Proceedings

1995 Symposium on Document Image Understanding Technology

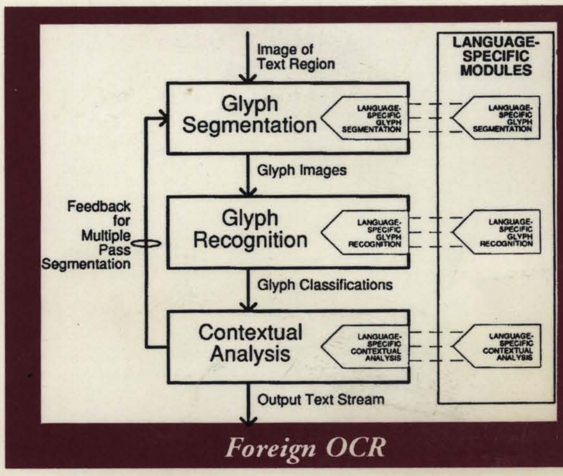
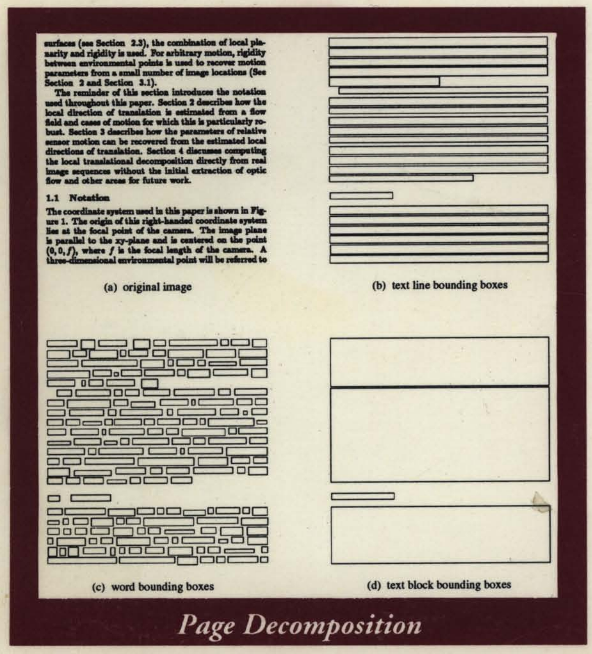
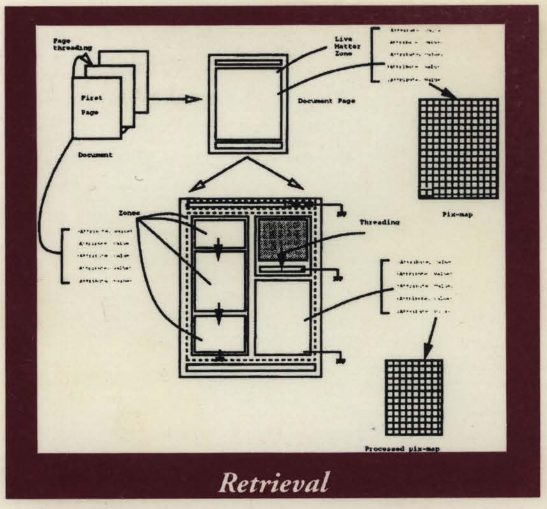


Image of Text Region
LANGUAGE-SPECIFIC MODULES
Glyph Segmentation
LANGUAGE-SPECIFIC GLYPH SEGMENTATION
Glyph Images
Feedback for Multiple Pass Segmentation
Glyph Recognition
LANGUAGE-SPECIFIC GLYPH RECOGNITION
Glyph Classifications
Contextual Analysis
LANGUAGE-SPECIFIC CONTEXTUAL ANALYSIS
Output Text Stream

Foreign OCR



Page Decomposition



Retrieval

October 24–25, 1995
Comfort Inn and Conference Center
Bowie, Maryland

Proceedings

SDIUT9~~5~~7

The 199~~5~~7 Symposium on Document Image Understanding Technology

~~Bowie, Maryland~~ Annapolis, Maryland
~~October 24-25, 1995~~ April 30-May 2, 1997

Sponsored by: The United States Department of Defense
Organized by: ~~Document Processing Group, Center for Automation Research,~~
University of Maryland, College Park, MD 20742-3275
The Laboratory for Language and Media Processing
Institute for Advanced Computer Studies

A limited number of additional copies of this proceedings is available for $50 from:

SDIUT9~~5~~7
~~Center for Automation Research~~ [illegible]
University of Maryland
College Park, MD 20742-3275
Phone: (301) 405-4526
Fax: (301) 314-9115
email: sdiut9~~5~~7@documents.cfar.umd.edu [illegible]

Table of Contents

KEYNOTE SPEAKERS

RETRIEVAL

PAGE DECOMPOSITION

SYSTEMS

Forms and Logos

Evaluation

Foreign OCR

Miscellaneous Document Understanding

Additional submissions

A Message from the Sponsors

We welcome you to this first Department of Defense/Ft. Meade, Maryland sponsored Symposium on Document Image Understanding Technology.

The purpose of this symposium is to provide a forum where technical personnel from industry, academia, and government can discuss emerging capabilities and their respective needs. We hope this will contribute future solutions which automate some aspect of the input, analysis, retrieval, or dissemination of data related to document images.

You were invited to participate because you are experienced in technology associated with document processing or have project responsibilities that can benefit from these technologies.

If we can be of any assistance to you during this Symposium, please feel free to contact us or the University of Maryland staff with the Center for Automation Research. Your feedback on this event will be helpful in future planning and will be used to determine how best to serve the interests of the research community, technology developers, and organizations engaged in applying these advanced capabilities for systems working in a variety of environments.

We encourage you to see out your colleagues and look for productive way to collaborate on meeting the technical challenges. We thank you for participating in this Symposium and extend our best wishes for a successful Symposium.

Linda Kay Document Analysis	Steve Dennis Imaging Research

A Message from the Organizers

On behalf of the Document Processing Group at the University of Maryland, I would like to welcome you to the 1995 Symposium on Document Image Understanding Technology.

The Symposium program contains over 30 presentations from academic, government and industrial researchers located all across the country, dealing with a wide variety of document image understanding topics. We have tried to organize each session so that a general overview talk is followed by progress reports by of researchers in the field. A demonstration session has also been scheduled on the first afternoon to give participants the opportunity to interact with working systems.

I would like to thank everyone who has contributed to this Symposium. A special thank you is extended to the staff of the Center for Automation Research: Jeannie Sullivan, Sailaja Akunuri, Janice Peronne, and Margaret Masters, for their many hours of work on the local arrangements; and Sandy German, for her technical expertise in preparing the program and proceedings.

David Doermann
University of Maryland

Adobe Acrobat Capture

David Emmett
Adobe Systems, Inc.
P.O. Box 7900
1585 Charleston Road
Mountain View, CA 94039-7900

Abstract

Adobe introduced Acrobat Capture at the AIIM show in April of this year. Capture integrates page decomposition, image processing, optical character recognition and font recognition to convert a scanned page into a look-alike Acrobat PDF document. The converted document can then be viewed distributed and managed by Acrobat on any platform. In this presentation Dave Emmett, Adobe's Director of Recognition Technologies will demonstrate Capture and he will also describe a modular version of Capture that allows integrators to incorporate Capture technologies into their own applications. The presentation will conclude with a description of Capture's FlexFrame architecture.

Begin [illegible] page numbering here

Retrieval

Document Image Retrieval

Stephen J. Dennis
Department of Defense
9800 Savage Road
Fort Meade, MD 20755-6000

1 Interest

The Department of Defense at Fort Meade, MD, has a long term interest in the research and development of technologies to analyze and represent documents images for a wide variety of image qualities, document formats, and written languages. The goal of our research is to develop the component technologies that enable robust system architectures to be portable across written languages, subject domains, and computer systems.

While there are many different Intelligence Community (IC) requirements for the automatic processing of document images, one of the most common application areas is content based selection and prioritization. This application area is very similar in concept to the information retrieval or routing of electronic text, but is significantly more complex given the flexibility of a less constrained input media, e.g. paper documents. Additional layers of complexity arise from uncontrollable damage to image data that can often cause significant disparity between an original document and the corresponding image data.

For years the IC has funded significant research and development efforts for image processing technology to address automatic processing requirements for all kinds of images apart from those of paper documents. These efforts have contributed many useful technologies and have served to unify the field of image processing into a separate and respected engineering discipline. There is an immediate need to begin a transfer of previously developed image processing technology to the context of processing document images.

The DoD has initiated internal research programs to execute short term technology development efforts that will address some of the desired capabilities. However much of the technology that is required to develop fully automatic content based selection systems for document images does not exist.

2 Research and Development

Previous commercial efforts in the research of document image understanding systems have been limited to very lucrative but narrow commercial niches that do not serve broader IC requirements. These systems rely on intense human interaction in order to manually process and index documents for future retrieval tasks. These systems are very limited in their capability to process variants of a single writing system, document styles, or image qualities and are comprised of very narrow and deep engineering solutions that are not easily ported to new problem sets.

We are involved in and are tracking other current government sponsored research efforts to address single component technologies that will aid in document image understanding. While these efforts have a positive impact on our eventual goal, there is no consolidated model of government operations that can be used to focus the research community in order to optimize research efforts.

The DoD research goals include the investigation of open architectures for document image processing that allow for a variety of feedback paths as opposed to the traditional pipeline feed forward architectures of today. This architecture will be comprised of a variety of modules for document image and text analysis that can be easily combined to identify the priority relevance of document images to an analyst query statement. There is little known about the performance of systems that combine the best of image analysis, language processing, and statistical features to discover useful representations from document images.

In a scenario of routing document images, the analysis and recognition techniques are required to automatically disseminate a document image corpus to end users based upon a query statement. Because the number of images is large, and a combination of image and text analysis could require significant computational complexity, the speed of any dissemination technology will be a factor in the development of resulting algorithms or systems.

Once a document has been successfully analyzed, there are requirements to create large archives of document images that can be used for topical research. This requirement forces the issue of document representation. Once the components of a document image have been discovered, there is little known about efficient structures for the maintenance of component relationships within a document image. Much less is known about the maintenance of the kinds of relationships that an analyst will form based upon reporting requirements and human inferences.

The final and most important area of research involves the investigation of technologies that enable an analyst to browse, research, and retrieve relevant document images on demand through intuitive computer interface operations. The success of any research and development program rests entirely upon the degree to which we are able to increase the efficiency of a process. Since the goals of these efforts are to deliver content based selection technologies for use by human analysts, the tools which enable successful interface technologies are as important as the analysis and representation technologies themselves.

3 Proposal

We continue to support the consolidation of IC and other government research requirements for document image processing and related technologies through common funding and project offices like those found in ARPA. It is clear that basic research in this area will be required to support the discovery and entrepreneurial development of technologies, algorithms and systems.

In addition to supporting these activities, the DoD has formed a committee to establish a government wide Document Understanding Conference (DUC). This conference will focus the image and language processing research communities on common tasks in document image processing that area beneficial to the IC, civilian government agencies, and commercial industry. This conference will serve to establish a common technology baseline, to identify component technologies, and to force the integration of existing image and language processing technologies to ensure rapid identification and retrieval of relevant document images in response to government analyst requirements.

The evaluation techniques that are currently being developed as part of the joint ARPA and CIA effort will serve as the basis for DUC evaluations and will provide requirements to the standards committee. Data collection efforts have been established that will provide initial image corpora for DUC tasks beginning as soon as June of 1996.

The format for this effort, and the inclusion of government, academic, and industry experts in the planning process is key to the success of this effort. The DUC will be modeled after the Message Understanding Conference (MUC) and the Text Retrieval Evaluation Conference (TREC). These efforts were key to successful revitalization of the language processing and information retrieval communities.

Media Independent Data Representation Context Vector Techniques for the Unification of Text and Image Information Spaces

Bill Caid
HNC Software Inc.
5930 Cornerstone Court West
San Diego, CA 92121-3728
619-546-8877 x322
caid@hnc.com

Abstract

Management of large sets of mixed media documents has become a major issue such that the ability to identify and locate information of interest has become a problem of large proportions. Retrieval of information that is in a text-only format is a historically difficult problem that has been studied for years. However, substantial progress has been made in improving the quality of information retrieval technology and many systems exist today. Images, however, are quite different and require a more creative approach for successful information retrieval. To date, only a very few image retrieval/management systems have been developed. None of these systems provide a unified information space for representing both text and images.

A vector space technique has been developed that can provide a unified representation for text and images. This technique can provide a set of unique capabilities that can be applied to the mixed media management problem. This context vector technique for text was developed as part of the ARPA-sponsored TIPSTER Text Program and has been demonstrated on multi-gigabyte corpora.

The context vector approach has been extended to data in the image domain as part of the Image Content Addressable Retrieval System (ICARS) SBIR. This research, though in a preliminary state, has demonstrated the viability of the context vector representation of images based upon a machine-derived set of image "symbols". The context vector approach is very powerful and generic because it can be applied to any form of data that can be represented by a finite set of symbols. A key aspect of the context vector technique is that it is fully automated. That is, no external knowledge is utilized to form the final context vector representation.

This paper provides an overview of the context vector representation and how this representation can be used to provide a media-independent representation of information, specifically text and images. Current efforts within HNC seek to develop a media independent context vector representation for both text and images that will provide the capability to use free text as a query to a mixed-media database resulting in text and images as the query result. Additionally, this research will provide the basis for the ability to use images as queries to the system resulting in both text and images as the query result.

1 Background

1.1 Problem Description

The proliferation of cost effective personal computers has caused an explosion in the growth of electronic documents. Additionally, the availability of relatively inexpensive document scanners has further increased the number of documents that are available in electronic format. Management of these large sets of documents has become a major issue such that the ability to identify and locate information of interest has become a problem of large proportions.

The pervasive use of PC-based word processors as well as document scanners has resulted in "mixed media documents" that contain both text and images (imbedded figures). Retrieval of information that is in a textual format is a historically difficult problem that has been studied for years, but substantial progress has been made in improving the quality of information retrieval for text-based information. Images, however, are different and require a more creative approach for successful information retrieval.

To date, only a few image retrieval systems have been developed.

A system that provides a unified information representation for both text and images could address a number of key issues associated with the management of mixed media documents. Specifically, a unified representation for both text and image data could provide a mechanism for effective multi-media information retrieval. Additionally, this representation could allow automatic retrieval of images mentioned or described in surrounding text when using the surrounding text as a query. Conversely, this technique could allow the use of images as queries to the system resulting in both images and surrounding text as query results.

HNC has developed a vector space technique that can provide a unified representation for text, images and numeric data. Key portions of the required technical capabilities for text and images have already been demonstrated. The text representation approach was developed as a result of the ARPA-sponsored TIPSTER effort. Image representation research is currently being sponsored by Rome Laboratory and early results have shown great promise.

This white paper describes an approach for integrating the information representation for text and images and proposes a set of research actions to demonstrate the viability of this concept. Additionally, this white paper provides an overview of the context vector technique for text and describes how this context vector approach has been extended to images.

1.2 Context Vector Approach to Information Representation

HNC has developed a mathematical approach to information representation based upon symbolic association of information symbols with high dimensional vectors. These vectors, called "context vectors", provide a media-independent representation that can learn N-way relationships between information symbols. These learned relationships can be used to provide a basis for information retrieval, routing and clustering of sets of symbols that have similar learned relationships.

The basic context vector technique is predicated upon three key concepts. First is the concept of a *"quantized information vocabulary"*. The second is *"symbolic association"* and the third concept is *"proximate co-occurence"*. The sections below provide details about these concepts and how these concepts can provide a basis for a unified representation for text and image data.

