered hypotheses that were not identical to the classes assigned according to location on the form. The third and fourth sorted the filtered hypotheses in the same ways. A considerable amount of analysis of data formats and values was carried out during the generation of the histogram data, and exception files were created. The only exceptions discovered by these analyses were the forms having unsatisfactory form removal and the phase errors on the digits. It was decided to remove all characters from these forms (fields in the case of phase error) rather than only those flagged during validation.

The second operation combined the histogram data to produce error-rate versus rejection-rate data for each engine and each test using both unfiltered and filtered results. The third operation interpolated the error-rate versus rejection-rate data at 0.01 intervals of rejection rate for plotting.

A final point about error-rate versus rejection-rate data may be in order. Error-rate versus rejection-rate curves are not continuous, and are uniquely defined only at as many different rejection rates as there are different confidence values. Nevertheless, it is common to interpolate them to uniformly-spaced rejection rate values for comparison purposes as if they were both continuous and unique.

The error-rate versus rejection-rate data had some widely separated rejection-rate points. Usually, interpolation between such widely separated points will produce values that are indicative of what would be obtained in a real application. If, however, the data are inadvertently or purposely sorted by the accuracy of the classification as well as by the confidence value, then very different shapes

of the error-rate versus rejection-rate curves would be obtained wherever the interpolation takes place between widely separated points.

The graphs containing the error-rate versus rejection-rate data were cutoff before interpolation between widely separated points took place. The cutoff for the letter data, which was 40%, was chosen because larger rejection rates are not considered commercially useful. The cutoff for the digit data, which was 15%, was chosen because the curves were quite flat over a large range of rejection rates above 15%, and involved interpolation between only two points over most of this range.

Figures 21 and 22 show two of the six different graphs that were generated to summarize the test results. Figure 21 shows the results for filtered digits. At zero rejection rate the new engine reduces the error rate by almost a factor of 2, while at larger rejection rates the new engine reduces the error rates by as much as a factor of 8. This is the largest difference encountered in the tests. Figure 22 shows the results for the unfiltered digits. For this data set form registration rather than character recognition was the accuracy-limiting step, so the differences between the performances of the two engines was not nearly so dramatic as in Figure 21. This illustrates the importance of locating and improving the accuracy-limiting module when attempting to improve OCR system performance. The relative differences in performance for the upper-case letter and alpha tests fell between the extremes shown in Figs. 21 and 22.

In summary, the new engine produced consistently lower error rates at all relevant rejection rates in all six tests, ranging from 81% of the industry-standard engine's error rate at a rejection rate of 4% on the dirty (unfiltered) digit test to 12% at a rejection rate of 15% on the clean digit test. On the letter tests, the new engine's error rate was typically 50% of that of the industry-standard engine at zero rejection, and an even smaller fraction at larger rejection rates.

Both setting up the FP system and validation of the test data required a substantial amount of work. As it turned out, the bad images flagged in the test data during the validation process had no effect on the conclusions of the test. However, if the performances of the two engines had been more similar, the overall conclusions might have depended upon which data set was used. Depending on the application, either the filtered data or the unfiltered data might be more representative of what would be encountered in normal use of these engines.

5.3 Second Census OCR Systems Conference

The test materials used in this Conference were by far the most difficult of the three tests discussed in this section. Nevertheless, they are typical of the types of applications that many organizations would like to convert to OCR from manual key entry. Two different types of source document were used to create the training and test materials for the Second Conference. For one set, the Industry and Occupation answers were scanned from a sample of microfilm copies of the 1990 Census Long Forms. For the other set, the Industry and Occupation answers were scanned directly from a sample of the 1990 Census Long Forms that had been reserved for this purpose. The image quality of the former was quite poor, while that of the latter, although not ideal, was much better. Only some of the participants attempted to recognize the images scanned from microfilm, and the error rates were higher than for the images scanned from paper, but not much higher in the case of one of the participants. The images scanned from microfilm will not be discussed further here.

The test materials for the Second Conference were mini-forms containing three phrases entered into three answer boxes, supplemented with a dictionary of the ASCII transcriptions of answers that were provided on a sample of similar forms. The resulting images were checked to assure a minimum level of image quality and readability. A small fraction of the images failed this test and were removed for this reason. A small fraction of the images were also removed because these images contained potentially sensitive information. Thus the Second Conference was concerned with the full task of converting unconstrained handprinted phrases (one answer per answer box) into ASCII transcriptions of the answers.

Reference data for scoring was generated by having the same Industry and Occupation answers keyed twice independently, and rekeyed a third time when differences occurred between the results of the first two keyings. This allowed a fair comparison of the machine recognition results with human performance on the same task. The training materials consisted of 6,000 mini-forms with 18,000 answer fields, while the test materials consisted of 3,000 mini-forms with 9,000 answer fields. The average answer field contained about 13 characters.

Two different error measures were employed in scoring the test. The first, the field error rate, scored a field as incorrect if it had a single error, and reported the ratio of the number of incorrect answers to the total number of answers. The second, the field distance rate, aligned each test answer with the corresponding reference answer, summed the number of deletion, insertion, and substitution errors, and reported the ratio of the sum to the total number of characters in the reference answer. The field-distance rate reduces to the character error rate when there are no deletion or insertion errors.

This test was much more difficult than the character recognition test at the First Conference. Of over 25 participants, only 10 provided a total of 31 submissions for scoring, and of these, only 14 were provided on time. The remainder were provided within an extended time period set up to allow the completion and submission of results that were only partially complete by the end of the originally allowed test period. Figures 23 and 24 show the field error and distance rates for the on-time submissions. None of the late submissions were better than the best on-time submissions, but many were significant improvements over the on-time submissions of the same participant. This is another illustration of the difficulty of the test at the current state of the art.

The field error rates shown in Fig. 24 are much larger than the character error rates obtained in the First Conference. The best machine system in the conference does not reach the 8.5% field error rate or the 1.6% field distance rate at zero rejection rate that was achieved by the human keyers working in the 1990 Census until over 45% of the fields have been rejected. This is very different from the results of the First Conference where the best system achieved the zero-rejection-rate error rate of a human reference at a rejection rate of less than 10%. Furthermore, a number of better results were submitted to NIST for scoring within less than two years following the First Conference. On the other hand, no better results for the Second Conference have so far been submitted to NIST for scoring. Finally, a great number of systems in the First Conference produced results that were roughly comparable to the best results from that conference. This was not true for the results of the Second Conference, where greater differences in performance among the participants were evident.

The main reason for these differences between the results of the First and Second Conferences was the difference between the tasks. For the Second Conference, it was necessary to segment multi-character images into single, isolated character images before they could be recognized by the same