1.2.1 Quantized Information Vocabularies

The principle of quantized information vocabularies is straightforward and is exploited every day by the music industry. This principle states that any continuous value information channel can be represented by a set of discrete symbols. This approximation can be achieved to an arbitrary degree of precision by increasing the number of symbols in the information vocabulary. The principles of modern digital signal processing exploit this principle in the compact disc where a continuous, time domain signal is encoded as a stream of discrete symbols that represent the amplitude of the signal. For the compact disc, these symbols are derived by quantizing the amplitude of the music signal using 16 bit linear pulse coded modulation. The resulting stream of 2^{16} possible symbols represent the associated music. The key concept is that a continuous-valued signal can be represented by a finite (but possibly large) set of discrete symbols.

1.2.2 Symbolic Association

The concept of symbolic association is also straightforward and states that a set of symbols can be associated with an alternate form of information. A common example of this concept is the association of a home with a phone number or a person with a social security number. The previous examples described associations between two discrete sets of symbols. Though this association is useful, it is very restricting. The concept can be extended to include associations between a discrete entity and a vector, in particular, a high dimensional, real vector. The context vector approach, as currently implemented, uses vectors of real numbers providing a tremendous increase in the representational power over a simple discrete symbol set (like a social security number, for instance). The key concept is that symbols in a finite set can be mapped to a more powerful representation using symbolic association. In the case of context vectors applied to text, words are associated with high dimensional floating point vectors[1]. For images, localized and quantized features are mapped to context vectors.

1.2.3 Proximate Co-Occurrence

In many information mediums, co-occurence or near (proximate) co-occurence of symbols convey information. Clearly, in the text domain, co-occurent and proximate co-occurence are the mechanisms used to convey information using a finite set of symbols (words). The characteristics of proximate co-occurence can be learned by example from a training set using symbolic association mapping of symbols to a richer (vector) information space. This mapping is followed by application of a self organizing learning law applied to the vector space. The learning, however, is driven by the proximate co-occurence of the original information symbols. In this way, the proximate co-occurence statistics of the symbols

[1] It should be noted that the act of symbolic association makes the context vector representation *language independent*.

determines the relationships of the associated vector field. When learning is complete, the directions of the symbol vectors encode the proximate co-occurence relationships. The power of this approach is derived from the ability to position vectors in the space based upon their symbolic associations. This technique is very generic and can be applied to any form of information that can be represented by a finite vocabulary of symbols[2]. The exact nature of the learning law used to position the symbol vectors can be tailored to system requirements and the characteristics of the symbolic data. In the text case, the learning law is designed such that words that are used in a similar context will have their associated vectors point in similar directions. For text, "context" is really proximate co-occurence. In the image case, image "primitives" that co-occur in fixed spatial relationships will have their associated vectors that point in similar directions. For a mixed-media case, it is clear that some form of enhancement to the learning law will be required.

1.3 Context Vectors Applied to Text: MatchPlus

Early work by Salton [1,2,3], and later work by Salton and Buckley [4] described a model for text where each document was represented by a vector. The vector space consisted of a number of coordinates that was equal to the number of unique terms in the system (the "vocabulary"). The direction of the document vector was determined by counting the number of terms in the document, then applying an appropriate term weighting and normalization [4,5] to these counts. This approach produces what we will refer to as a "term orthogonal" space. That is, each term in the vocabulary is a coordinate and all coordinates are, by definition, orthogonal.

This approach results in several unavoidable consequences. First, is the high dimensionality of the resulting vector space. Second, there is no provision for encoding similarity of meaning at the term level. Given the availability and capacity of low cost, high performance computing resources, the first consequence (high dimensional space) has little practical impact and can be ignored. The second consequence, term orthogonality and inability to represent similarity of meaning, however, remains.

Extensions to the term orthogonal model have been proposed and demonstrated in the form of Latent Semantic Indexing (LSI) [6]. The basic LSI model extracts information in vector form by identifying the dependencies between terms and documents. This approach is based on constructing and processing a term-by-document matrix. In LSI, the resulting dimension of the vector model is determined via singular-value decomposition of the term-by-document matrix and, in general, will be significantly smaller than either the term or document count (typically 100 - 300). As a consequence, the vectors that represent words *cannot* be orthogonal. As such, the LSI approach encodes a form of similarity of usage, but at a low resolution, term-by-document level. Efforts by Deerwester, Furnas, Dumais, et al., [6,7,8] demonstrated the viability of the non-term orthogonal approach and provided the motivation for the early *MatchPlus* concept as proposed by Gallant [9]. HNC's approach used in *MatchPlus* is, in a sense, a close relative of LSI in that:

1. The vector space is not term orthogonal.
2. Relationships are computed (learned) from a training corpus.
3. The resulting dimensionality of the vector space is much smaller than either the term or document count in the corpus.

However, the *MatchPlus* approach has several key differences:

1. Relationships are determined at a finer resolution than the document level. That is, proximity of occurrences is taken into account in the *MatchPlus* approach. This provides *sensitivity to similarity of usage at the word level.* Word sense identification and disambiguation is a direct consequence of this sensitivity [10].
2. The MatchPlus learning algorithm is based on self organization and employs an adaptive neural network learning law. This is in contrast to LSI's matrix manipulation approach which is a "block update" approach.
3. No orthogonal basis sets are derived in the MatchPlus technique. The learning technique chooses a coordinate space that is convenient for the algorithm and is unintelligible to the human.

The key innovations behind the MatchPlus approach are motivated by the desire to exploit neural network learning techniques to discover similarity of usage at the word level, in a language-independent manner, without the need for external dictionaries, thesauri or semantic networks. Additionally, since many domains use terms in a manner that are specific to that domain, we seek to have the MatchPlus algorithm determine these usages from the training corpus directly. Additionally, we seek to use the resulting word vectors to provide retrieval, routing, document clustering and summarization.

The HNC MatchPlus system was developed as part of the ARPA-sponsored TIPSTER text program. During this effort, the initially proposed context vector approach using human defined coordinates and initial conditions was extended and refined to allow fully automatic generation of context vectors for text

[2] It should be noted that the approach can be extended to any symbolic information of interest. For example, context vectors could be associated with bank accounts, dollar amounts, phone numbers, people, places, etc. Because of this symbolic association property, this technique has the potential to be the ideal mechanism for integrating multi-media data.

symbols (stems) based upon their demonstrated context of usage in training text. The MatchPlus system learns relationships at the stem level and then uses those relationships to construct a context vector representation for sets of symbols. For the text case, these sets of symbols are paragraphs, documents and queries. To start the learning process, each stem is associated with a random vector in the context vector space. Random unit vectors in high dimensional floating point spaces have a property that is referred to as "quasi-orthogonality". Stated mathematically, this quasi-orthogonal condition can be expressed as:

$$E\left[CV_i \bullet CV_j\right] \le \varepsilon \quad \forall_{i,j}$$

$$\varepsilon \approx 0$$

Equation 1. Quasi-Orthogonality Condition.

In words, the above relationship shows us that "the expected value of the dot product between any pair of random context vectors selected from the set is less than epsilon, where epsilon is "small" (approximately equal to zero). The value epsilon can be made arbitrarily small by increasing the dimensionality of the context vector space when the quasi-orthogonal set is created. Dimensionality of the space, then, translates into capacity of the space where capacity is defined as the total number of vectors that can occupy the space and meet the quasi-orthogonality condition of dot products less than epsilon [16].

This property of quasi-orthogonality is important because it serves as the initial condition for the context vector learning algorithm. The usage of the context vector technique is predicated upon the rule that symbols (stems) that are used in a similar context (exhibit proximate co-occurrence behavior) will have trained vectors that point in similar directions. Conversely, stems that never appear in a similar context will have context vectors that are approximately orthogonal. A graphical example of this condition is shown below in Figure 1.

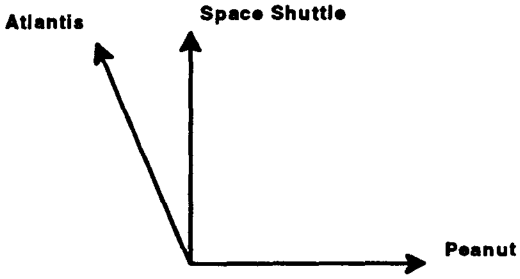


Figure 1. Similarity of Context and Direction.

To achieve the desired representation, the context vector learning algorithm must take the context vectors for symbols that co-occur and move them toward each other. Symbols that do not co-occur are left in their quasi-orthogonal original condition. It is a basic tenet of the MatchPlus approach that "words that are used in a similar context convey similar meaning". Since the learning is driven by proximate co-occurrence of words, the learning results in a vector set *where closeness in the space is equivalent to closeness in subject content.*

To perform learning, a convolutional window is used to identify local context. The window has one target stem and multiple neighbor stems. An example of this convolutional window is shown in Figure 2.

Source Text

The Maoist rebel group, "Sendero Luminoso", bombed the DEA office in Lima

Stemmed Text

Mao rebel group Sendero^Luminoso bomb DEA offic Lim

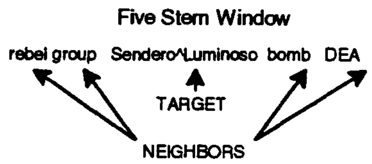


Figure 2. Convolutional Window Example.

Once the context window has been determined, the learning rule of "Move context vector for target in the direction of the context vector of the neighbors" is applied. Once correction is made, move convolutional window to next location and the learning operation is repeated. The equation for this learning is shown in Equation 2.

$$T_j^{New} = T_j^{Old} + \gamma \cdot \sum_{i=window} \left(\alpha_{ij} - T_j^{Old} \bullet N_{ij}\right) \cdot N_{ij}$$

$$T_j^{New} = \frac{T_j^{New}}{\left\| T_j^{New} \right\|}$$

where:

T_j^{New} = Target context vector after update

T_j^{Old} = Target context vector befor update

γ = Adjustment step size

N_{ij} = Context vector for neighbor i, window j

α_{ij} = Proximity constraint for neighbor i, target j

Equation 2. MatchPlus Learning Equations.

Several points should be noted:

- It can be seen in Equation that the learning technique is a constrained self-organization where "alpha" is the constraint. Alpha determines the maximum allowable dot product proximity for stem context vectors in a window.
- All stem vectors are of length 1 (unit vectors). In this paradigm, only the direction of the vector carries information.

- Fully trained vectors have the property that words that are used in a similar context will have vectors that point in similar directions as measured by the dot product.
- Words that are never used in a similar context will retain their initial condition of quasi-orthogonality. That is, approximately orthogonal with a dot product of approximately zero.
- Trained context vectors result in a concept space where similarity of direction corresponds to similarity of meaning.
- No human knowledge is required for training to occur. Only free text examples are needed.
- The algorithm determines the coordinate space of the context vectors.

When the training is complete, "words that are used in a similar context will have their associated vectors point in similar directions". Conversely, words that are never used in a similar context will have vectors that are approximately orthogonal.

Once the stem learning is complete, it is possible to "query" the vector set to determine the nature of the learned relationships. To perform this operation, the user selects a "root" word and the trained context vector for that word is determined by a table lookup in the context vector vocabulary. MatchPlus computes the dot product of every other word vector in the vocabulary to the selected word. The resulting dot products are sorted by magnitude where larger means closer in usage. Figure 3 below shows a graphical example of the relationships learned from 250 Megabytes of the 1993 IRS Federal Tax Code, US Tax Court Rulings and US Supreme Court Tax Rulings.

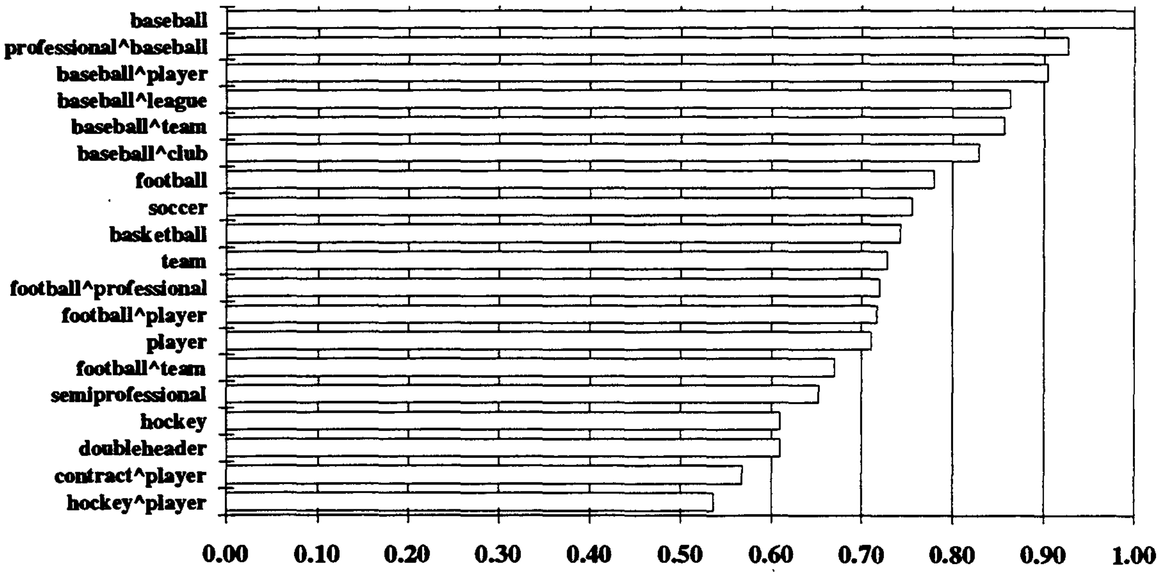


Figure 1. Learned Relationships for Baseball

Notice that in Figure 3, the length of the horizontal bar corresponds to the strength of the learned relationship (dot product). Also note that the words are ordered from strongest at the top to weakest at the bottom. Baseball is related to itself with a strength of 1.00. Note that *professional baseball* is very close in this concept space to *baseball*. This is because in terms of context of usage in the tax codes, they are very similar. Additionally, it can be seen from inspecting Figure 3 that *football, soccer, basketball and hockey* are also close to *baseball*. These relationships are learned solely from the training text based on demonstrated proximate co-occurrence of stems.

Sets of words (text passages and queries) and documents can also be represented by context vectors in the same information space. Document context vectors are derived as the inverse document frequency-weighted sum of the context vectors associated with words in the document. Document context vectors are normalized to prevent long documents from being favored over short documents. This process is shown in Equation 3.

$$DCV_i = \frac{\sum_{j=1}^{j=N\_Stems_i} \log(\frac{N}{N_j}) \cdot CV_j}{\left\| \sum_{j=1}^{j=N\_Stems_i} \log(\frac{N}{N_j}) \cdot CV_j \right\|}$$

where:

N = Total number of documents in the corpus.

N_j = Number of documents that contain stem j.

CV_j = Context vector for stem j.

N_Stems_i = stems in document i.

Equation 3. Document Context Vector Formation.

The resulting document context vectors have the property that *documents that discuss similar themes will have context vectors that point in similar directions*. It is this property that translates the problem of assessment of similarity of content for text into a geometry problem. Documents that are similar are close in the space and dissimilar documents are far away. Additionally, it should be noted that all document vectors are unit length. This prevents system biases in retrieval due to document length. Document retrieval is performed by converting the user free text query to a context vector using Equation 3. Then, using the database of document context vectors, the relevance of each document is determined by dot product. The resulting list is sorted by relevance to produce the final ranked list of relevant documents that are presented to the user.

1.4 Context Vectors Applied to Images: ICARS

1.4.1 ICARS System Overview

The ICARS image characterization concept represents a revolutionary step forward in image management. However, it is an evolutionary step beyond the text characterization and retrieval approach employed in the MatchPlus system. Stated differently, ICARS is the MatchPlus approach extended to handle images. A key aspect of this technique is the recognition of the parallels between the context vector approach for text and the context vector approach for image retrieval. Specifically, from the standpoint of context vector generation and learning, there is a close analogy between text and images. Consider the following analogies: **MatchPlus Text Case**	**ICARS Image Case**
A document is a collection of words in a specific sequence. A document might be a message, a news article, etc.	An image is a set of "atomic" elements. These atoms might be primitive shapes, textures, or attributes of the image that can be identified by wavelet transformations. These atoms are in a specific spatial orientation to one another in the image.
The specific words, their sequence and the proximity of these words to one another define the context and content information of the document.	The specific atoms, their number and spatial orientation of atoms relative to one another defines the context and content information of the image.
For a collection of documents, the set of unique words in the collection defines the "vocabulary" of the language used in the documents. This vocabulary defines the "frame of reference" used for conveying information.	For a collection of images, the set of unique atoms in the collection defines the "vocabulary" of the atomic language used in the images. Stated differently, the set of atoms defines the basis of communicating meaning in the image collection.

Figure 4. Text-Image Analogies

Thus, using a slight redefinition of normal usage of terms, text and images are a direct analogy if "atom" is substituted for "word" and "image" is substituted for "document". There is the additional complexity of two dimensional spatial relationships rather than simple sequence. However, this complexity can be easily accommodated within the framework of the context vector concept.

It is important to note that for both the text and image cases, there is no requirement to *understand* the data objects being characterized. For text, MatchPlus does not attempt to understand the words or their meanings. Only the relationships between the words are learned - which is a much simpler problem. Similarly, in the case of images, the ICARS system does not attempt to recognize objects within images. ICARS only needs to determine features and characterize the relationships between usage of these features to be effective.

Recent research has established that there exists an efficient approach to localized characterization of information contained within images, [11 - 15]. These studies show that the Gabor "wavelet" transformation provides a robust representation scheme that is sensitive to orientation and localized spatial frequency content. As such, wavelets provide a mechanism for characterizing the information content of images in a compact and efficient fashion. There is strong evidence that the wavelet approach is employed in biological organisms for vision [15]. These wavelets fit the receptive profiles found in the mammalian vision system, and their parameters capture the neurophysiological properties of localization, spatial frequency and orientation sensitivity [17]. Consequently, wavelet representations provide local, multi-resolution features that are robust and relatively noise immune.

A key concept of the ICARS system is the representation of images by a set of atoms. These atoms are most easily thought of as primitive building blocks of images: an "atomic vocabulary" for images. These building blocks, when combined in the correct sequence and in the correct spatial positions, define the image. Under this definition, images can consist of any number of individual atoms in any combination. The information content of the image used in the ICARS system is defined solely in terms of the specific atoms that comprise the image and the spatial orientation of the atoms to each other. For ICARS, an atomic vocabulary consisting of ensembles of Gabor wavelet transformations are used. The resulting feature vector vocabulary represents a comprehensive set of image primitives that serves as a basis for atomic characterization of arbitrary images.

ICARS uses a multiple-wavelet representation of the information contained within the image as the basis for defining the vocabulary for context sensitive retrieval. To implement this approach, wavelet parameters are selected to capture orientation sensitive spatial frequencies at a variety of orientation/frequency combinations over the range of interest. The final combination, for example, might be eight orientations, five spatial frequencies. Using this approach a total of 8*5 (40) real valued numbers would be computed for each point in the image that is sampled. Stated differently, each sample point is characterized by a 40 dimensional vector. An example of this set of wavelet kernels is shown in Figure 5.

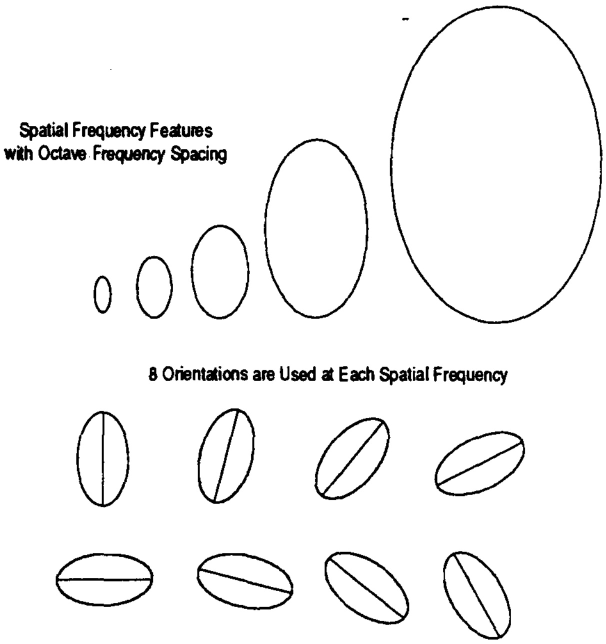

Spatial Frequency Features
with Octave Frequency Spacing
8 Orientations are Used at Each Spatial Frequency

Figure 5. Wavelet Ensemble Example.

Using this approach, an image would be characterized using wavelet ensembles representation evaluated at each point on a grid superimposed on the image ("sample points"). The current technique uses a simple, offset grid sampling. Clearly, the accuracy of the representation is linked to both the number of points sampled, where they are sampled and the discrimination of the wavelet transformations employed.

Since the actual values that result from the wavelet transformations are real valued vectors, the feature vector values would, in general, be unique possibly resulting in a near-infinite number of combinations. The next step is to statistically select the "atomic vocabulary" from the set of 40 dimension feature vectors using a vector quantizer. This vocabulary, then, yields the finite symbols set required to allow context vector learning. A block diagram of this operation is shown inFigure 6.

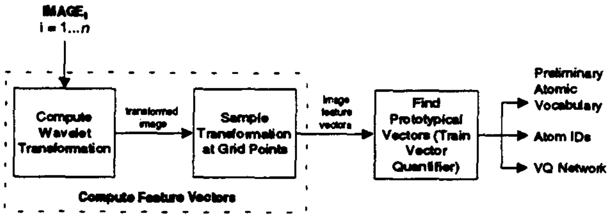

IMAGE$_i$
i = 1...n
Compute Wavelet Transformation
transformed image
Sample Transformation at Grid Points
image feature vectors
Find Prototypical Vectors (Train Vector Quantifier)
Preliminary Atomic Vocabulary
Atom IDs
VQ Network
Compute Feature Vectors

Figure 6. Atomic Vocabulary Definition.

A key feature of the ICARS system is the ability to learn the context of image atoms from examples. This learning procedure is referred to as "bootstrapping" and is based on the concept of atomic proximate co-occurrence. That is, atoms that appear spatially close to one another in a significant number of images are related. The closer the atoms appear to one another in the image, the stronger the relationship between the atoms. Additionally, the importance of the atoms is related to both the local frequency (within this image) and the global frequency (within the image corpus as a whole[3]). The proximate co-occurence and frequency relationships are handled as part of the two-dimensional learning algorithm.

Once atomic learning is complete, a context vector representation for an image can be computed in the same fashion as text. Specifically, the image context vector is the frequency weighted sum of its constituent atom context vectors. This operation is shown in Figure 7.

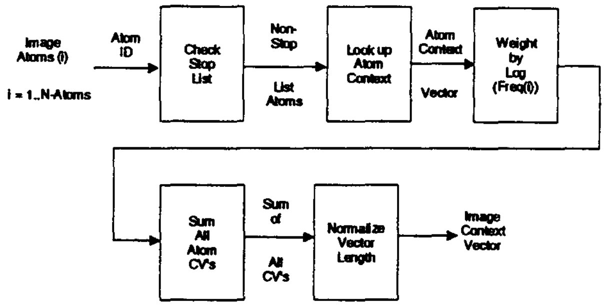
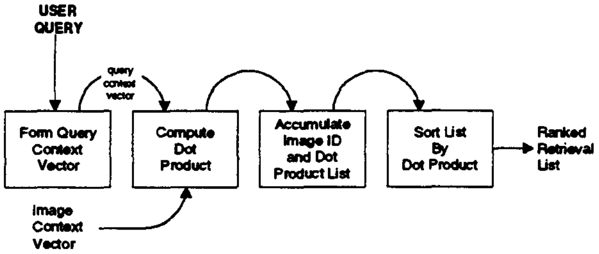


Figure 7. Image Context Vector Generation.

Once the capability to generate image context vectors is available, image retrieval "by example" can be achieved. Like the context vector approach used in MatchPlus, all image components are represented by a context vector: atoms, groups of atoms, images, and queries. This consistent framework allows queries to be created using several approaches. Regardless of the approach used to generate the query, all queries will be represented by a context vector. Whole images can be used as queries. This is, in essence, asking the system to retrieve images that "look like" the image used as the query. Image retrieval is performed in an identical fashion to document retrieval in MatchPlus. Image relevance is assessed by computing the dot product of the image context vector with the query context vector. Large dot products correspond to close similarity, lower products to less similar images. Images will be ranked by dot product and are presented in a ranked list for easy identifications of images "close" to the user query. This process is shown inFigure 8.



Indexing of unlabeled imagery can also be achieved by using a database of human-labeled images as examples. The basic assumption is that "images that have similar content will have similar index terms". Thus, this approach relies on the context vector approach to find images with similar content. The block diagram for this operations is shown in **Error! Reference source not found..**

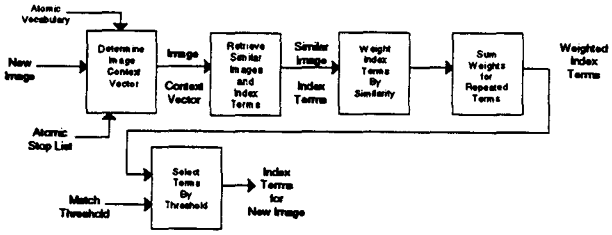

Atomic Vocabulary
New Image
Determine Image Context Vector
Image Context Vector
Retrieve Similar Images and Index Terms
Similar Image Index Terms
Weight Index Terms By Similarity
Sum Weights for Repeated Terms
Weighted Index Terms
Atomic Stop List
Match Threshold
Select Terms By Threshold
Index Terms for New Image

Figure 8. Image Retrieval.

Indexing of unlabeled imagery can also be achieved by using a database of human-labeled images as examples. The basic assumption is that "images that have similar content will have similar index terms". Thus, this approach relies on the context vector approach to find images with similar content. The block diagram for this operations is shown in **Error! Reference source not found..**

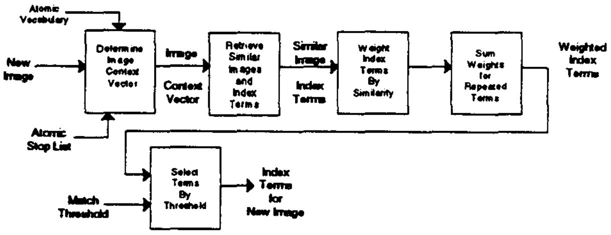

Atomic Vocabulary
New Image
Determine Image Context Vector
Image Context Vector
Retrieve Similar Images and Index Terms
Similar Image Index Terms
Weight Index Terms By Similarity
Sum Weights for Repeated Terms
Weighted Index Terms
Atomic Stop List
Match Threshold
Select Terms By Threshold
Index Terms for New Image

Figure 9. Indexing Mode Block Diagram.

1.4.2 ICARS Conclusion

A summary of attributes of the ICARS approach are as follows:

- Defines image content based on automatically derived atomic features.
- Retrieves unlabeled imagery based on learned proximate co-occurence of atomic features. ICARS ranks all images in the database according to similarity to query. As many images as desired are returned for any query.
- Provides fully automated generation of system. Given an atomic vocabulary and a set of images, generation of the ICARS system is automatic. No human knowledge base is needed for creation of the image retrieval portion of the system.
- Assigns index terms to unlabeled imagery based upon a set of human indexing examples.

1.5 Summary

As described in the preceding sections, the context vector approach is very powerful and can be applied to multiple information domains. The MatchPlus effort has demonstrated the viability of the context vector approach for text. The ICARS effort has provided a preliminary demonstration of the applicability of this approach for images. This foundation can provide the

[3] ICARS, in its current version, supports the concept of an atomic "stop list" where very high frequency atoms can be eliminated from subsequent computations. This is a direct extrapolation from the text case.

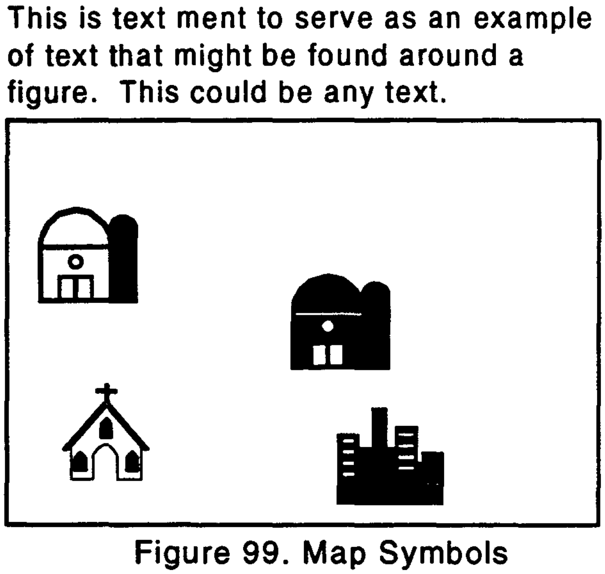
basis for a unified information representation. However, no work has been performed to date on a unified information space that contains context vectors associated with both text and image symbols. It is precisely this area of research that is believed to hold the key to a generic, media-independent information representation.

2. Approach to Integration of Text and Image Information Spaces

The previous sections describe how the context vector technique can be applied to learning relationships from text and images. In the text case the principles of quantized information vocabularies (stemming) and symbolic association are combined with proximate co-occurence to learn contextual similarity of usage at the stem level. These context vectors are then used to perform useful text processing tasks such as stem-stem relationship discovery, word sense identification, document retrieval and document clustering.

In the image case, the principles of quantized information vocabularies were applied to Gabor wavelet-based image feature vectors using a vector quantizer to produce a finite symbol set. These symbols were then associated with random context vectors prior to application of the context vector learning law. The resulting "image symbol" vectors can be used to form a context vector representation for images and can be used to perform image retrieval by similarity of content.

Many documents written and stored in today's information processing systems are mixed media. That is, they contain both free text and imbedded figures. This white paper is an example of such a document. A general characteristic of these documents are that the text that surrounds a figure, in general, describes or refers to the figure. As such, there is a proximate co-occurence relationship between the image symbols in a figure and the symbols in the surrounding text (stems). It is possible to develop a technique to learn the relationships between the image symbols and the symbols in the surrounding text. Since a context vector representation will be used for both text and image atoms, a common frame of reference is available for the learning process. An example of the proximate co-occurence between text and image symbols is shown below in Figure 10. This figure represents a segment of a mixed media document and has both "fake" text and an associated figure consisting of some map symbols.



This is text ment to serve as an example of text that might be found around a figure. This could be any text.

Figure 99. Map Symbols

Figure 10. Example Mixed-Media Document.

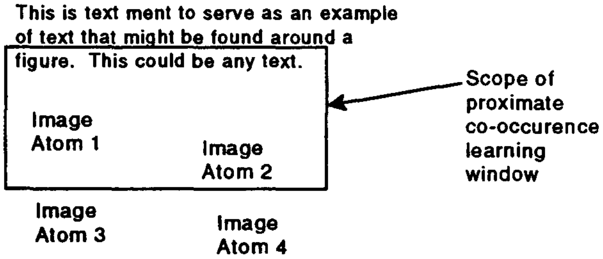

This is text ment to serve as an example
of text that might be found around a
figure. This could be any text.
Image
Atom 1
Image
Atom 2
Scope of
proximate
co-occurence
learning
window
Image
Atom 3
Image
Atom 4

Figure 11. Example Co-occurence Learning Window.