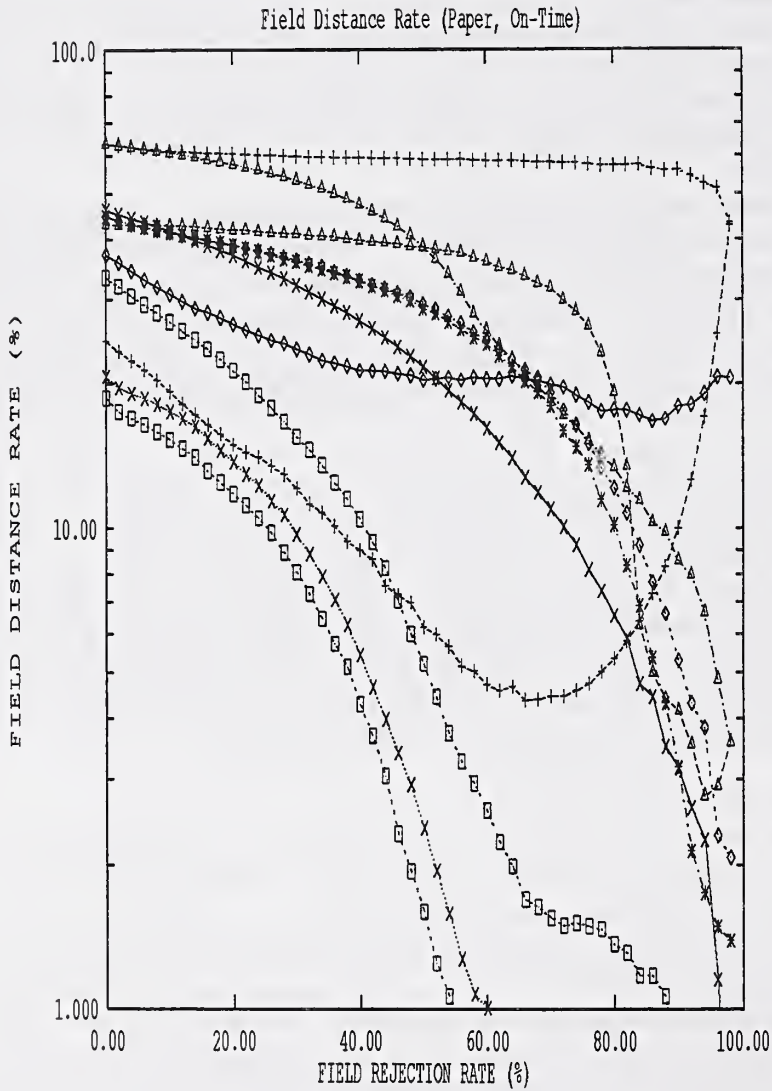


Figure 23: Field error rate versus rejection rate for systems in the Second OCR Systems Conference. See [3] for a detailed explanation

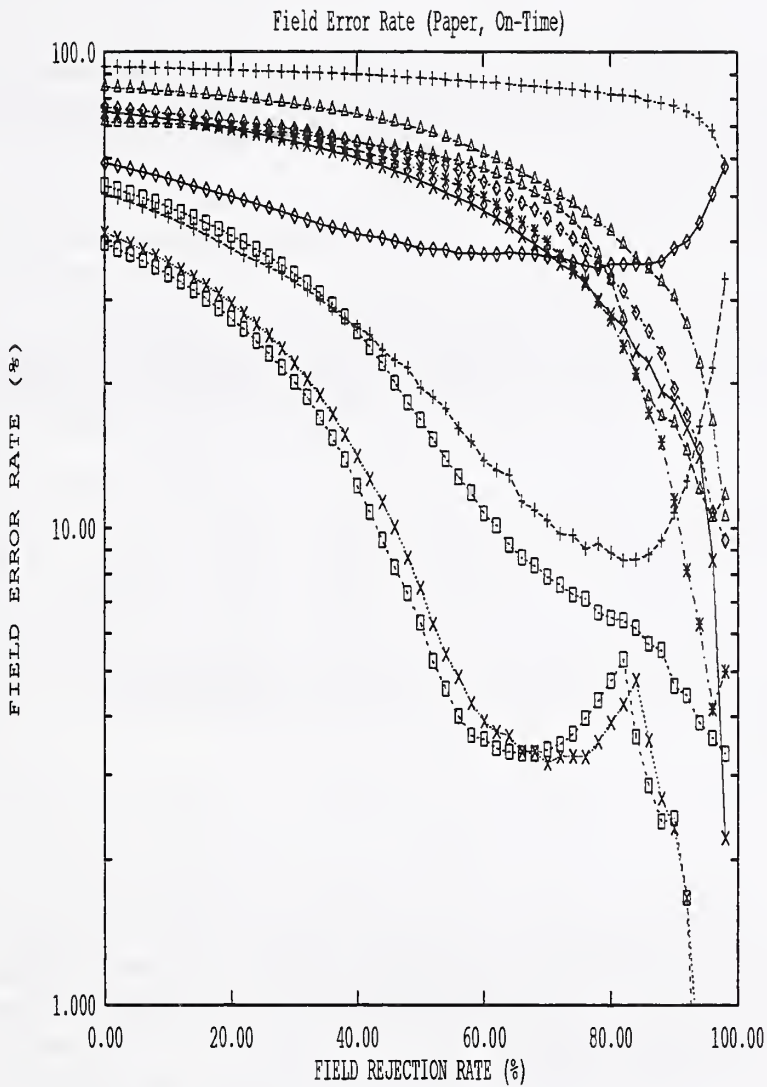


Figure 24: Field distance rate versus rejection rate for systems in the Second OCR Systems Conference. See [3] for a detailed explanation

or similar algorithms used in the First Conference. All of the participants in the Conference found this to be the accuracy-limiting step, even with the help of dictionaries of likely answers. All but one of the participants accepted a great deal of over-segmentation (breaking the images of single characters into images of pieces of characters) in order to avoid under-segmentation (combining multiple characters into a single image). All of these participants had to then reconstruct or recognize single characters from the pieces. The success in carrying out these two operations was the single most important determinant of the error rate.

The major conclusions of the Second Conference, which are quite different in tone from those of the First Conference, are the following:

- The accuracy-limiting step in the recognition of unconstrained handprinted phrases is the segmentation of each phrase into isolated characters. This task is apparently so much more difficult than the other OCR-system tasks that it can wash out differences in the accuracy with which the other tasks are carried out.
- Dictionaries and language models can be used to assist in the correct segmentation of unconstrained handprint.
- Humans are still significantly better at the recognition of unconstrained handprint than are machines, but the best machine performance is close enough to human performance to suggest that it is only a matter of time before a hybrid system can outperform either type of pure system.

6 Summary and Conclusions

Four categories of handprint are relevant to the design and evaluation of OCR systems:

- i) one character (digit or letter) per box
- ii) one multi-digit number per box
- iii) one multi-letter phrase per box
- iv) unconstrained

Separate comparison and evaluation of isolated OCR modules (subsystems) would be convenient but misleading. There are many algorithms which can perform these isolated tasks satisfactorily, when evaluated in isolation, but which are not the best components to use in OCR systems. The proper question is which customized combination of modules is best for a given application. It is not even possible to clearly define what might be meant by the best module, because a module that is best in an OCR system for one application is not necessarily best for a different application.

Interactions among components are extremely important to the successful design of an OCR system. Simply linking together modules developed by different vendors without considering these interactions is unlikely to result in an optimal system. Furthermore, prior estimates of recognition rates based on tests of the isolated components are likely to be overstated, because isolated tests are necessarily based on idealized input that is likely to differ from the kind of input that will be produced by earlier modules in the real system. This was one of the reasons for the failure of the

IRS Document Processing System. By contrast, the Census Bureau has recognized the importance of interactions, and this is a major reason for the anticipated success of their OCR efforts.

Analysis of interactions is facilitated by partitioning the output of each module into at least three distinct categories: "correct", "incorrect", and "incorrect, but it doesn't matter". The designation depends on the effect on the performance of the next module. "Correct" means that the module made the correct decision, which cannot hurt the next module's performance. "Incorrect" means that the module's output deviates from the correct output, and further, that this caused the next module to produce an incorrect result, whereas the next module would have produced a correct result had the first module's output been correct. "Incorrect, but it doesn't matter" means that the module did not make the correct decision, but that this did not affect the correctness of the next module's result, one way or the other. Without this kind of analysis, it is impossible to predict system performance from module performance, and it is quite possible for system performance to decline when a module is replaced by another module that has a higher isolated performance score.

1

Segmentation is the major obstacle to improved performance of OCR systems, with the difficulty of segmentation varying enormously over the four categories. For applications in the first category, segmentation is relatively easy, particularly if the size and spacing of boxes are uniform. For the last two categories, however, segmentation is very difficult, and sufficiently difficult that the limited accuracy of the segmentation step, more than any other factor, is responsible for limitations in the recognition capabilities of OCR systems for applications in these categories. In fact, feedback from other modules is needed before useful segmentation results can be obtained.