Figure 11 shows an example of a co-occurence learning window that spans both text and image symbols. In Figure, it can be seen that the map symbol icons have been replaced by their image symbols. Also, it can be seen that the size and scope of the learning window will impact which word and image symbol context vectors will be modified during the training.

As described above, both the text and image cases have a series of key similarities (see Figure). Indeed, the ICARS concept was developed by using analogies to extend the text approach to images. However, in both cases, the key information symbols are associated with context vectors prior to learning. To achieve the goal of mixed media learning, the following steps are required:

1. Identify a set of appropriate features for the image data and generate a set of image symbols. Use stems as the text symbols and quantized atomic features for image symbols.
2. Provide initial conditions for context vectors for both text and image symbols from the same context vector space.
3. Utilize a learning law to exploit proximate co-occurrence for both text and image symbols in documents. This learning algorithm would, by necessity, have to be robust enough to accommodate overlap conditions of mixed media symbols in the learning window. In other words, the relationships between the text symbols and

image symbols in the window will be learned based on spatial co-occurence within the learning window.

4. Exploit these relationships for document management tasks. Exploitation will require the development of a preliminary access and retrieval capability. However, the basic system architecture utilized for the MatchPlus and ICARS systems would be sufficient to provide this capability.

This technique will automatically learn and exploit relationships from mixed media documents, for both text and image components. As such, this approach can provide advantages in terms of the ability to manage these mixed media documents. Free text could be used as a query to the system and the result would be both text and images. Conversely, an image could be used as a query and both images and associated text would be retrieved. Many other advantages may exist, but due to the newness of this concept, they are not yet fully understood.

3. Summary and Conclusions

The techniques described above represent work in progress for several related research efforts. Many technical issues remain to be resolved. However, major progress has been made in developing a conceptual framework for the integration of text and images in the same information space. Efforts in these areas will continue as part of the DOCUVERSE and ICARS projects and a substantial number of "reportable results" are expected within the next 9 months.

Key areas and technical issues remaining for investigation are as follows:

1. More investigation of context vector learning laws that operate optimally in a mixed media environment are required. One vital issue is the development of a scoring algorithm to permit rapid evaluation of learning techniques.
2. Experiments must be conducted to determine the optimum "scope" of the learning window and the amount of allowable overlap between the image symbols and surrounding text.
3. Optimum feature vector configurations for mixed media environments need further research. Additionally, optimum vocabulary sizes need to be investigated.

Research efforts in this area have only just begun. However, the context vector technique shows true promise as demonstrated by the single media results embodied in both the MatchPlus and ICARS systems.

4. References

[1] Salton, G. (ed.), *The SMART Retrieval System - Experiments in Automatic Document Processing, Prentice-Hall* (1971).

[2] Salton, G., Another Look at Automatic Text Retrieval Systems, Communications of the ACM, Vol. 20, 1986, pp. 648 - 656.

[3] Salton, G., *Automatic Text Processing* (Addison-Wesley, 1989).

[4] Salton, G. Buckley, C. Term Weighting Approaches in Automatic Text Retrieval, Information Processing and Management, Vol. 24, No. 5, 1988, pp 513-523.

[5] Buckley, C., Allan, J., Salton, G., Automatic Routing and Ad-hoc Retrieval Using SMART: TREC-2, in Proceedings TREC-2 Conference, D. Harman, ed, Gaithersburg, MD. Aug. 1993.

[6] Deerwester, S., Dumais, S.T., et al, Indexing by Latent Semantic Analysis, Journal of the Society for Information Science, 1990, Vol 41, No. 6, pp 391-407.

[7] Dumais, S.T., LSI Meets TREC: A Status Report, in Proceedings TREC-1 Conference, D. Harman, ed, NIST Special Publication 500-207, Rockville, MD. Nov. 1992.

[8] Dumais, S.T., Latent Semantic Indexing (LSI) and TREC-2, in Proceedings TREC-2 Conference, D. Harman, ed, Gaithersburg, MD. Aug. 1993.

[9] Gallant, S.I., Context Vector Representation for Document Retrieval, AAAI-91 Natural language text Retreival Workshop, Anaheim, CA. July 1991.

[10] Gallant, S.I., A Practical Approach for Representing Context and Performing Word Sense Disambiguation Using Neural Networks, Neural Computation, Vol. 3, No. 3, 1991, pp 293-309.

[11] Daugman, J, "Six Formal Properties of Two-Dimensional Ansiotropic Visual Filters: Structural Principle and Frequency/Orientation Selectivity", IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-13, p.p. 882-887, Sept. 1983.

[12] Daubechies, I, "Time-frequency Localization Operators: A Geometric Phase Space Approach", IEEE Transactions on Information Theory, Vol. 34, p.p. 605-612, July 1988.

[13] Mallat, S.G. "Multifrequency Channel Decompositions of Images and Wavelet Models", IEEE Transactions on ASSP, Vol. 37, p.p. 2091-2110, Dec. 1989.

[14] Mallat, S.G., "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation", IEEE PAMI, Vol. 11, p.p. 674-693, July, 1989.

[15] Porat, M., Zeevi, Y., "The Generalized Gabor Scheme of Image representation in Biological and Machine Vision", IEEE PAMI, Vol. 10, p.p. 452-468, July 1988.

[16] Watson, G.S. "Statistics on Spheres", Wiley, p.p. 40-43, 1983

[17] Daugman, J.G., "Uncertainty Relation for Resolution in Space, Spatial Frequency and Orientation Optimized by Two-Dimensional Visual Cortical Filters", J. Optical Soc. Am., Vol. 2, p.p. 1160-1169, 1985.

Combining Text and Image Information in Content-Based Retrieval

Rohini K. Srihari[*]
Center for Document Analysis and Recognition (CEDAR)
State University of New York at Buffalo
Buffalo, NY 14228-2567
rohini@cedar.buffalo.edu

Abstract

This paper explores the interaction of textual and photographic information in an integrated text/image database environment. Specifically, we have developed an automatic indexing system for captioned images of people (i.e., human faces) where groups can consist of one or more members; the indexing information along with other textual information is subsequently used in a content-based image retrieval system. Our approach is unique since it goes beyond a superficial combination of existing text-based and image-based approaches to information retrieval. By understanding the caption accompanying a picture, we are able to (i) extract information useful in retrieving the picture and (ii) extract information useful in directing an image interpretation system identify relevant objects (in this case, faces) in the picture. A multi-state system, PICTION, which uses captions to identify human faces in an accompanying photograph has been developed. We discuss the use of PICTION's output in content-based retrieval of images to satisfy focus of attention in queries.

[*]This work was supported in part by ARPA grant 93-F148900-000.

DocBrowse: A System for Mixed Graphics/Text Document Image Analysis and Retrieval

M. Y. Jaisimha, Jeremy Worley, Andrew Bruce and Emmanuel Arbogast
StatSci Division of MathSoft, Inc.,
1700 Westlake Ave. N, Suite 500,
Seattle, WA 98109.

Abstract

This paper presents the software architecture for DocBrowse: a system for mixed text/graphics document image analysis and retrieval. DocBrowse is intended to be an open and extensible environment that permits the user to manage and perform queries on highly degraded document image databases while also serving as a research environment for developing document image analysis and query by image example (QBIE) algorithms. DocBrowse consists of a user interface, an object oriented document database and a variety of document image analysis engines. Using DocBrowse, it is possible to perform queries that retrieve documents based on both graphical and textual content. We describe the graphical user interface that is used to perform such queries. We also describe briefly our approach to QBIE along with some example results on a large test database of document and logo images that we have developed. We conclude with a description of the document database and the analysis engines used in the current prototype.

1 Introduction

Digital documents are a fixture in modern office working environments. Digital documents come in many different forms, ranging from simple ascii text files to complex compound documents stored in a special formats involving both text and graphics. With the advent of word processors, scanners, and FAX machines, and the drop in the cost for digital data storage, massive numbers of digital documents are being acquired, analyzed, and stored. When the document is acquired by a scanner or FAX machine, the digital image will contain at least some noise and may be highly degraded. Analyzing and managing this massive flow of information remains a major challenge.

Several commercial and public domains systems have been developed expressly for the problem of management and analysis of images. These include Excalibur EFS, Visual Recall from XSoft, PageKeeper from Caere among others. These systems usually feature sophisticated techniques for document analysis problems such as page segmentation and optical character recognition (OCR). They offer state-of-the-art tools and techniques for document analysis and management, and some achieve excellent performance in dealing with clean (low-noise) digital image data. All of the existing systems, however, suffer some of the following limitations:

- The primary orientation is for analysis of digital documents based on roman text. Non-roman text and graphics is supported in varying degrees, and often not at all.
- The performance is seriously impaired for very noisy or highly degraded documents, sometimes to the point of being useless.
- No single system is omnipotent, it is usually necessary to develop specialized interfaces between multiple applications.
- Many systems have a closed software architecture and do not allow the user to extend the system.

The aim of our project is to overcome these limitations in the development of DocBrowse, a software system for managing large collections of complex digital documents which have both text and graphics and may be subject to serious degradations. DocBrowse is intended to be used both as a research test-bed for document image analysis algorithms as well as an on-line mixed mode document query interface.

The overall system architecture for DocBrowse is displayed in figure 1.

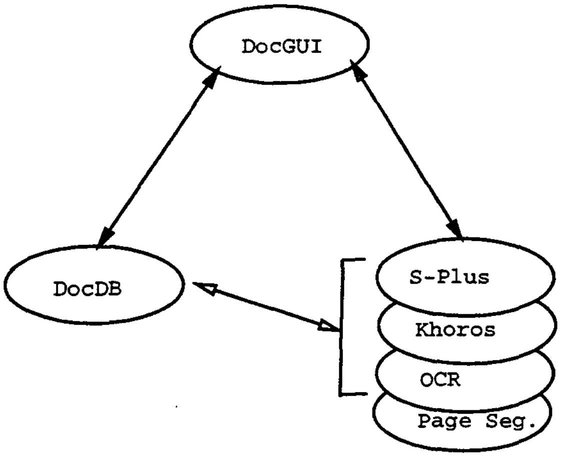


Figure 1: DocBrowse Architecture

DocBrowse consists of three main components

1. A browser/GUI for visual querying and sifting through a large digital document image database. Support is provided for queries based on both textual and graphical information.
2. An object-relational database management system for storing the data.
3. Several analysis engines, including specialized document analysis software (OCR and page segmentation) and general purpose quantitative programming environments.

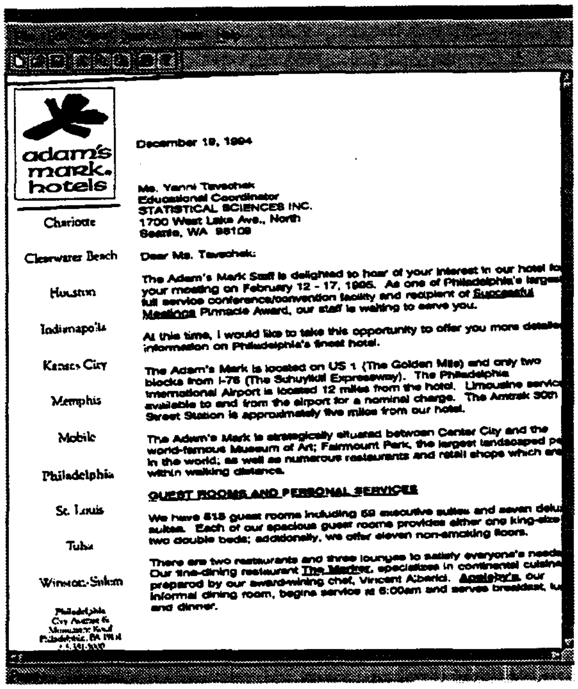
An important feature of DocBrowse is an open architecture with interchangeable software components. This permits the user to extend the system in various ways. For example, the user can reconfigure the user interface, add additional databases of documents, add additional algorithms to existing document image analysis engines that are part of the system, add user-defined or derived attributes to new or existing documents, or even add other analysis engines. Document analysis software is highly specialized and rapidly evolving. Consequently, extensibility and flexibility is the key to development of a next generation software system.

In this paper we give an overview of DocBrowse and its software architecture. Section 2 presents DocBrowse from a "user's viewpoint", describing the user interface and presenting several scenarios of common usage. One scenario describes how to perform query by image example (QBIE) using logos on business letters as cues. In section 3, we describe the methodology used to perform QBIE in DocBrowse. Section 4 discusses the object relational database management system and the schema for the document image data. Section 5 describes the various components that make up the analysis engines of DocBrowse. Future directions are discussed in section 7.

2 A User's View of DocBrowse

Document image management and analysis systems are often used in scenarios where the user would like to browse, retrieve and analyze large numbers of highly degraded scanned document images. These collections of documents and the user who is browsing these collections could reside on various computers connected in a network. The document images in the database could contain both textual and graphical components. Graphical components include all regions of the image that would not ordinarily be readable by an OCR for e.g. signatures, logos, line drawings etc. Each document in the database is associated with a variety of attributes both derived (extracted from the document image in some automatic fashion) and user-defined. Derived attributes could include - OCR'ed text, page layout, a segmentation of the page into zones, signatures associated with graphical/non-text zones etc. Given these tags it is then possible to compose relatively complex queries that retrieve documents based on a combination of document attributes. The query mechanisms themselves are sophisticated enought to allow the user to iteratively refine the scope of a search on a large collection of documents.

Figure 2 shows an overall view of the GUI for DocBrowse.



adam's mark. hotels

Charlotte

Clearwater Beach

Houston

Indianapolis

Kansas City

Memphis

Mobile

Philadelphia

St. Louis

Tulsa

Winston-Salem

Philadelphia
City Avenue &
Monument Road
Philadelphia, PA 19131
215.581.5000

December 19, 1994

Ms. Yanni Tavechek
Educational Coordinator
STATISTICAL SCIENCES INC.
1700 West Lake Ave., North
Seattle, WA 98109

Dear Ms. Tavechek:

The Adam's Mark Staff is delighted to hear of your interest in our hotel fo your meeting on February 12 - 17, 1995. As one of Philadelphia's large full service conference/convention facility and recipient of Successful Meetings Pinnacle Award, our staff is waiting to serve you.

At this time, I would like to take this opportunity to offer you more detaile information on Philadelphia's finest hotel.

The Adam's Mark is located on US 1 (The Golden Mile) and only two blocks from I-76 (The Schuylkill Expressway). The Philadelphia International Airport is located 12 miles from the hotel. Limousine servic available to and from the airport for a nominal charge. The Amtrak 30th Street Station is approximately five miles from our hotel.

The Adam's Mark is strategically situated between Center City and the world-famous Museum of Art; Fairmount Park, the largest landscaped p in the world; as well as numerous restaurants and retail shops which ar within walking distance.

GUEST ROOMS AND PERSONAL SERVICES

We have 515 guest rooms including 59 executive suites and seven delu suites. Each of our spacious guest rooms provides either one king-size two double beds; additionally, we offer eleven non-smoking floors.

There are two restaurants and three lounges to satisfy everyone's need Our fine-dining restaurant The Marker, specializes in continental cuisin prepared by our award-winning chef, Vincent Alberici. Appleby's, our informal dining room, begins service at 6:00am and serves breakfast, lu and dinner.

Figure 2: Using DocBrowse

DocBrowse offers three modes for the user: 1) view and analyze, 2) insert, and 3) query. All operations in each of the three modes are supported by the GUI in an intuitive and easy to use manner. These modes are described in detail below.

2.1 View and Analyze Mode

In the first mode (which we call the *view and analyze* mode) the user scans, views and analyzes document images. The user is able to pan and zoom on document images. The user can invoke an OCR to produce text output, use page segmentation software to segment the page into text and non-text regions, use any other analysis algorithms present to operate on the image. These will eventually include skew detection and correction algorithms, and denoising algorithms. In view and analyze mode, the user will also be able to compute signatures/features on segmented zones and store these feature vectors in the database. These can be then used to create a training/test set for a classifier that is part of S-Plus or any other analysis engine.