There are two distinct ways of utilizing feedback for segmentation of handprint in categories (iii) and (iv). Under the first approach, segmentation windows of different widths are used, starting from the current position, with a specified minimum width, and expanding towards the right, up to a specified maximum width. Recognition is performed for each window, and the result with the highest confidence level is chosen. Processing continues from the right side of the window associated with the chosen result. Under the second approach, character segmentation is performed first, followed by recognition. The system then searches through phrases in a dictionary of expected phrases, or uses a more sophisticated language model in order to find the best match.²

Machines and humans have complementary strengths and weaknesses. Machines can work continuously and indefatigably at full speed, and can also provide an estimate of correctness probability as part of the recognition process. On the other hand, machines cannot process the most difficult fraction of the workload accurately, and cannot handle unexpected data. The major strength of humans is the ability to utilize a hierarchy of recognition procedures, ranging from highly reflexive to highly cognitive. They may also be able to handle unexpected data. On the other hand, humans display rapid fatigue when functioning at high speed, and experience boredom when operating at low speed, in both cases with consequent deterioration of performance. They are also incapable of providing estimates of correctness probabilities. Further, processing speed is rather slow for the more cognitive recognition procedures.

OCR systems designed to take advantage of the complementary abilities of humans and machines

¹Theoretically, one might also consider making a complete partition of the results produced by an OCR module, and then using this exhaustive analysis to evaluate the performance of a system based on the components. This is impractical because the types of errors that a module can make vary continuously over a multi-dimensional spectrum, and also evidence statistical variations across the spectrum.

²The ERIM system combines both of these approaches

are capable of attaining higher accuracy than systems that only use one or the other. In some applications, such as remittance processing, the improved accuracy has been accompanied by a reduction in total system cost. Presently, it would not be reasonable to attempt to remove humans completely from the most difficult and demanding of OCR tasks. It is reasonable, however, to design OCR systems that require fewer human workers by changing their role from key entry to checking and correcting the small fraction of the data that the machine cannot recognize.

7 Appendix: Literature Review

Up to now this report has focused on a few systems. This appendix provides a review of a number of different algorithms, some of which were used in the systems reviewed here and many of which were not. As pointed out earlier, many different algorithms can be used to successfully solve the various problems encountered in designing an end-to-end OCR system. In this review of relevant literature, we focus mostly on work reported since the publication of the special issue in the Proceedings of The IEEE [34].

7.1 Preprocessing

Typical preprocessing techniques include noise removal, form identification, form registration, skew detection, size normalization, and thinning algorithms. Of these only skew detection and form registration are discussed here. The other preprocessing techniques mentioned above have been discussed extensively in the literature on document processing and understanding. A comprehensive review of all of the relevant preprocessing techniques is beyond the scope of this paper. Typical approaches used for some of these problems are outlined in the context of designing a form-based OCR system in Section 4.3. A detailed survey of thinning algorithms can be found in [35]. A methodology for comparing thinning algorithms using human subjects is discussed in [36]. A more recent paper characterizing the performance of the thinning algorithms is [37]. For a comparison of normalization methods for character recognition, the reader is referred to [38].

Registration: Form registration is the process of registering the form to be processed with a blank form. Subsequent to this process, pixels corresponding to the form's boxes and instructions are removed, leaving the hand-printed data in each field of the image. In form processing applications, the registration problem has been treated as a 2-D problem, with the goal of estimating the translation along the x and y directions, the rotation angle, the skew and the scale difference between the input and reference forms.

Earlier approaches to skew detection have relied upon Hough transform techniques [39, 40]. Baird's [39] approach projects the bottoms of the bounding boxes of connected components parallel to a set of angles, and chooses the angle which maximizes a measure of the variance of the counts in the projection. Measures such as local region complexity [41] and cross-correlation between two lines at a fixed distance [42] have been used for skew estimation. In the latter work the shift for which the cross-correlation measure accumulated over all pairs of lines in the image (separated by a fixed distance) is a maximum determines the skew angle. A novel application of an array processing technique for the angle of arrival problem [43] is used for skew estimation in [44]

The 2-D approach to the estimation of translation, scale, and rotation is outlined in [8]. Feature points needed for registration are located using spatial histogram projections, where spatial histograms are constructed using black pixel densities summed across an image region in either the vertical or horizontal directions. The transformation parameters are estimated by applying least-squares techniques to the required minimum of three points. Additional feature points that are within the respecified regions of points projected using the computed transformation are used to refine the transformation. After the input form is transformed to the blank form, pixels comprising the blank form are removed, leaving only the hand-printed material for subsequent analysis.

By establishing point correspondences between five points in the reference and input forms using methods from projective geometry, the 2-D transformation between the two forms is computed [45]. The feature points selected for establishing the correspondence do not depend on any special markings.

Often, in addition to global warping, one may have to deskew sections inside the document. Although this is an unlikely situation for forms, global as well as local skew detection can be done as described in [46]. This method first estimates and then corrects the global page skew using a least-squares technique. The corrected page is then divided into blocks using a fractal-based segmentation technique. The text blocks are skew-corrected using the least-squares algorithm.

7.2 Character Segmentation

This sub-area is probably one of the most studied ones in OCR. From low-level, model-free thresholding and thinning techniques to the more recent NN-based approaches, hundreds of papers have been published and several modules have also been built. Recent attempts formulate and solve segmentation and recognition in an integrated manner. Earlier attempts treated segmentation as an isolated problem. This strategy worked reasonably well for handprinted digit recognition; in handprint and cursive writing most of the errors in character recognition can be attributed to poor segmentation. Feedback from character segmentation algorithms is almost always useful for improving the quality of segmentation and consequently that of recognition.

While surveying the state of the art in character segmentation, it became obvious that this sub-area deserves a detailed paper in its own right. Due to space constraints and a focus on the hand-printed and cursive OCR problem, we will not present a detailed assessment of the state of the art in character segmentation. In particular, we do not discuss algorithms for the segmentation of isolated, machine-printed characters. Detailed surveys of these and other related issues can be found in [47]. As in many other subareas of OCR (and image analysis in general), metrics and performance measures for the evaluation of segmentation algorithms are sorely lacking.

Segmentation methods for handwritten digits and words can be divided into three groups: a) segmentation of handprinted words, b) segmentation of handwritten numerals as in zip codes and bank checks, and c) segmentation of handwritten cursive words. In the first group at-least three possibilities arise: boxed characters, spaced characters, and touching characters. Of these three, the last one, segmentation of touching characters, is the challenging one. Segmentation methods developed for the machine-printed OCR problem work quite well, with minor modifications in the first two cases.

Following [48], strategies in character segmentation can be divided into three groups: a) In the classical approach, the segments are “dissected” into meaningful components. b) In recognition-based

segmentation, segmentation algorithms produce tentative segmentations which are then presented to recognition algorithms for verification. The interactions between the segmentation and recognition modules are so closely coupled, that these techniques are often called “segmentation-free” methods. c) The last group is the holistic approach, where segmentation of individual characters is not attempted, but the whole word is viewed as the character to be segmented. A purely bottom-up approach to character segmentation will not scale up to real applications. Introduction of context in the form of post-processing, verification, etc. is crucial. Such contextual information is used in different ways in each of the groups described above.

Although techniques that rely on attributes such as white space and pitch do not work for the cursive character segmentation problem, other techniques such as (vertical) projection analysis have been attempted for hand-printed characters such as addresses, zip codes, etc. In general, 1-D approaches based on projections can at best serve as initial conditions, even for handprinted isolated character segmentation problems. Their use for cursive handwritten character segmentation is very doubtful.

Segmentation of handwritten numerals is an easier task than segmentation of handwritten cursive words. In situation involving zip codes, one can use the dictionary of valid zip codes, or derive context-based rules to refine segmentation. Some of the heuristic algorithms for the segmentation of handwritten numerals and zip codes are reported in [49, 50, 51, 52].