2.2 Insert Mode

In the second mode (which we call *insert* mode), the user inserts a document page (or pages) into DocDB. The various steps the user performs are - scan the page, define values for pre-existing document attributes, define new document attributes, insert into database. Pre-existing document attributes include page text, segments, signatures for QBIE on graphics zones or segments, pixmap resolution, author etc. If certain attributes apply to all documents the user sets up session-wide attribute-value pairs that are automatically associated with all documents inserted in that insertion session. Examples of such session-wide attributes or tags include scan resolution, insertion person etc. User defined attributes can be either numerical, textual or logical entities.

2.3 Query Mode

In the third mode (which we call *query* mode) the user uses the attributes of the documents to retrieve and visualize documents in the database. The user can query in a combination of three different ways. Figure 3 shows the DocBrowse query manager interface that allows the user to combine various types of queries. The documents returned by the query are thumbnailed and presented to the user and appear below the query manager window in the figure. The scale at which the thumbnailing is performed is determined by a combination of the screen resolution and the number of documents returned by the query.

The first query sub-mode called the *tag query* sub-mode allows the user to query on attribute value pairs. For example a tag query can take the form "Retrieve all documents that have a tag called `author` with a value `Andrew Bruce`." These queries are entered via a check-box GUI interface. In the second query sub-mode (called the *text query*) sub-mode the user queries on text keywords that are present in the OCR text of the document pages in the database. Key words can be combined in a boolean fashion. Documents are returned rank-ordered in decreasing similarity with the query words. The measure of similarity used in the query engine is the well-known and accepted

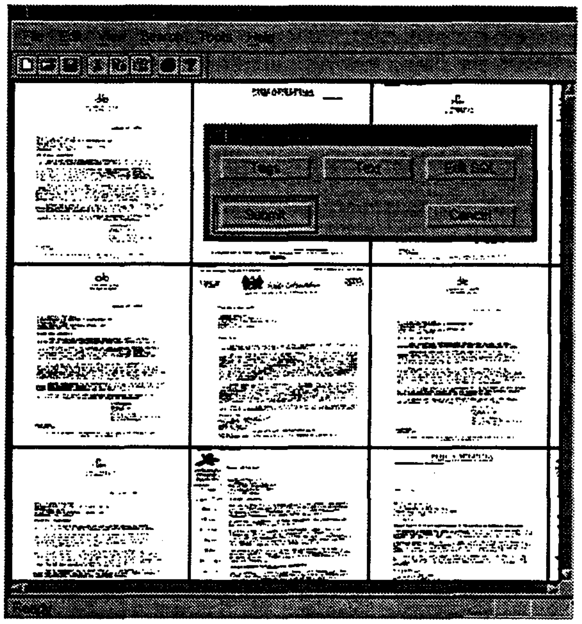
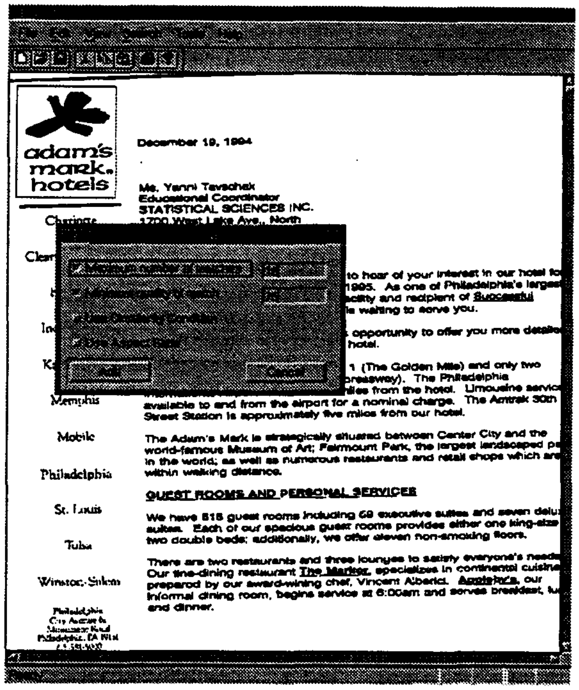


Figure 3: Querying using DocBrowse

cosine distance measure. The third query sub-mode (called the *QBIE* sub-mode) allows the user to select a graphical zone on a page and search for all other images with similar graphics regions. The query interface for QBIE searches is shown in Figure 4. The user selects the document graphic that is to be used for the query. A variety of signature representations are pre-computed and stored with each document in the database and it is possible to use one or a combination of signatures in performing the query. Documents containing graphics zones similar to the zone of interest are returned in decreasing order of similarity. Figure 5 shows the results of one such query.

3 Graphical Queries on Document Images

There are a large number of ongoing research efforts that tackle the problem of query by image example. IBM's QBIC system [10] and Illustra's Visual Intelligence Recall product [5] both address the problem of querying for color photographic images based on attributes of these images. In [8], the authors describe a retrieval system for graphical documents, principally line drawings of machine parts. [6] de-



adam's mark. hotels

December 19, 1994

Ms. Yanni Tevschek
Educational Coordinator
STATISTICAL SCIENCES INC.
1700 West Lake Ave., North

to hear of your interest in our hotel fo
1995. As one of Philadelphia's larges
cility and recipient of Successful
is waiting to serve you.

opportunity to offer you more detaile
hotel.

1 (The Golden Mile) and only two
ressway). The Philadelphia
miles from the hotel. Limousine servic
available to and from the airport for a nominal charge. The Amtrak 30th
Street Station is approximately five miles from our hotel.

The Adam's Mark is strategically situated between Center City and the world-famous Museum of Art; Fairmount Park, the largest landscaped pa in the world; as well as numerous restaurants and retail shops which ar within walking distance.

QUEST ROOMS AND PERSONAL SERVICES

We have 515 guest rooms including 69 executive suites and seven delu suites. Each of our spacious guest rooms provides either one king-size two double beds; additionally, we offer eleven non-smoking floors.

There are two restaurants and three lounges to satisfy everyone's need Our fine-dining restaurant The Marker, specializes in continental cuisin prepared by our award-wining chef, Vincent Alberici. Appleby's, our informal dining room, begins service at 6:00am and serves breakfast, lu and dinner.

Memphis
Mobile
Philadelphia
St. Louis
Tulsa
Winston-Salem

Philadelphia
City Avenue &
Monument Road
Philadelphia, PA 19131

Figure 4: Graphical queries using DocBrowse

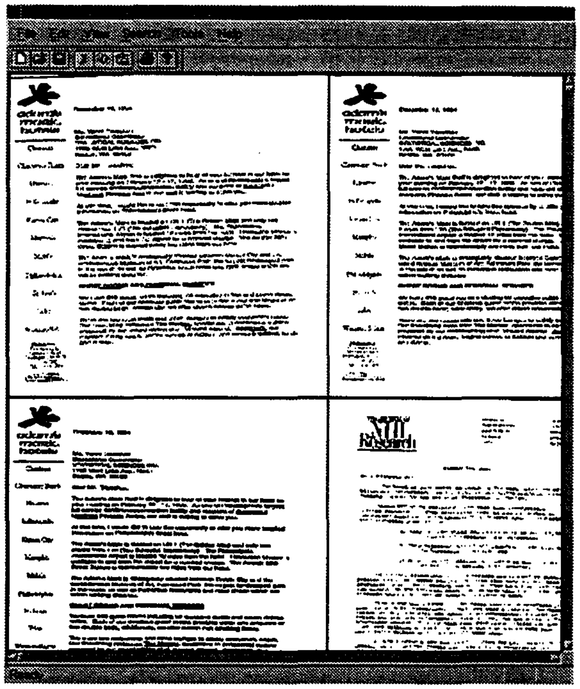


Figure 5: Graphical query results in DocBrowse

scribes a system that uses wavelet transform based methods for sketch based retrieval of digitized color paintings. As mentioned earlier, unlike these systems DocBrowse supports graphical queries on on highly degraded document images.

Querying based on graphical content is an intrinsically difficult problem. The search is not based on matches or partial matches, but instead on a measure of similarity. Similarity searching is often based on some statistical technique (e.g., linear discriminant analysis or nearest neighbor analysis). In classical statistical classification/discrimination, the dimension of the space to search (p) is relatively small. In graphical querying, the dimension is enormous: $p = 10^6$ or 10^7 and traditional statistical techniques are neither practical nor appropriate.

Our underlying approach towards graphical querying is based on reducing the dimension of the problem by computing a variety of signatures from each of the graphics zones (logos) present on document pages. These signatures are designed to be robust to noise, rotation, translation and scale changes in the data. A variety of signatures (shape descriptors) have been proposed for 2-d object recognition and binary machine vision [11]. Many of these techniques that are robust with respect to noise are based on Fourier shape descriptors. Recently, wavelet transforms [2] have been shown to provide certain advantages over Fourier transforms in terms of better signal compaction for a given signal to noise ratio. With the exception of recent work [3] that deals with overcomplete wavelet representations, wavelet transforms are not rotation or translation invariant. Our approach proposes to overcome this problem while retaining the other advantages of wavelet representations. Details of the procedure are given in [7].

The algorithms for signature computation proceed in three separate stages:

1. Filter to minimize the impact of noise.
2. Compute a rotation, translation and scale invariant representation of the logo. This procedure assumes that the page orientation is upright when the document image is scanned.
3. Compute three signatures for each logo
   - Wavelet transforms of projections in the two axes directions.
   - Wavelet transforms of a parametric representation of the bounding contour of each logo.
   - Two-dimensional wavelet transforms.

In order to describe the capabilities of the system as it stands, we present a few sample results.

We now show three examples to illustrate the performance of various aspects of our QBIE algorithms.

In the first example, the test database consists of 4 versions of each of 100 logos - digitized at 300 dpi and subsampled versions at 200, 150 and 100 dpi. Figure 6 shows the result of a query logo and the top eight logos returned by DocBrowse using the wavelet boundary signature. The top left hand corner is the query logo at 100 dpi. The remaining 7 (top-to-bottom, left-to-right) are the logos returned in descending order of Euclidean distance from query vector in the signature space.

In the second example, the test database consists of 12 versions of each of 73 logos - rotated at eight different angles between +60 and -60 degrees about the vertical. (Typical skew angles in document images are in the range of less than 5 degrees) Figure 7 shows the result of a query logo and the top ten logos returned by DocBrowse using the wavelet projection signature.

In the third example, the test database consists of 5 versions of each of 500 logos contaminated by shot-noise ranging between 1 and 5% of the pixels. Figure 8 shows the result of a query logo and the top ten logos returned by DocBrowse using the two-dimensional wavelet transform signature.

4 Document Database

The document structure in DocBrowse is based on several existing standards for document description and layout including the ISO Office Document Architecture standard [1], SGML, DAFS from RAF Technology [12], and RTF [9]. Figure 9 illustrates the structure of a document in DocBrowse. A *document* is defined as a collection of *document pages*. Document pages are in turn decomposed into *zones*. The decomposition of pages into zones operates at multiple granularities. At the coarsest level, the page can be decomposed into *header*, *footer* and *live matter* zones. At the finest level of granularity each character on the page can be considered a zone. At

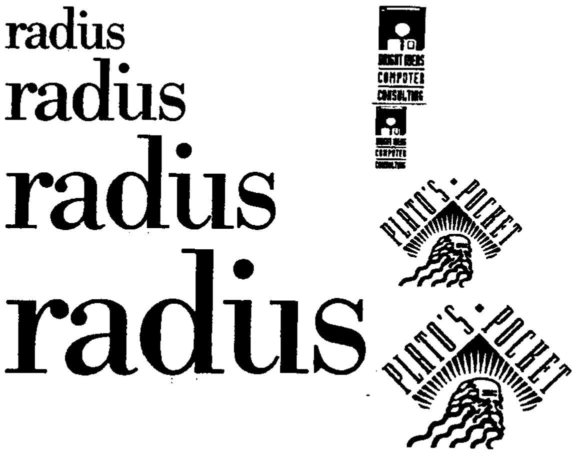


Figure 6: Effect of scan resolution

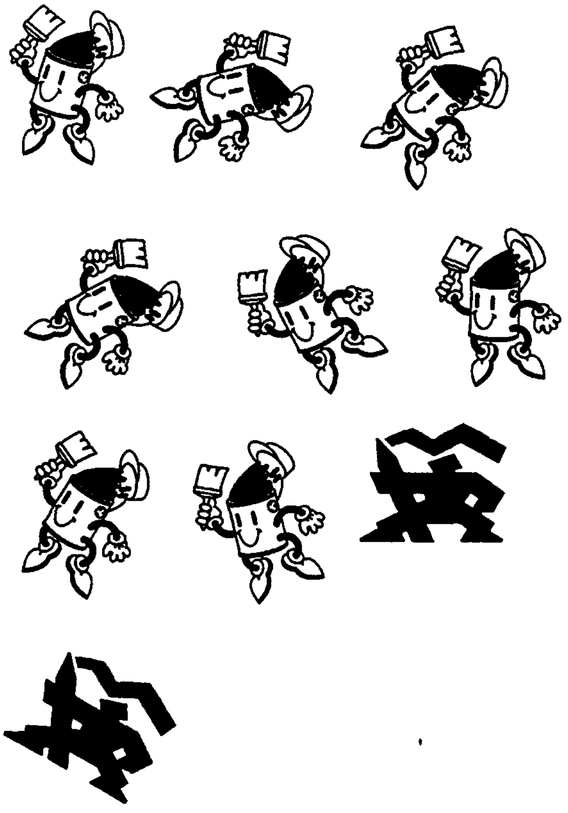


Figure 7: Effect of rotation

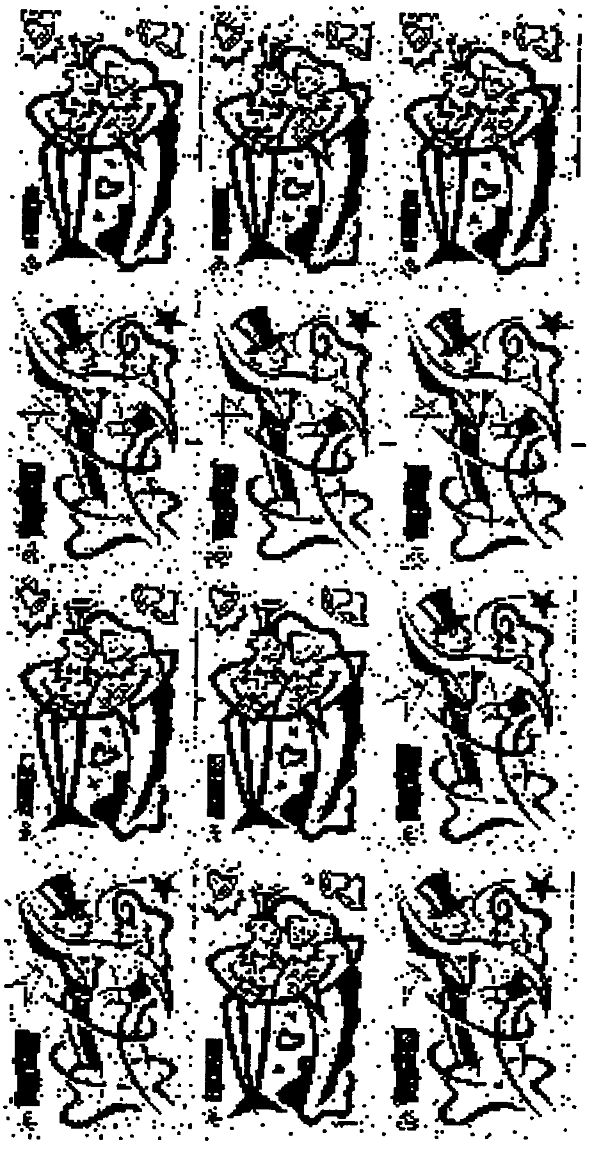


Figure 8: Effect of noise

an intermediate level of granularity each paragraph or body of text which is distinctly separated from adjoining bodies of text or figures can be referred to as a zone. It is this level of granularity that we adopt for this project. Zones can be of two types - text and non-text or graphics zones. A graphics zone contains information such as figures, line drawings, half-tones or bit-maps (such as logos). Each document, document page and zone can have associated with it one or more tags in the form of attribute value pairs. Specifically, these tags could include in the case of a document page, the scanned pixmap of a page, the type of document, scan resolution, OCR text etc. In the case of a zone it could include a processed pixmap of the zone or features extracted from the zone that could be used for zone classification or classifier construction. The focus of the DocBrowse system is to permit the user to interact with and process a large set of documents all the while adding to what she/he knows about these documents by using various visualization and processing tools.

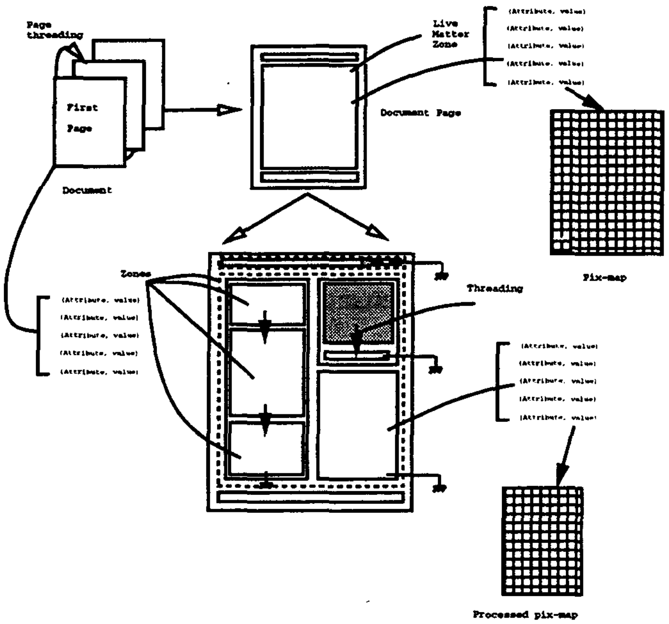


Figure 9: Documents in the DocBrowse

The database component (DocDB) of this project is implemented using an object-relational database management system (ORDBMS) called *Illustra*, which is produced by Illustra Information Technologies, Inc. [4]. ORDBMS have the advantages of both relational DBMS (RDBMS) and pure object-oriented DBMS (OODBMS) in that unlike an RDBMS they support complex datatypes and inheritance mechanisms and unlike an OODBMS they have a sophisticated query language (SQL3 in this case). In addition it is also possible to extend the database server using "datablades" that support query and computational operations on specialized datatypes. For example, the text datablade incorporates support for a variety of operations on text data that are used to extend the functionality of DocBrowse. These capabilities included the ability to construct and maintain inverted full-text indices from document collections, and a weighted boolean information retrieval engine for text based retrieval. It is possible to easily extend the database with any other IR engine/algorithm that operates on the same set of text data.

5 Analysis Engines

DocBrowse utilizes a variety of document image analysis algorithms to compute derived attributes of documents from their image pixmaps. These algorithms are based on two types of software:

- Specialized "off-the-shelf" document analysis software.
- Customized routines typically written in a language supported by a quantitative programming environment (QPE).

The prototype of DocBrowse integrates the ScanWorx OCR API from Xerox Imaging Systems [14]. This OCR is used to extract text from document images which is then stored in DocDB via the Illustra Text datablade.

The prototype also incorporates S-Plus and Khoros, two general QPE's for scientific computing. Background on these systems, and why they are valuable for document analysis, is given below.

5.1 S-Plus

In order to meet the requirements of providing an extensible environment for image analysis, feature extraction and classifier construction one of the analysis engines used in the DocBrowse prototype is the

S-Plus data analysis environment from the StatSci Division of MathSoft Inc. S-PLUS is a system with extensive capabilities for data analysis and scientific computing. It originates from the S language developed by researchers at AT&T Bell Labs. It is an interactive interpreted language with the following attractive features:

- S-PLUS is a versatile matrix calculator, drawing its inspiration from the language APL.
- S-PLUS has a built-in transparent database mechanism to make objects persistent, maintaining data ands results across sessions.
- S-PLUS supports "Lisp-like" data abstractions, providing considerably more flexibility than conventional array based interactive languages.
- S-PLUS supports both object oriented programming and functional programming.
- S-PLUS treats procedures as data and permits creation of new procedures, making the system fully extensible.
- S-PLUS makes it easy to interface to existing Fortran and C code.
- S-PLUS has powerful and flexible interactive graphics, and can produce publication-quality plots.
- S-PLUS has an extensive collection of over 2000 statistical and mathematical functions providing a broad range of capabilities, such as clustering analysis, nonlinear regression, time series analysis, and optimization.

S-Plus is similar to other scientific computing systems, such as MATLAB or Mathematica. What sets S-Plus apart from other systems is its support for object-oriented programming, complex data structures, and its extensive collection of functions devoted to data analysis and statistics.

In addition, S-Plus has been extended with S+WAVELETS which is an object-oriented toolkit for wavelet analysis of signals, time series, images, and other technical and scientific data. In addition to the basic wavelet decomposition, S+WAVELETS also includes sophisticated time-frequency decompositions such as a wavelet packets and cosine packets. We use the S+WAVELETS toolkit extensively in developing signatures for QBIE.

5.2 Khoros

In addition, we also use the Khoros image analysis environment [13] from Khoral Research, Inc. to develop image analysis algorithms. Khoros is a scientific data analysis and software development environment which has a strong following in the image processing and signal processing communities. Khoros offers an extensive image processing and image analysis capability and is capable of operating on large document images in a variety of file formats. As part of the DocBrowse project we propose to implement an interface between Khoros and S-Plus that will allow us to exchange data easily between the two and use S-Plus as a scripting language for Khoros. So for example, we can use Khoros to perform image feature extraction and use the extensive library of classification routines available in the S-Plus environment to develop a zone classifier.

6 Digital Document Data Testbeds

To evaluate DocBrowse, we have generated a database of 1500+ logos which have been degraded in different ways (combinations of photocopying and FAX transmission) for a total of 6000+ logos. We also have a database of 200 letters degraded in five different ways - faxing, faxing and photocopying, first generation photocopies, second generation photocopy with normal contrast and second generation photocopy with dark contrast - that we use for our experiments. This database continues to grow to meet our experimental needs. In addition to "real" degradation, we also multiplied our logo database by introducing various artificial degradation sources and studying how well our QBIE algorithms perform (as discussed in Section 3). Figure 10 shows a few example logos from the logo database. They include logos that have been photocopied and FAXed prior to being scanned. We are exploring the possibility of making this database available to interested researchers via anonymous ftp.

7 Future Extensions

In future generations of DocBrowse, we propose to use the concept of abstract services in a message

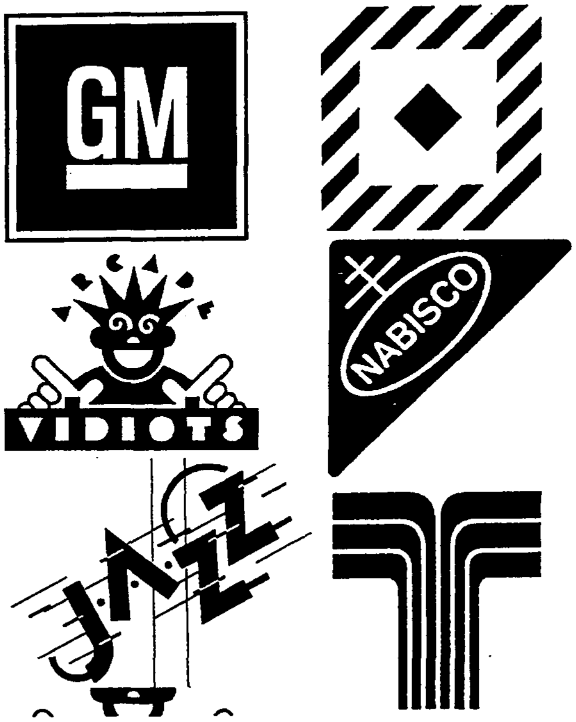


Figure 10: Example logos in Logo Database

passing context to allow the integration of arbitrary analysis engines. We will also incorporate object request broker technology to enable interchange of data between heterogeneous platforms and software packages.

In terms of user functionality future versions of DocBrowse will incorporate spelling variations and equivalences (synonymy) between tags. For example **author** and **writer** and **auther** would all be interpreted as the same tag. Document similarity searches will also be supported. We also propose to incorporate topological information into the signatures as well as use intelligent strategies to combine results from various signature spaces. Scalability (with respect to database size) of QBIE algorithms is desirable. We propose to explore mechanisms for index construction and database clustering in signature space to reach this goal.

References

[1] Wolfgang Appelt. *Document Architecture in Open Systems: the ODA standard.* Springer-Verlag, 1991.

[2] Andrew Bruce and Hong-Ye Gao. *S+ Wavelets Users Manual.* StatSci Division of MathSoft, Inc., Seattle, WA, 1994.

[3] R. L. Coifman and D. L Donoho. Translation invariant denoising. Technical report, Stanford University, 1995.

[4] Illustra Information Technologies, Inc. *Illustra Users Manual*, 1994.

[5] Illustra Information Technologies Inc. Illustra announces visual intelligence recall datablade, July 1995.

[6] C. E. Jacobs, A. Finkelstein, and D. Salesin. Fast multiresolution image querying. In *Computer Graphics Proceedings.* ACM SIGGRAPH, 1995.

[7] M. Y. Jaisimha. Similarity based retrieval of degraded logo images. StatSci Division of MathSoft, Inc.

[8] O. Lorenz and Gladys Monagan. A retrieval system for graphical documents. In *Proceedings of UNLV Symposium on Document Analysis and Information Retrieval*, pages 291–300, 1995.

[9] Microsoft Corp. *Rich Text Format Specification*, version 1.2 edition.

[10] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. The qbic project: Querying images by content using color, texture and shape. *Proceedings of SPIE Storage and Retrieval for Image and Video Databases*, pages 173-187, 1993.

[11] W. K. Pratt. *Digital Image Processing.* John Wiley and Co., 1991.

[12] RAF Technology, Inc., Redmond, WA. *Document Attribute Format Specification*, 1995.

[13] J. Rasure and S. Kubica. The khoros application development environment. In H. I. Christensen and J. L. Crowley, editors, *Experimental Environments for Computer Vision and Image Processing.* World Scientific, 1994.

[14] Xerox Imaging Systems, Inc. *ScanWorx API: Users Manual*, 1994.

Page Decomposition

An Architecture for Document Understanding

Al Alli
Department of Defense
9800 Savage Road
Ft. Meade, MD 20755-6000

Document Structural Decomposition

Robert M. Haralick
Su Chen Jaekyu Ha
Electrical Engineering FT-10
University of Washington
Seattle, WA 98115

Ihsin Phillips
Computer Science and
Software Engineering
Seattle University
Seattle WA

Abstract

The first level of document structural decomposition consists of zone delineation and zone classification. The purpose of zone delineation is to define the regions which cover the page such that each region, excluding its holes, (the regions which it contains), is entirely a text region or a non-text region and such that in each text region no text line is fragmented. After zone delineation, each text zone is then further decomposed into text lines and words. The purpose of zone classification is to identify each delineated zone into one of the classes: text or non-text. Non-text regions may be further classified into line drawing, graph, half-tone, etc. Zones which have been classified as text regions can then be given to an OCR system to produce computer readable text for the zone.

We have examined doing zone delineation by means of dynamic thresholding of recursively computed morphological open and closing transforms. We also have done zone delineation by grouping of the bounding boxes of the connected components of the black pixels. And we have performed zone classification on the basis of first and second line length moments taken in four directions of the black and the white pixels in each zone. Text vs non-text classification accuracy was above 97% on a database of over 12,000 zones.

1 Introduction

The first step of zone delineation consists of determining the geometric page layout of a document image page. This means a specification of the geometry of the maximal homogeneous regions and the spatial relations of these regions. A region is homogeneous if all its area is of one type: text, half-tone, figure, etc. and each text line of the page lies entirely within some text region of the layout. A Manhattan page layout is one where the regions of the page layout are all rectangular and the rectangles are in the same orientation. Hence after an appropriate page rotation the sides of the rectangles will all be either horizontal or vertical. Furthermore, each pair of rectangles either is mutually exclusive or one contains the other.

The purpose of zone delineation is to define the regions which cover the page such that each region, excluding its holes, (the regions which it contains), is entirely a text region or a non-text region and such that in each text region no text line is fragmented. Then within each text region the text lines and text words within the lines must be delineated. The purpose of zone classification is to identify each delineated zone into one of the classes: text or non-text. Non-text regions may be further classified into line drawing, graph, half-tone, etc. Zones which have been classified as text regions can then be given to an OCR system to produce computer readable text for the zone.

We have examined doing zone delineation by means of dynamic thresholding of recursively computed morphological open and closing transforms. And we have performed zone classification on the basis of line length moments taken in four directions of the black and the white pixels in each zone. Text vs non-text classification accuracy was above 97% on a database of over 12,000 zones.

2 Literature Review

Early work on zone delineation was done by Wahl et al. (1982). They use a technique called the constrained run length smoothing algorithm. It actually consists of a morphologically closing of the document image with a horizontal structuring element of specified length (they used 300) intersected with a morphological closing of the document image with a vertical structuring element of specified length (they used 500). The intersection is then morphologically closed with a horizontal structuring element of specified length (they used 30). The bounding rectangles of the connected components of the resulting image constitute the block segments. Features of the blocks include the area of the connected component of the block, the number of black pixels in the block on the original document image. the mean horizontal black run lengths of the original image within the blocks, and the height and width of the bounding rectangle of the block. Text areas are classified into *text, horizontal solid black lines, graphic and halftone im-*

ages, and *vertical solid black lines*. No measure of performance is given.

Nagy and Seth (1984), and Nagy et al. (1986) employ an X-Y tree as the representation of a page layout. The root node of an X-Y tree is the bounding rectangle of the full page. Each node in the tree represents a rectangle in the page. The children of a node are obtained by subdividing the rectangle of the parent node either horizontally or vertically, with horizontal and vertical cuts being alternately employed in successive levels in the tree. Hao et al. (1993) describe a variation on this technique.

Fisher et al. (1990) sample a 300dpi document image by a factor of 4 and use the run length smoothing algorithm. They then compute the connected components of the run length smoothed image. The connected components and their bounding boxes constitute the blocks of the geometric page layout. They extract connected component features such as component height, width, aspect ratio, density, perimeter, and area for classifying each block as *text* or *non-text*.

Lebourgeois et al. (1992) sample the document image by a factor of 8 vertically and 3 horizontally. Each pixel on the sampled image corresponds to an 8×3 window on the original image. If any pixel on the 8×3 window of the original image is a binary one then the sampled image has a binary one in the corresponding pixel position. Then the sampled image is dilated by a horizontal structuring element to effectively smear adjacent characters into one another. Each connected component is then characterized by its bounding rectangle and the mean horizontal length of the black runs. Connected components having a vertical height within given bounds and mean horizontal run length within given bounds are then labeled as a text lines and outside the given bounds are labeled as a non-text lines. Components assigned as text regions are then vertically merged into larger blocks using rules taking into account alignment. Blocks are also subdivided to separate their horizontal peninsulas. No measure of performance is given but an indication that the method needs improvement was stated.