One of the popular tools in the category of dissection methods is connected component analysis, which was popular in the image analysis literature during the early seventies. Since algorithms for connected components operate in a bottom-up manner, several post-processing strategies are required to extract “character-like” segments. Some post-processing techniques incorporate context. One of the simplest post-processing techniques is the so-called bounding box [53] technique, where assumptions about the expected number of characters, which of the characters appear in lower/upper cases or joined, etc. are used for cleaning the results of the connected components algorithms. The nature of these assumptions makes this approach useful in limited handprinted digit recognition problem (as in the case of postal addresses).

Segmentation of handwritten, cursive characters poses many challenges, one of the most difficult being the segmentation of touching or merged characters. Both rule-based and NN approaches have been suggested for this problem. Several examples [54, 55] of bottom-up approaches to segmentation of cursive handwritten characters can be found in the literature. These approaches feed the results of the segmentation algorithm to recognition algorithms, but no attempts are made to refine the segmentation using the recognition results. Typically these algorithms have pre- and post-segmentation steps. The goal of the pre-segmentation step is to generate potential breakpoints corresponding to characters and partial characters. All the possible segmentations of the word are evaluated in the post-segmentation step using heuristic rules or a Viterbi algorithm. Pre-segmentation is typically achieved using connected components algorithms or singularities and regularities [56].

Recent examples of rule-based methods for segmenting cursive handwriting may be found in [54]. A set of heuristic rules based on associations between specific geometric and topological attributes and English-language characters is used. These attributes include geometric shapes such as horizontal or vertical bars or loops. Topological attributes are defined using end, branch, crossing, convex and concave points. The alphabet is divided into six classes, each class being characterized using rules combining the geometric and topological features. Limited experimental results are reported. A

correspondence-based approach using the maximum and minimum excursions of the handwriting image is suggested in [55] for the cursive word segmentation problem. Once correspondences are established, ascenders, descenders and loops are identified. A set of heuristic rules is then used for locating coarse segmentation points.

Another technique for segmenting touching hand-printed words relies on locating concavities at touching points. Of the many possible segmentations that are hypothesized by selecting any subset of these touching points, optimal segmentation is achieved using a classifier to evaluate the different segmentation hypotheses.

The second class of methods realizes the importance of integrating the segmentation and recognition steps. Methods based on search, hidden Markov models, and relaxation procedures have been attempted. Search methods divide the image into many overlapping segments. These segments are passed on to a recognizer for confirmation. Many of the earlier attempts utilized an image analysis approach. Recently, NN approaches have been taken. We will discuss these techniques in Section 7.4. A more holistic strategy is to combine segmentation and recognition at the word level. Many of these techniques have their origins in speech recognition and differ mainly in the way comparisons are made between word hypotheses and reference words. The use of hidden Markov models and dynamic programming techniques are common. A recent overview of segmentation techniques for handwritten words can be found [57].

7.3 Feature Extraction

The feature extraction stage is an important component of any recognition system. It is also very much dependent on the task, input, and recognition algorithm to be used. In OCR applications, the types of features extracted depend on whether gray-scale, binary, or vector representations are used. If statistical pattern recognizers are used, then the standard features tend to be based on contours of characters, with features such as Fourier descriptors, moment invariants, and other boundary features. If structural recognizers are used, features that represent the structure of the character are preferred. When NN's are used, the feature extraction stage has gone through some evolutionary developments. Earlier NN approaches used local feature maps. Methods developed later side-stepped the issue of feature extraction, and used the segmented character or digit image as the input. This works well for the case of isolated digits or characters. When touching characters or digits are present, the input image (corresponding to a Zip code, handprinted or cursive word) is presented as such. One of the important considerations in the feature extraction stage is invariance with respect to shape transformations such as translation, rotation, and dilation. Some features are designed to be insensitive to these transformations, while in other cases transformations of the extracted features achieve invariance to some of the shape transformations. Another consideration is whether the original patterns (digits or characters) can be reconstructed. Global descriptions such as Fourier descriptors, moments and coefficients of circular autoregressive models can reconstruct the original patterns, the quality of reconstruction being dependent on the number of coefficients or parameters retained. Reconstructability is easy to achieve in the case of isolated digits or characters but can be quite hard for cursive handwriting

A recent survey of most of the feature extraction techniques for OCR can be found [58]. Techniques for feature extraction from gray-scale and binary images, binary contours and vector representations such as skeletons are described in detail. For gray-scale images, methods based on template matching, deformable templates, unitary image transforms, zoning, geometric moment invariants, and

Zernike moments are surveyed. In addition to template matching, unitary image transforms, zoning, and geometric moment invariants, projection histograms are also discussed as features derived from binary images. For the case of binary contours, contour profiles, elliptic and other Fourier descriptors, moment invariants, and spline representations are reviewed. Lastly, when patterns are represented in vector form, character graphs (generated by the topographical primal sketch), graph descriptions, zoning, and discrete features such as the number of T-joints, X-joints, bend points, etc. are reviewed. As this article surveys most of the feature extraction methodologies that are useful for isolated hand-printed digits and characters, we decided not to review them here. Instead, we will make brief remarks about feature extraction techniques that are not included in [58].

The image analysis approach to OCR has traditionally relied on classification or matching of specific features such as those mentioned above. On the other hand, NN approaches seem to have shied away from the need to define specific features. They have rather used windows of raw pixels or global transforms such as principal component analysis [59] or the projection on to a Gabor basis set. Features derived from principal components have been used in the NIST form-based recognition system [8, 59]. Projections on the Gabor basis set have been used in the NIST [60] as well as the KODAK system [61]. In [61] the utility of global transforms such as Gabor, Fourier, and Cosine has been evaluated as a source of features for a standard NN recognizer of isolated handprinted characters. Experimental results indicate that when windows of raw image pixels are replaced by projections on the Gabor basis set, significant reductions in errors have been observed. An NN recognizer using Gabor basis set projections was evaluated in the First OCR System Conference and was one of a tight group of leaders. This feature set was also used in the KODAK system reported in Section 4.3. Another trend in NN approaches, is to avoid deciding any sort of features explicitly, but to let the NN determine the appropriate features in an autonomous manner. The convolutional NN (CNN) developed by researchers at AT&T is a good example of this approach [62].

Deformable spline models whose shapes are determined by the positions of eight control points have been suggested as features [63] for handprinted digit recognition in Zip codes. Shapes are deformed when the control points wander. An energy function that penalizes shape deformations due to displacement of the control points is minimized. The fitted models are then used for recognition by deciding on the model of the digit that could have generated the data.

As noted in [58], deformable templates have been used for feature extraction for both binary and gray-scale images. One of the vexing problems of fitting deformable templates is initialization of the parameters characterizing the surface. In [64], an NN is used to learn the associations between images and the instantiation parameters of deformable templates that characterize these images. A 3-layer NN was trained using the BP technique and the CEDAR CDROM and database. The trained NN was used to provide better initial conditions for the new images.

7.4 Recognition

7.4.1 Statistical Techniques

Techniques that fall under this category typically extract numerical features such as Fourier descriptors [65, 66, 67], moments [68, 69, 70], and other global descriptors [71, 72] and use any of the existing statistical pattern recognition techniques. In the early seventies, much of the reported work

on shape recognition using the statistical approach was geared to isolated machine-or hand-printed digits and alphabets. Some of the global descriptors are sensitive to one or more of the typical shape transformations, such as translation, tilt, dilation, etc., while others are insensitive to these transformations.

The success of statistical methods very much depends on the existence of a perfect segmentor. In the case of touching/broken characters, there is no simple way to relate the features extracted from the forms to the training samples. A recent comparison [73] of classical classifiers such as the linear and k-nearest neighbor classifiers and several NN based classifiers developed at AT&T using a modified NIST dataset of isolated digits has shown that NN classifiers and hybrid NN and nearest neighbor classifiers perform much better. One would extrapolate that the performance improvement would be even greater when connected digitis and/or words are used for comparison since the classical statistical approaches treat segmentation/feature extraction as an independent stage from classification. As discussed later, the ISR approach which is natural for NN based approaches are more appropriate for these difficult cases. The NN techniques developed at AT&T are reviewed in Section 7.4.3.

Similar conclusions are drawn about the nearest-neighbor rule based approach reported in [74]. By increasing the size of the training set, nearest-neighbor techniques using simple metrics have been shown to be as good as the best system reported in the NIST First OCR Conference [1] where isolated character recognition systems were evaluated. The utility of these approaches for the general cursive handprint problem is doubtful.

Over the last ten years, traditional statistical pattern recognition schemes have been superseded by hidden Markov model (HMM)-based techniques [75, 76, 77, 78, 79]. The premise of HMM approaches is to avoid individual character recognition, but to perform recognition at the word level using extensive contextual knowledge in the form of transition probabilities.

7.4.2 Image Analysis Techniques

The image analysis approach to the OCR problem is based on four major steps: preprocessing, feature extraction, description, and classification. Major techniques used have relied upon template matching, graph and subgraph matching, and correspondence matching. Features used include binary templates, polygonal approximations of skeletons, their locations and associated orientations, strokes or arcs, topological and geometric features, morphological features, and ascenders and descenders. There are differences among the techniques in the matching stage.

A two-stage binary template matching algorithm using two similarity measures known as the Jac-card and Yule measures has been used for the handprinted digit recognition problem in [80]. The similarity measures are used to create templates from the training samples using a sequential clustering procedure. Typically, several templates are stored for each digit to account for inter-digit variations. Since recognition of the testing samples is accomplished through matching with stored templates, the success of this system depends on how well the templates represent inter-digit variations, how many of them are needed, and the numerical values of thresholds. In the two-stage method proposed in [80] the errors made by the template matching step in the first stage are used to refine the templates used in the second stage. This approach works well for isolated digits, but has problems with touching digits. One may partially overcome this by increasing the number of

templates characterizing the different ways the digits touch; this may require a prohibitively large number of templates, often leading to a decrease in recognition accuracy.

One of the recurring problems in any template matching approach is the need to reduce the number of templates stored. In the context of a feature-based system, an indexing approach that uses a coarse description as a pre-screener is proposed in [81]. Polygonal approximations of characters are decomposed using a curve fitting method to produce meaningful parts, which are termed super-features. Super-features are used in a pre-classifier that decides which of the more detailed prototypes should be used for final recognition. Results with USPS data demonstrate the effectiveness of this approach.

A scaled-down version of an object recognition approach used in the computer vision literature is applied to the recognition of handprinted digits in [82]. Using the locations and associated orientations of the segments of polygonal approximations of the digits, the outputs of a k -nearest-neighbor rule are used to build a decision tree classifier. One of the features of this work is the use of decision trees for automatically setting the thresholds needed in an optimal boundary error matching algorithm. Extensive experiments with the NIST and CEDAR databases produced impressive results; the failures are almost always due to poor segmentation. A similar alignment-based approach to this problem was previously described in [83].

An interesting approach to recognition of handprinted alpha-numerics using the “recognition by parts” paradigm [74, 84] is described in [85]. Eight or nine variations of 36 alpha-numerics are decomposed into 2 or 4 or 6 parts. The recognition rates achieved using a computer algorithm are evaluated in comparison to humans’ performance [86]. As part of the analysis, crucial parts that by themselves could produce 100% recognition for alpha-numerics are identified. Crucial parts could provide useful clues for feature extraction. A better understanding of recognition by parts may yield promising techniques for the touching/broken character problem, where parts can be more reliably extracted.

The image analysis approach to off-line cursive recognition has a long history of basic and applied research. In the recent past, applications of ideas pursued by computer vision researchers working on object recognition have made an impact on this difficult problem. A key feature in all cursive recognition systems is the lexicon or dictionary to provide top-level constraints in the post-processing stage. Since even the best segmentors fail to produce acceptable results for cursive, handwritten text, many possible character streams are produced; later, matching constraints provided by the lexicon are used. The dominant strategies used are some form of Levenshtein distance based matching, or more recently, hidden Markov models. The NN-based approaches described later also use a dictionary to provide high-level constraints.

One of the standard paradigms used in object recognition is the so-called hypothesis verification paradigm, where based on matching of a partial set of image and object features, the set of possible objects is hypothesized. Subsequently, the remaining features, extracted with guidance from the chosen models, are matched to find the best-matching object. In the same vein, Favata and Srihari [87] propose a generate-and-test paradigm for cursive word recognition. The generate phase of the system segments the cursive words into characters using one of uniform, ligature, or heuristic segmentation technique. The uniform segmentation technique generates cut points at fixed intervals, while the ligature method finds the relatively horizontal strokes in the word. The heuristic technique combines the uniform and ligature techniques. In the test paradigm a two-step procedure is used. In the first step, a string matching algorithm with a modified Levenshtein metric

is used to find the match between a word in the lexicon and the interpretations generated by the segmentation step, subject to bounds on the number of editing operations (insertions, deletions and substitutions). After the best pairing is accomplished, the second stage verifies the local spatial consistency between each sequential pair in the lexicon word and the interpretation. The character recognition system used in the test stage uses eight normalized features that include the entire character, character holes, vertical and horizontal strokes, and concavities pointing up, down, left and right. Matching involves computing an evaluation function that measures the similarities between the features extracted from the unknown character and those in the training set. This system has been demonstrated on a database of handwritten city names extracted from addresses in envelopes, using a dictionary containing 271 words. A generalized Demerou-Levenshtein distance measure that includes merges, splits and visually pleasing edit operations is presented in [88]. Adaptation of the system developed for postal applications to reading names and address blocks in IRS tax forms is described in [89]. Using a loosely coupled multiprocessor architecture upto 8500 forms per hour can be processed.

In [90], researchers at the University of Michigan, Dearborn reported the design and evaluation of a system that recognizes handwritten words and digits in the context of USPS applications. Names of cities, streets, states, persons and businesses are represented in the test. Writing styles range from strictly printed to strictly cursive. Three lexicons of sizes 10, 100, and 1000 are associated with each word in the testing database. The preprocessing steps include standard techniques for binarization and slant correction. Initial pre-segmentation points (PSP's) are selected at or near valley points of the upper contour of the preprocessed image. A connected component algorithm followed by bounding box detection determines primitive segments. Using a lexicon-driven dynamic programming technique, the segments created by the PSP cuts are merged and matched so that a character likelihood measure is maximized. Using a data set of about 3000 word images, a careful evaluation of recognition results pointed to problems due to poor pre-segmentation and incorrect estimates of word length. After improvements in these areas and a redefined character likelihood measure, better recognition accuracies were reported using a much larger dataset.

7.4.3 Neural Network Techniques

Since the late 1980's, applications of NNs to recognition of handprinted digits, characters and cursive handwriting has become a very active area. Earlier attempts concentrated on recognition of handprinted, isolated digits and characters, with more recent efforts looking at ISR methods.

One of the earlier projects that explored the applicability of NNs along with classical recognition methods such as Parzen window and k -nearest-neighbor techniques is reported in [62]. The problem considered is that of recognizing handwritten Zip code digits. After pre-processing steps such as skew correction and skeletonization, about 49 convolutional masks are used for feature extraction, focusing on lines and line ends. Examples of typical features include horizontal strokes, right-hand ends of horizontal strokes, etc. These 49 feature outputs are combined (subsets are combined using logical AND or OR) to produce 18 feature maps. These feature maps are input to a Parzen window, a k -nearest-neighbor classifier, and a two-stage NN. Although initial results did not show improved performance using the NN, a finely-tuned preprocessor and a large training set enabled the NN to exceed the performance of the statistical pattern recognition techniques mentioned above. In follow-up work [91], a multi-layer network with back propagation (BP) training has been used for the handwritten digit recognition problem. Using a combination of alternating hidden and

averaging/sampling layers and local convolution feature detectors in the hidden layers improved classification rates are reported. One of the issues in situations involving multiple layers is the need to keep the number of free parameters to a minimum. Using the concept of weight sharing [92], which in effect imposes shift invariance on the feature maps, the number of free parameters is reduced. The effects of weight pruning using principal component analysis (PCA), optimal brain damage (OBD) [93], and weight decay are reported in [94] for a modest data set consisting of handwritten digits. The effects of smoothing followed by regularization, as well as using higher-order networks, have also been studied. All of these embellishments yield improved performance.