Bloomberg (1991) uses morphological operations on the document image at various resolutions to determine identify font style for each word. Class labels include *bold*, *italic*, and *normal*. The method employs a small vertical dilation followed by a close open sequence to remove noise followed by a hit and miss transform to identify seed points of characters in the italic class or bold class. Then the words which are in italic or bold can be delineated by conditionally dilating the seed with a precalculated word segmentation mask. No accuracy performance results are given.

Saitoh and Pavlidis (1992) proceed sampling by 8 vertically and 4 horizontally and then extracting the connected components. They then classify each component into *text*, *text or noise*, *diagram or table*, *halftone image*, *horizontal separator*, or *vertical separator*, using block attributes such as block height, height to width ratio, and connectivity features of the line adjacency graph, and whether there are vertical or horizontal rulings. Page rotation skew is estimated from a least squares line fit to the the center points of blobs belonging to the same block. Blocks are subdivided based on the vertical distance between lines in a block, and the height of the lines in a block. The technique was tried on 52 Japanese documents and 21 English documents. No quantitative measure of performance was given.

Pavlidis and Zhou (1991) determine the geometric page layout by analyzing the white areas of a page by computing the vertical projection and looking for long white intervals from the projections. Then the column intervals are converted into column blocks, merging small blocks into larger blocks. Blocks are clustered according to their alignments and the rotation angle estimated for each cluster. The column blocks are then outlined. Finally, each block is labeled as text or non-text using features such as ratio of the mean length of black intervals to the mean length of white intervals, the number of black intervals over a certain length, and the total number of intervals. No performance results are given.

Baird (1992) discusses a computational geometry technique for geometric page layout by finding the maximal rectangles covering the white ares of the page. The rectanglar regions not covered by the maximal sized white rectangles then constitute regions which can then be classified as text or nontext.

Amamoto et al. (1993) determine the geometric page layout by operating on the white space of the sampled document image. They open the white space of the sampled document image tith a long horizontal structuring element and open it with a long vertical structuring element. The union of these two openings then constitute the white space of the blocks. The blocks are then extracted from this white space. They decide that a block is a text block if the length of the longest black run length in the vertical and horizontal directions is smaller than a given threshold. A decision is made whether the writing is horizontal or vertical based on the number N_H of blocks whose width is greater than twice its height and the number N_V of blocks whose height is greater than twice its width. If $N_H > N_V$, then the decision is horizontal writing. Else vertical writing. Each block is then assigned a class label from the set: *text, figure, image, table*, and *separation line*. No performance results are given.

O'Gorman (1992) discusses what he calls the docstrum technique for determining geometric page layout. This technique involves computing the k-nearest neighbors for each of the black connected components of the page. Each pair of nearest neighbors has an associated distance and angle. By cluster the components using the distance and angle features, the geometric regions of a page layout can be determined.

Hirayama (1993) develops a technique for determining the geometric layout structure of a document which begins by merging character strings into text groups. Border lines of blocks are determined by linking edges of text groups. Then blocks which have be oversegmented are merged and a projection profile method is applied to the resulting blocks to differentiate text areas from figure areas. Hirayama reports that on a data set of 61 pages of Japanese technical papers and magazines 93.3% of the text areas and 93.2% of the figure areas were correctly detected.

Ittner and Baird (1993) determine a geometric layout by doing skew and shear angle corrections, partitioning the page into blocks of text, inferring the text line orientation within each block, partitioning each block into text lines, isolating symbols within each text line, and finally merging the symbols into words. The rotation skew angle is determined by taking the projections of the centers of the connected components of the black pixels on the page at a given angle. The angle is iteratively updated to optimize the alignment without having to compute the projection over each possible angle. After rotating the image shear is corrected by a similar technique. They report an accuracy to within 3 minutes of arc indicating that the method fails perhaps one in one thousand images. Blocks are determined by the white space covering technique of Baird (1992). They report that on 100 English document image pages from 13 publishers and 22 styles, 94% of the layouts were correctly determined. The orientation of text lines in a block is determined from the minimum spanning tree of the connected components of the black pixels. The mode of the histogram of the directions of the edges in the minimum spanning tree is the orientation of the text lines in a block. Symbols in a text line are determined by taking the projection in an orthogonal direction to the text line. The projection profile is checked for a dominant frequency and the segmentation into characters is done from the projection profile with the knowledge of the dominant frequency. To determine the words in a text line, they determine a scalable word-space threshold for each text block separately. Then each text line is independently segmented to distinguish the inter-character spacing with the inter-word spacing.

Ankindele and Belaid (1993) determine a geometric page layout that permits blocks to be polygonal as well as rectangular. The determine the elongated white spaces in the document image and then determine the intersections of these white spaces. Points of intersection are candidate vertices for a polygonal block. The polygonal blocks are then extracted from the geometry of the intersection points. No performance results are given.

Tsujimoto and Asada (1990) assume that each block of the geometric page layout contains exactly one logical class. They organize the geometric page layout as a tree. Each new article in a document such as a newspaper begins with a headline which is in the head block. They find the paragraphs which belong to the head block by rules relating to the order of the geometric page layout tree and are able to assign logical structure labels of *title, abstract, subtitle, paragraph, header, footer, page number*, and *caption*. They worked on 106 document images and correctly determined the logical structure for 94 document images.

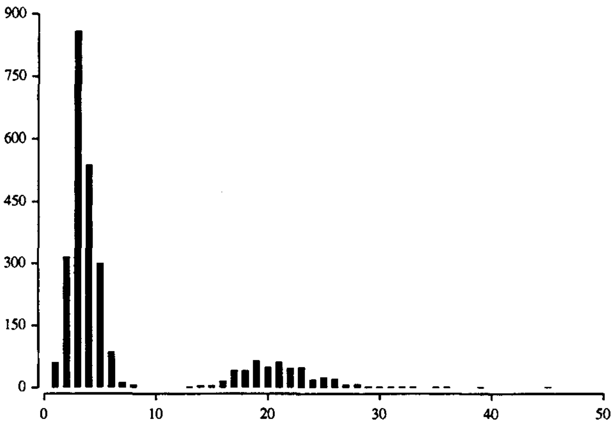


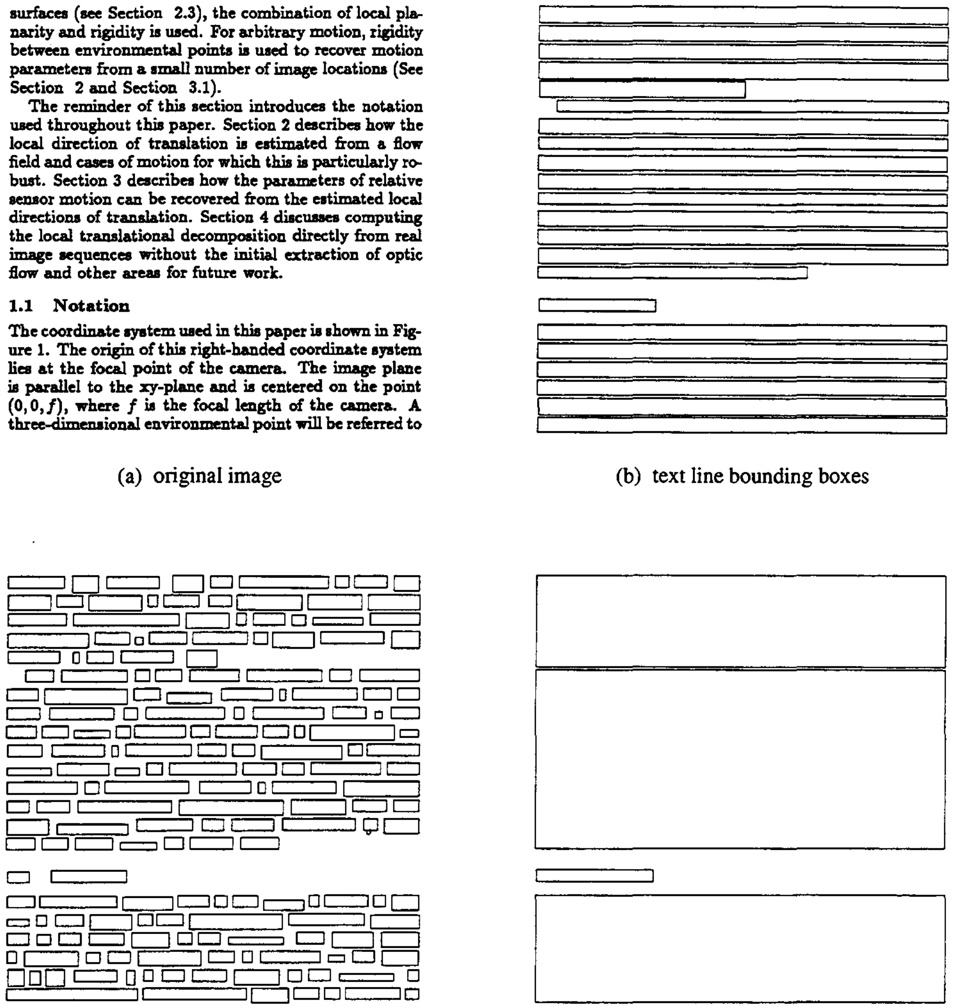
Figure 1: Inter-character Spacing Distribution for a 12-pt font size document: The abscissa ,measured in the pixel unit, represents the horizontal distance between the bounding boxes of two adjacent connected components in a text-line and the ordinate denotes the number of occurrences. The image resolution is 300 dpi and the number of characters used in the document image is 2935.

Visvanathan (1990) employs an X-Y tree to represent the geometric layout and then employed a regular grammar scheme to label the document image blocks. Block labels included: *title, author, abstract, section titles, paragraphs, figure, table, footnote, footers* and *page numbers.* No performance results were given.

Yamashita et al. (1991) use a model-based method. Character strings, lines, and half-tone images are extracted from the document image. Vertical and horizontal field separators (long white areas or black lines) are detected based on the extracted elements, then appropriate labels are assigned to character strings by a relaxation method. Label classes included: *header, title, author, affiliation, abstract, body, page number, column, footnote, block* and *figure.* The technique was applied to 77 front pages of Japanese patent applications. They reported that the logical structure for 59 were determined perfectly.

Dengel (1993) discusses a technique for automatically determining the logical structure of business letters. He reports that on a test set of 100 letters, the recipient and the letter body could be correctly determined.

Saitoh et al. (1993) determine logical layout with text block labels of *body, header, footer*, and *caption.* They tested the technique on 393 document images of mainly Japanese and some English docu-



surfaces (see Section 2.3), the combination of local planarity and rigidity is used. For arbitrary motion, rigidity between environmental points is used to recover motion parameters from a small number of image locations (See Section 2 and Section 3.1).

The reminder of this section introduces the notation used throughout this paper. Section 2 describes how the local direction of translation is estimated from a flow field and cases of motion for which this is particularly robust. Section 3 describes how the parameters of relative sensor motion can be recovered from the estimated local directions of translation. Section 4 discusses computing the local translational decomposition directly from real image sequences without the initial extraction of optic flow and other areas for future work.

1.1 Notation

The coordinate system used in this paper is shown in Figure 1. The origin of this right-handed coordinate system lies at the focal point of the camera. The image plane is parallel to the xy-plane and is centered on the point $(0,0,f)$, where f is the focal length of the camera. A three-dimensional environmental point will be referred to

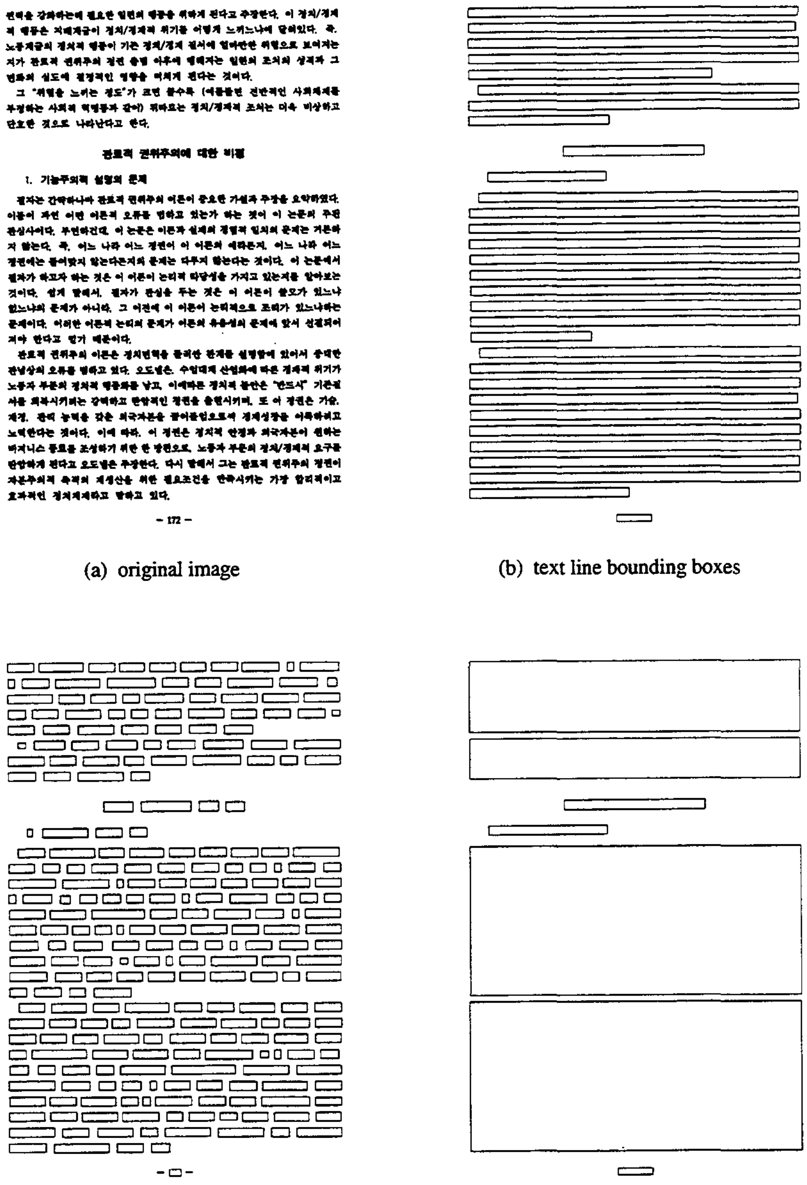
(a) original image

(b) text line bounding boxes

(c) word bounding boxes

(d) text block bounding boxes

Figure 2: Example of Document Image Decomposition: (a) a document image written in English, (b) text line bounding boxes, (c) word bounding boxes, (d) text block bounding boxes.



(a) original image

(b) text line bounding boxes

(c) word bounding boxes

(d) text block bounding boxes

Figure 3: Example of Document Image Decomposition: (a) a document image written in Korean, (b) text line bounding boxes, (c) word bounding boxes, (d) text block bounding boxes.

ments. To characterize performance they measured the average number of times per document image an operator has to correct the results of the automatically produced layout. They report that on the average 2.17 times per image areas not suitable for output have to be discarded, .01 times per image mis-classified areas have to be correctly labeled, and 1.09 times per image does a text area have to be reset. With respect to text ordering they report that it required moving connections .47 times per image, on the average, making new connections .11 times per image, and re-assigning type of text .36 times per image.

3 The Bounding Box Algorithm

The input to the bounding box algorithm is a deskewed binary document image. A connected component algorithm is applied to the black pixels of the deskewed binary image. For each connected component, the smallest rectangular box which circumscribes the connected component is computed. Each of the bounding boxes is considered as a single unit on the page. The algorithm does the horizontal and vertical projections of these bounding boxes. The projection profiles are analyzed and the textlines are extracted. The spatial configuration of bounding boxes within each of the extracted textlines is analyzed to extract words. Textlines are merged into text-blocks based upon the intertextline spacing distribution and also based on the changes in the textline justification. The details are as follows.