Instead of the local feature maps used in [95, 62], a holistic approach that uses PCA to generate representations for handwritten digit images is reported in [8]. This can be viewed as an “eigen-digits” approach to digit recognition. A multilayer network with BP training has also been independently reported in [96], for recognizing handprinted digits scanned from bank checks, and letters interactively entered through a stylus digitizer. This work argues in favor of a large, representative training set. Another departure from [62, 95] is the use of pre-segmented, size-normalized gray-scale character or digit images, instead of local feature maps. The weight sharing principle is used to reduce the number of free parameters. The absence of specific feature maps, appears to have enabled easy generalization to ISR schemes, as discussed later on.

An application of auto-associative MLP to the recognition of handwritten characters taken from the NIST 3 dataset is reported in [97]. In this work, a network is designed for each character, using only instantiations of that character for training. Each NN can thus be viewed as generating a discriminant function. Since it is preferable for an object recognition system to be insensitive to transformations such as translational, and rotation, error measures derived from tangent distances [98] are suggested for learning. One can view this approach as letting the NN learn (or unlearn) skew. Since in most applications, one can deskew and normalize using simple image-processing operations, the advantage of requiring the NN to account for translation and rotation at the cost of increasing the complexity of the NN may be minimal. One can obtain better payoffs by integrating segmentation and recognition, as discussed below.

By feeding candidate segments produced by vertical cuts to a recognizer and scoring the output of the recognizer, the best possible segmentation of an input digit string is obtained in [99]. Knowledge of number of the expected digits is used as an input; since the recognizer used is a convolutional NN, the best segmentation can be obtained by segmentating the feature map.

A method for ISR using a BP algorithm is described in [100]. In a subsequent extension of this work [101], an NN with two hidden layers, the second of which is connected to a block of exponentials before being connected to the output layer, is used for simultaneous segmentation and recognition of handwritten digits from the NIST database. The interconnections between the hidden layers, the second hidden layer and the exponential block are local 3-D interconnections. The output levels are derived from ratios of the sums of the responses of the exponential blocks. The synaptic weight updating rules are derived from minimizing the total sum of squared error measure. A combination of hand-segmented and original (not hand segmented) field data sets is used for training two NNs. The major drawback may be the need to normalize the digits, albeit crudely. The system is able to handle normalizations up to a factor of 2 or more; severe changes in scale seem to reduce the accuracy by 3 to 5%

The NNs for ISR described above have primarily used multilayer feed-forward networks trained using the BP technique. It has been argued in the NN literature that networks that permit feedback

from activation levels at the output could yield improvements in classification as well as generalization. One form of networks that allows such a feedback is the recurrent NN [102]. In [103], the use of a recurrent NN for ISR of digits in the NIST database has been experimented with. Comparisons with other NN-based ISR techniques using the same NIST dataset seem to indicate that the recurrent NN method compares favorably to other NN-based ISR techniques.

Other applications of NNs include modifications to the Neocognitron [104], and a booster technique for improving the accuracy of NN approaches [105].

References

- [1] R. A. Wilkinson, J. Geist, S. Janet, P. J. Grother, C. J. C. Burges, R. Creecy, B. Hammond, J. J. Hull, N. J. Larsen, T. P. Vogl, and C. L. Wilson. The First Optical Character Recognition Systems Conference. Technical Report NISTIR 4912, National Institute of Standards and Technology, August 1992.
- [2] C. L. Wilson and M. D. Garris. Handprinted character database. Technical Report Special Database 1, **HWDB**, National Institute of Standards and Technology, April 1990.
- [3] temp. Information science research institute annual reports. Technical report, University of Nevada, Las Vegas, Las Vegas, Nevada, 1992-1996.
- [4] 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, and C. L. Wilson. The Second Census Optical Character Recognition Systems Conference. Technical Report NISTIR 5452, National Institute of Standards and Technology, May 1994.
- [5] L. K. Hansen, C. Liisberg, and P. Salamon. The error-reject tradeoff. computer reprint, 1995.
- [6] R. A. Wilkinson, M. D. Garris, and J. Geist. Machine-Assisted Human Classification of Segmented Characters for OCR Testing and Training. In D. P. D'Amato, editor, *Conference on Character Recognition Technologies*, volume 1906, pages 267-278, San Jose, 1993. SPIE.
- [7] M. D. Garris. Methods for Evaluating the Performance of Systems Intended to Recognize Characters from Image Data Scanned from Forms. Technical Report NISTIR 5129, National Institute of Standards and Technology, February 1993.
- [8] 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.
- [9] P. J. Grother. Handprinted forms and characters database. Technical Report Special Database 19, **HFCD**, National Institute of Standards and Technology, March 1995.
- [10] M. D. Garris and P. J. Grother. Generalized Form Registration Using Structure-Based Techniques. In *Fifth Annual Symposium on Document Analysis and Information Retrieval*, pages 321-334, Las Vegas, April 1996. UNLV.
- [11] W. Postl. Method for automatic correction of character skew in the acquisition of a text original in the form of digital scan results. United States Patent Number 4,723,297, Feb. 1988.
- [12] M. D. Garris. Intelligent Form Removal With Character Stroke Preservation. In *Conference on Document Recognition III*, volume 2660, pages 321-332, San Jose, February 1996. SPIE.
- [13] M. D. Garris. Unconstrained Handprint Recognition Using a Limited Lexicon. Technical Report NISTIR, National Institute of Standards and Technology, 1993.
- [14] M. D. Garris. Teaching Computers to Read Handprinted Paragraphs. Technical Report NISTIR 5894, National Institute of Standards and Technology, September 1996.

- [15] M. D. Garris. Component-Based Handprint Segmentation Using Adaptive Writing Style Model. Technical Report NISTIR 5843, National Institute of Standards and Technology, June 1996.
- [16] P. J. Grother. Karhunen Loève feature extraction for neural handwritten character recognition. In *Conference on Applications of Artificial Neural Networks III*, volume 1709, pages 155–166, Orlando, April 1992. SPIE.
- [17] 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*, 27(4):485–501, 1994.
- [18] D. F. Specht. Probabilistic neural networks. *Neural Networks*, 3(1):109–118, 1990.
- [19] C. L. Wilson, P. J. Grother, and C. S. Barnes. Binary Decision Clustering for Neural Network Based Optical Character Recognition. Technical Report NISTIR 5542, National Institute of Standards and Technology, December 1994.
- [20] C. L. Wilson, J. L. Blue, and O. M. Omidvar. The Effect of Training Dynamics on Neural Network Performance. Technical Report NISTIR 5696, National Institute of Standards and Technology, August 1995.
- [21] O. M. Omidvar and C. L. Wilson. Information Content in Neural Net Optimization. *Journal of Connection Science*, 6:91–103, 1993.
- [22] J. L. Blue and P. J. Grother. Training Feed Forward Networks Using Conjugate Gradients. In *Conference on Character Recognition and Digitizer Technologies*, volume 1661, pages 179–190, San Jose, February 1992. SPIE.
- [23] M. D. Garris and S. A. Janet. NIST Scoring Package User’s Guide Release 1.0. Technical Report NISTIR 4950, National Institute of Standards and Technology, October 1992.
- [24] M. D. Garris. Public Domain Optical Character Recognition. In *Document Recognition II*, pages 2–14, San Jose, February 1995. SPIE.
- [25] P. J. Grother. Cross Validation Comparison of NIST OCR Databases. In D. P. D’Amato, editor, *Conference on Character Recognition Technologies*, volume 1906, pages 296–307, San Jose, 1993. SPIE.
- [26] D.J. Hepp. Recognition of handprinted and cursive words by finding feature correspondences. *Conference on Document Recognition*, 2181:47–58, 1994.
- [27] M.J. Ganzberger, R. Rovner, A.M. Gillies, D.J. Hepp, and P.D. Gader. Matching database records to handwritten text. In *Proc. SPIE on Document Recognition*, 2181:66–75, 1994.
- [28] G.L. Martin. Centered-object integrated segmentation and recognition of overlapping handprinted characters. *Neural Computation*, 5:419–429, May 1993.
- [29] G.L. Martin and J. Talley. Recognizing handwritten phrases from u.s. census forms by combining neural networks and dynamic programming. *J. of Artificial Neural Networks*, 2:167–194, 1995.
- [30] A. Shustorovich and C.W. Thrasher. Neural network positioning and classification of handwritten characters. *Neural Networks*, 6:685–693, 1996.

- [31] S. Ullman. *The interpretation of visual motion*. MIT Press, Cambridge, MA, 1979.
- [32] G. L. Martin. Centered-object integrated segmentation and recognition for visual character recognition. In J. Moody, S. Hanson, and R. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4, pages 504–511. Morgan Kaufmann, Denver, December 1991.
- [33] . Kanai J, T.A. Nartker, S.V. Rice, and G. Nagy. Performance metrics for document understanding systems. *Proc. Second Intl. Conf. on Document Analysis and Recognition, Tsukuba Science City, Japan*, pages 424–427, October 1993.
- [34] C. Pavlidis and S. Mori. Special issue on optical character recognition. In *Proceedings of the IEEE*, July, 1992.
- [35] C.Y. Suen. A survey of thinning algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7:114–119, 1992.
- [36] R. Plamondon, M. Bourdeau, Chouinard, and C.Y. Suen. Validation of preprocessing algorithms: A methodology and its application to the design of a thinning algorithm for handwritten characters. In *In Proc. Second. Intl. Conf. of Document Analysis and Recognition, Tsukuba City, Japan*, pages 262–269, 1993.
- [37] M.Y. Jaisimha, R.M. Haralick, and D. Dori. A methodology for the characterization of the performance of thinning algorithms. In *Proc. Second Intl. Conf. on Document Analysis and Recognition, Tsukuba City, Japan*, pages 282–286, 1993.
- [38] G. Srikantan, D.S. Lee, and J.T. Favata. Comparison of normalization methods for character recognition. In *Proc. Third Intl. Conf. on Document Analysis and Recognition*, pages 719–722, Montreal, Canada, August 1995.
- [39] H.S. Baird. The skew angle of printed documents. In *Proc. Conf. Soc. Photog. Scient. Eng.*, pages 14–21, Rochester, NJ, May 1987.
- [40] S. N. Srihari and V. Govindaraju. Analysis of textural images using the hough transforms. *Machine Vision and Applications*, 2:141–153, 1989.
- [41] Y. Ishitani. Document skew detection based on local region complexity. In *Proc. Second Intl. Conf. on Document Analysis and Recognition, Tsukuba City, Japan*, pages 49–52, 1993.
- [42] H. Yan. Skew correction of document images using interline cross-correlation. *CVGIP: Graphical Models and Image Processing*, 55:538–543, November 1993.
- [43] H. K. Aghajan, B.H. Khalaj, and T. Kailath. Estimation of skew angle in text image analysis by sensor array processing techniques. In *Conference on on Character Recognition Technologies III*, volume 1906, pages 49–60. SPIE, 1993.
- [44] N. Rondel and G. Burel. Cooperation of multi-layer perceptions for the estimation of skew angle in text document images. In *Proc. Intl. Conf. on Document Analysis and Recognition*, pages 1141–1144, Montreal, Canada, August 1995.
- [45] R. Safari, N. Narasimhamurthi, M. Shridhar, and M. Ahmadi. Document registration using projective geometry. In *Proc. I hird ntl. Conf. on Document Analysis and Recognition*, pages 1161–1164, Montreal, Canada, August 1995.

- [46] C.L. Yu, Y.Y. Tang, and C.Y. Suen. Document skew detection based on the fractal and least squares method. In *Proc. Third Intl. Conf. on Document Analysis and Recognition*, pages 1149–1152, Montreal, Canada, August 1995.
- [47] Y. Lu. Machine printed character segmentation-an overview. *Pattern Recognition*, 28:67–80, 1995.
- [48] R.G. Casey and E. Lecolinet. A survey of methods and strategies in character segmentation. *IEEE Patt. Anal. Mach. Intell.*, PAMI-18:690–706, July 1996.
- [49] F. Kimura and M. Shridhar. Segmentation recognition algorithm for zip code field recognition. *Machine Vision Applications*, 5:199–210, 1992.
- [50] M. Cheriet, Y.S. Huang, and C.Y. Suen. Background region-based algorithm for the segmentation of connected digits. In *Proc. 11th Intl. Conf. on Pattern Recognition*, pages 619–622. The Hague, August 1992.
- [51] N.W. Strathy, C.Y. Suen, and A. Krzyzak. Segmentation of handwritten digits using contour features. In *Proc. Second Intl. Conf. on Document Analysis and Recognition, Tsukuba City, Japan*, pages 577–580, 1993.
- [52] O. Matan et al. Reading handwritten digits: A zip code recognition system. *IEEE Computer*, pages 59–62, July 1992.
- [53] G. Seni and E Cohen. External word segmentation of off line handwritten text lines. *Pattern Recognition*, 27:41–52, January 1994.
- [54] K. Han and I.K. Sethi. Off-line cursive handwriting segmentation. In *Proc. Third Intl. Conf. on Document Analysis and Recognition*, pages 894–896, August 1995.
- [55] A. Negi, K.S. Swaroop, and A. Agarwal. A correspondence based approach to segmentation of cursive words. In *Proc. third Intl. Conf. on Document Analysis and Recognition*, pages 1034–1037, Montreal, Canada, August 1995.
- [56] M.Y. Chen et al. Off-line handwritten word recognition using hidden markov model. In *U.S. Postal Service 5th Adv Technology. Conf.*, pages 563–577, Washington, D.C., November 1992.
- [57] Y. Lu and M. Shridhar. Character segmentation in handwritten words an overview. *Pattern Recognition*, 29:77–96, 1996.
- [58] O. D. Trier, A. K. Jain, and T. Taxt. Feature extraction methods for character recognition a survey. *Pattern Recognition*, 29:641–662, 1996.
- [59] P.J. Grother. Karhunen-loeve feature extraction for neural handwritten character recognition. *SPIE Proc. Applns. of Artificial Neural Networks*, 1709:155–166, 1992.
- [60] M. D. Garris, R. A. Wilkinson, and C. L. Wilson. Analysis of a biologically motivated neural network for character recognition. In *Proceedings: Analysis of Neural Network Applications*, pages 160–175. ACM Press, May 1991.
- [61] A. Shustorovich. A subspace projection approach to feature extraction: the two-dimensional gabor transform for character recognition. *Neural Networks*, 7:1295–1301, 1994.

- [62] J.S. Denker et al. Neural network recognizer for hand-written zip code digits. In *In Neural Information Processing System, D. Touretzky, (ed.)*, volume 1, pages 323–331, 1989.
- [63] G.E. Hinton, C.K.I. Williams, and M.D. Revow. Adaptive elastic models for handprinted character recognition. In *In Advances in Neural Information Processing Systems*, pages 512–519. J.E. Moody, S.J. Hanson and R.P. Lippmann (eds.), 1992.
- [64] C.K.I. Williams, M.D. Revow, and G.E. Hinton. Using a neural net to instantiate a deformable model,. In *Advances in Neural Information Processing Systems, G. Tesanto, et al (eds.)*, 7:965–972, 1995.
- [65] G. H. Granlund. Fourier processing for handprint character recognition. *IEEE Trans. Computers*, 21:195–201, February 1972.
- [66] C.T. Zahn and Roskies. Fourier descriptors for plane closed curves. *IEEE Trans. Computers*, 21:269–281, March 1972.
- [67] E. Person and K.S. Fu. Shape discrimination using fourier description. *IEEE Trans. Syst., man, Cybern.*, 7:170–179, 1977.
- [68] M.K. Hu. Visual pattern recognition by moment invariants. *IRE Trans. Inform. Theory*, IRE-8:179–187, February 1962.
- [69] S.O. Belkasim, M. Shridhar, and A. Armadi. Pattern recognition with moment invariants: A comparative study and new results. *Pattern Recognition*, 24:1117–1138, December 1991.
- [70] J. Flusser and T. Suk. Affine moment invariants: A new tool for character recognition. *Pattern Recognition Letters*, 15:433–436, April 1994.
- [71] R.L. Kashyap and R. Chellappa. A stochastic model for the representation of closed contours. *IEEE Trans. Inform. Theory*, 27:627–637, September 1981.
- [72] S.R. Dubois and G.H. Glanz. An autoregressive model approach to two-dimensional shape classification. *IEEE Trans. PAMI*, 8:55–66, January 1986.
- [73] L. Bottou et al. Comparison of classifier methods: a case study in handwritten digit recognition. *Proc. Intl. Conf. on Pattern Recognition., Israel*, 2:77–82, October 1994.
- [74] S.J. Dickinson, A.P. Pentland, and A. Rosenfeld+. 3-d shape recovery using distributed aspect matching,. *IEEE Trans. PAMI*, 14:174–198, 1992.
- [75] E. Levin and R. Pieraccini. *Planar Hidden markov modeling: from space to optical character recognition*, volume 5. In *Advances in Neural Information Processing System* S-J Hanson, J.D. Cowan and C.L. Giles (eds.), 1993.
- [76] A. Kundu, Y. He, and P. Bahl. Recognition of handwritten word: First and second order hidden markov model based approach. *Pattern Recognition*, 22:283–297, 1989.
- [77] M.Y. Chen, A. Kundu, and J. Zhou. Off-line handwritten word recognition using a hidden markov model type stochastic network. *IEEE Trans. Patt. Anal. Mach. Intell.*, 16:481–496, May 1994.
- [78] H. Bunke, M. Roth, and E.G. Schukat-Talamazzni. Off-line cursive handwriting recognition using hidden markov models. *Pattern Recognition*, 28, 1399-1413 1995.

- [79] W. Cho, S.W. Lee, and J. H. Kim. Modeling and recognition of cursive words with hidden markov models. *Pattern Recognition*, 28:1941–1953, 1995.
- [80] P. Gader et al. Recognition of handwritten digits using template and model matching. *Pattern Recognition*, 24:421–431, 1991.
- [81] A. Marcelli, N. Likhareva, and T. Pavlidis. A structural indexing method for character recognition. In *Proc. Second. Intl. Conf. on Document Analysis and Recognition, Tsukuba City, Japan*, pages 175–178, 1993.
- [82] T.M. Breuel. Recognition of handprinted digits using optimal bounded error matching. In *Proc. Second. Intl. Conf. on Document Analysis and Recognition, Tsukuba City, Japan*, pages 493–496, 1993.
- [83] S. Edelman, S. Ullman, and T. Flash. Reading cursive handwriting by alignment of letter prototypes. *Intl. Jl. Computer Vision*, 5:303–331, 1990.
- [84] S.J. Dickinson, A.P. Pentland, and A. Rosenfeld. From volumes to views: An approach to 3-d object recognition. *CVGIP: Image Understanding*, 55:130–154, 1992.
- [85] C.Y. Suen, J. Guo, and Z.C. Li. Analysis and recognition of alphanumeric handprints by parts. *IEEE Trans. on Systems, Man Cybern*, 24:614–631, April 1994.
- [86] C.Y. Suen. Human factors in character recognition. In *Proc. Intl. Conf. on Syst., Man and Cybern*, pages 253–258, Dallas, Texas, 1994.
- [87] J. T. Favata and S. N. Srihari. Off line recognition of handwritten cursive words. In *Proc. SPIE*, volume 1661, pages 224–234. ?, 1992.
- [88] G. Seni, V. Kripasundar, and R.K. Srihari. Generalizing edit distance of handwritten text recognition. *Proc. SPIE*, 2422:54–65, 1995.
- [89] et al. S.N. Srihari. Name and address block reader system for tax form processing. *Proc. Third Intl. Conf. on Document Analysis and Recognition, Montreal, Canada*, pages 5–10, 1995.
- [90] F. Kimura, M. Shridhar, and N. Narasimhamurthi. Lexicon directed segmentation-recognition of unconstrained handwritten words. In *Proc. Third Intl. workshop on frontiers in handwriting recognition*, Buffalo, 1993.
- [91] I. Guyon, V. N. Vapnick, B. E. Boser, L. Y. Botton, and S. A. Solla. Structural risk minimization for character recognition. In R. Lippmann, editor, *Advances in Neural Information Processing System*, volume 4, pages 471–479. Morgan Kaufman, 1992.
- [92] D. E. Rumelhart and G.E. Hinton and R.J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing : Explorations in the Microstructure of Cognition, D.E. Rumelharf and J.L. McClelland (eds.)*, volume 1, pages 318–362. Brandford Books, MD, 1986.
- [93] Y. LeCun et al. Optimal brain damage. In *Advances in Neural Information Processing Systems, D.S. Touretzky (ed.)*, Vol. 2, volume 2, pages 598–605, 1990.
- [94] I. Guyon et al. Structured risk minimization for character recognition. ?, ?

- [95] Y. LeCun et al. Back propagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.
- [96] G.L. Martin and J.A. Pittman. Recognizing hand-printed letters and digits. *Neural Networks*, 3:258–267, 1991.
- [97] H. Schwenk and M. Milgram. Transformation invariant auto association with application to handwritten character recognition. In *advances in neural information processing systems*, 7:991–1006, 1995.
- [98] P. Simard, Y. Le Cun, and J. Denker. Efficient pattern recognition using a new transformation distance. In *Advances in Neural Information Processing Systems*, volume 5, pages 50–58. S. Hanson et al (eds.), 1993.
- [99] O. Matan and C.J.C. Burges and Y. LeCun and J.S. Denker. Multi-digit recognition using a space displacement neural network. In *Advanced in Neural Information Processing Systems*, J.E. Moody (eds.), volume. 4:pages 488–495, 1992.
- [100] J.D. Keeler, D.E. Rumelhart, and W.K. Leow. Integrated segmentation and recognition of hand-printed numerals. In *In Neural Information Processing Systems*, R. Lippmann, J. Moody and D. Touretzky (eds.), volume 3, pages 557–563. ?, 1991.
- [101] J. Keeler and D. Rumelhart. A self organizing integrated segmentation and recognition neural net. In *advances in neural information processing systems*, J.E. Moody, S.J. Hanson and R.L. Lippmann (eds.), 4:496–503, 1992.
- [102] S. Haykin. *Neural Networks*. Prentice-Hall, Englewood Cliffs, NJ, 1993.
- [103] S.W. Lee and E.J. Lee. Integrated segmentation and recognition of connected handwritten characters with recurrent neural network. In *Proc. Third Intl. Conf. on Document Analysis and Recognition*, pages 413–416, Montreal, Canada, August 1995.
- [104] K. Fukushima. Connected character recognition with a neural networks. In *Proc. Intl. Conf. on Document Analysis and Recognition*, pages 240–242, Montreal, Canada, August 1995.
- [105] H. Drucker, R. Schapire, and P. Simard. Boosting performance in neural networks. *Intl. Jl. Patt. Recn. and Artificial Intell*, 7:705–719, 1993.