Step 1: Bounding Boxes of Connected Components

Let I be the input binary image. A connected-component algorithm (described in [13]) is applied to the black pixels in I to produce a set of connected components. Then, for each of the connected components, the smallest rectangular box which circumscribes the connected component is computed. Such a rectangular box is called the bounding box. A bounding box can be represented by giving the coordinates of the upper left and the lower right corners of the box. Figure 6 shows an example of the bounding box of the character 'g'.

Many kinds of symbols are used in document pages: alphanumeric characters, punctuation marks, mathematical symbols and so on. These symbols can be classified into two groups: "single-component" symbols and "multiple-component" symbols. Character 'g' in Figure 6 is an example of single-component symbols, while character 'i' in Figure 6 is an example of multiple-component symbols. All English alphabets except 'i' and 'j' belong to the former group. Of course, a single-component symbol can be "broken" into several pieces on a degraded image. In this case, the symbol is also considered as a collection of subsymbols, that is, a multiple-component symbol.

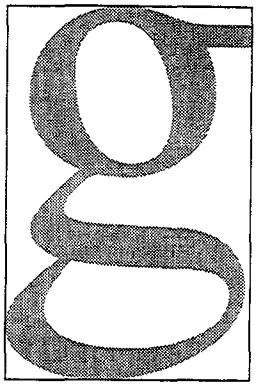


Figure 4: Character 'g' and its bounding box.

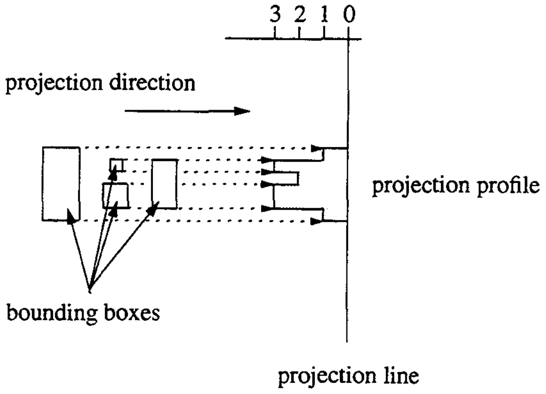
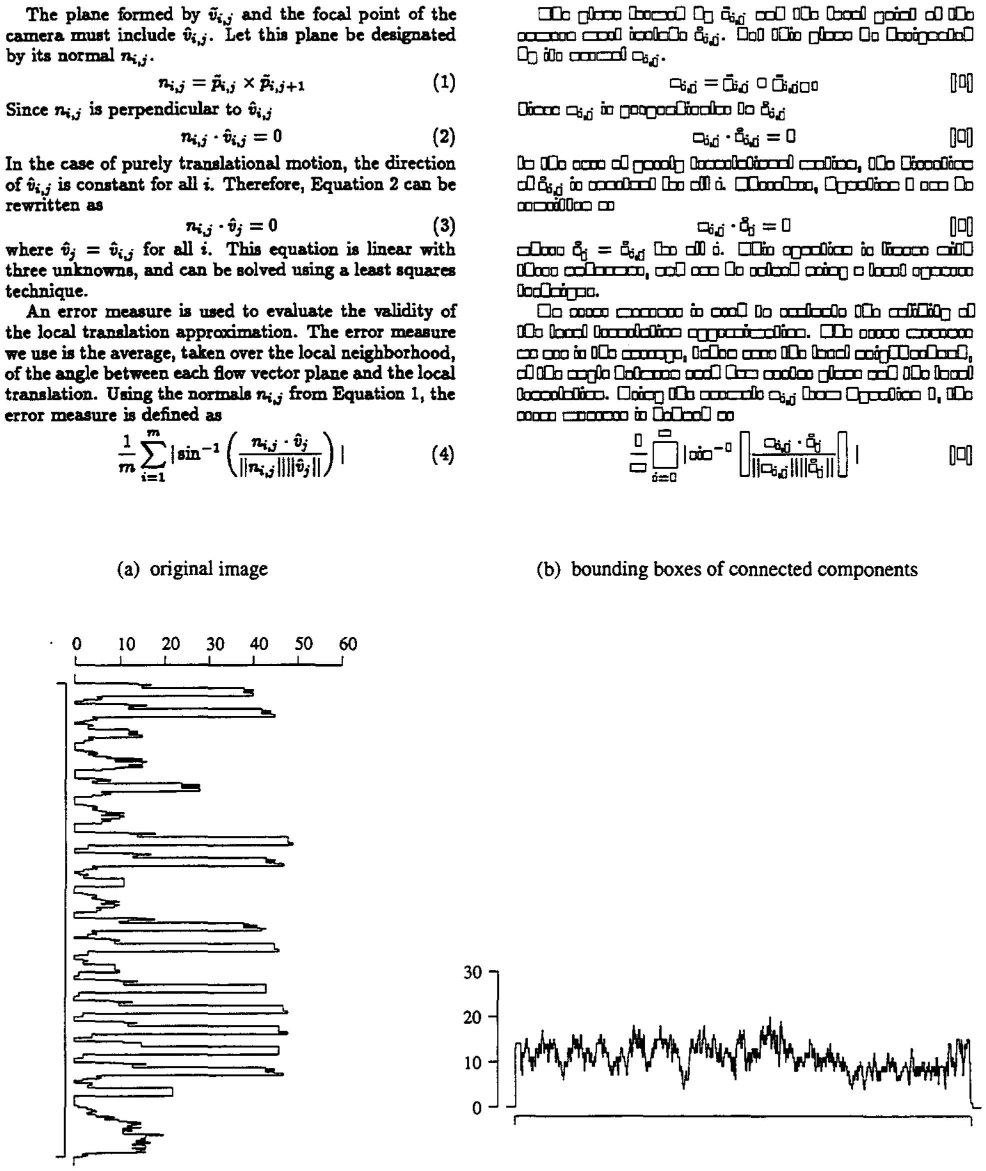


Figure 5: Horizontal Projection Profile, which is obtained by accumulating the bounding boxes onto the vertical line. It is a frequency histogram of the number of projected bounding boxes.

Figure 6a shows a segment of an English document image (taken from the UW English Document Image Database-I, page id "L006SYN.TIF") and Figure 6b shows the bounding boxes produced by the algorithm in this step. Note that, the number of bounding boxes are larger than the number of symbols since multiple bounding boxes are produced for multi-component symbols. Our page decomposition scheme analyzes the spatial configuration of those bounding boxes of connected components to extract textlines, words, and paragraphs.

Step 2: Projections of Bounding Boxes

Analysis of the spatial configuration of bounding boxes can be done by projecting those bounding boxes onto a straight line. Since paper documents are usually written in the horizontal or vertical direction, projections of bounding boxes onto the vertical and horizontal lines are of particular interest. While projecting bounding boxes onto the horizontal or vertical line, they will accumulate onto that line, which results in the projection profile (Figure 5). A projection profile is a frequency distribution of the projected bounding boxes on the pro-



The plane formed by $\tilde{v}_{i,j}$ and the focal point of the camera must include $\hat{v}_{i,j}$. Let this plane be designated by its normal $n_{i,j}$.

$$n_{i,j} = \tilde{p}_{i,j} \times \tilde{p}_{i,j+1} \qquad (1)$$

Since $n_{i,j}$ is perpendicular to $\hat{v}_{i,j}$

$$n_{i,j} \cdot \hat{v}_{i,j} = 0 \qquad (2)$$

In the case of purely translational motion, the direction of $\hat{v}_{i,j}$ is constant for all i. Therefore, Equation 2 can be rewritten as

$$n_{i,j} \cdot \hat{v}_j = 0 \qquad (3)$$

where $\hat{v}_j = \hat{v}_{i,j}$ for all i. This equation is linear with three unknowns, and can be solved using a least squares technique.

An error measure is used to evaluate the validity of the local translation approximation. The error measure we use is the average, taken over the local neighborhood, of the angle between each flow vector plane and the local translation. Using the normals $n_{i,j}$ from Equation 1, the error measure is defined as

$$\frac{1}{m}\sum_{i=1}^{m} |\sin^{-1}\left(\frac{n_{i,j} \cdot \hat{v}_j}{\|n_{i,j}\|\|\hat{v}_j\|}\right)| \qquad (4)$$

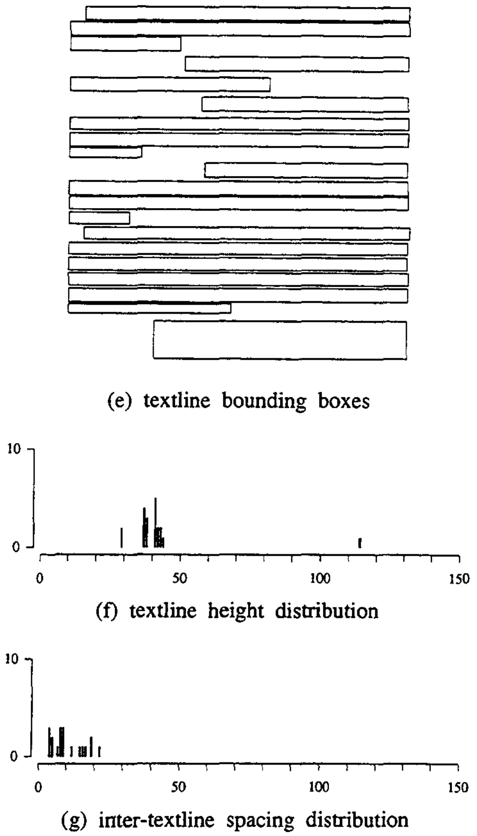
(a) original image

(b) bounding boxes of connected components

(c) horizontal projection profile

(d) vertical projection profile

Figure 6: Bounding Boxes of Connected Components and the Projection Profiles: (a) a document image written in English, (b) bounding boxes of connected components, (c) horizontal projection profile, (d) vertical projection profile.



(e) textline bounding boxes

(f) textline height distribution

(g) inter-textline spacing distribution

Figure 6: (continued) (e) text-line bounding boxes, (f) text-line height distribution, (g) inter-text-line spacing distribution. The abscissa represents the heights of text lines/inter-text-line spacings in pixel unit and the ordinate represents the number of occurrences.

jection line. The bounding box projection profiles provide important information about the number of bounding boxes aligned along the projection direction. Figure 6(c) and 6(d) shows the horizontal and vertical projection profiles of the bounding boxes in Figure 6(b).

Step 3: Extraction of Text-Lines

In this step, our algorithm first determines the textline direction of the page by analyzing both horizontal and vertical projection profiles. Once the text-line direction of the page is determined, the algorithm partitions the page bounding box into textline bounding boxes.

If we take a careful look at the projection profiles in Figure 6(c) and 6(d), it is clear that the text-lines are horizontally oriented: Within the horizontal projection profile, there are distinct high peaks and deep valleys at somewhat regular intervals, whereas within the vertical projection profile, there is no such distinction. Since the bounding boxes are represented by a list of coordinates of the two-points, the text-lines are easily extracted. The result is shown in Figure 6(e).

Other important informations can be deduced from the horizontal projection profile: the frequency distribution of textline heights and inter-textline spacings (Figure 6(f) and 6(g)). These distributions will be employed in the text-block extraction process.

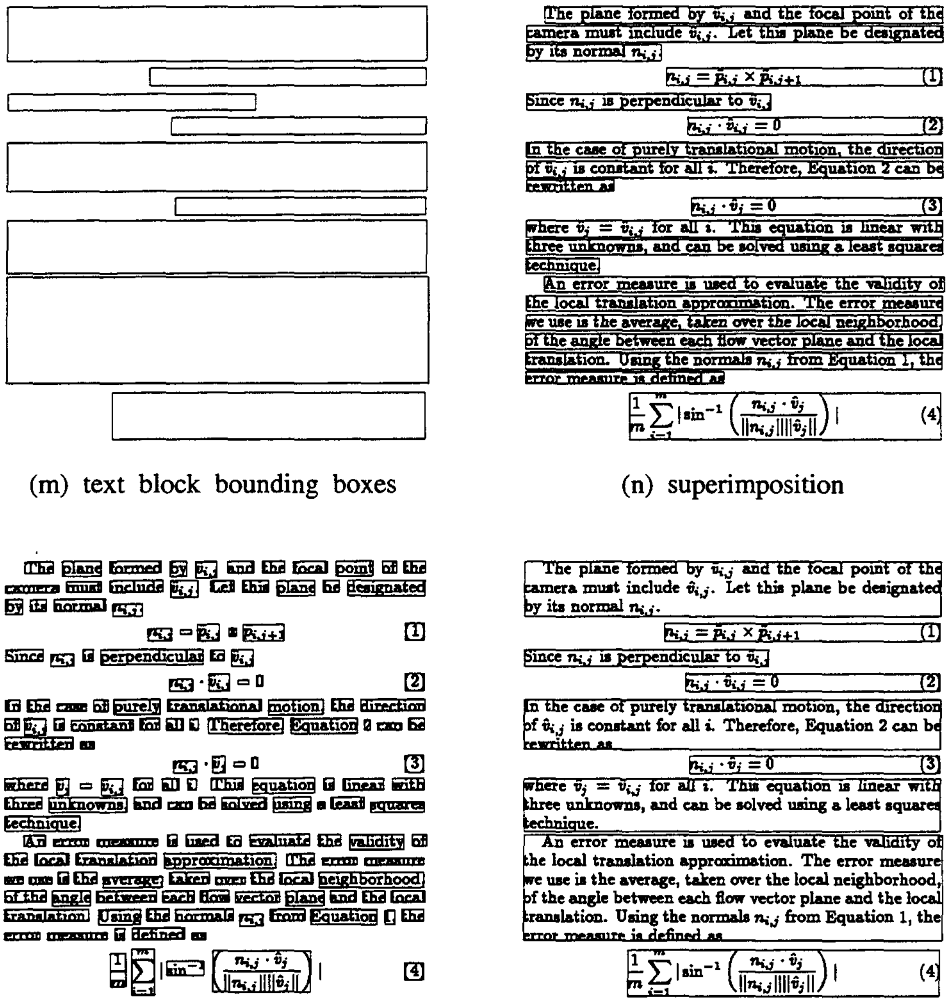
Step 4: Extraction of Words

In this step, the algorithm groups the bounding boxes on each text-line (produced from the last step) into bounding boxes of words.

Our algorithm first computes the projection profiles within each of the textline bounding boxes. Again, the units of the projection profile are bounding boxes, not pixels. Next, the algorithm considers each of the projection profiles as a one-dimensional *gray-scale image*, and threhsolds each of the images to produce a binary image. The threshold value is set to 1. Figure 6(h) shows projection profiles within textlines and Figure 6(i) shows the corresponding binarized 1-D profiles. Note that, during the binarization, a symbol (or a broken symbol) with multiple bounding boxes may be merged into one, as well as, those adjacent symbols within the same textline whose bounding boxes are overlapping with each other. But this will not cause any problem in the result of our word extraction process, since our algorithm extracts words by merging symbols' bounding boxes to form words.

After binarization, the algorithm performs a morphological closing operation on each of the binarized text-line projection profile with structuring element of appropriate size (The fastest algorithm for one-dimensional morphological closing operation is introduced in the Appendix II). The length of the structuring element is determined by analyzing the distribution of the run-length of the zero's on the binarized text-line projection profile. In general, such a run-length distribution is bi-modal. One mode represents the inter-symbol spacing, and the other represents the inter-word spacing. Figure 6(j) shows the frequency distribution of inter-character spacings, The length of the structuring element is chosen between the two modes. As the result, the morphological closing operation closes spaces between symbols, but not the spaces between words. (See Figure 6(i) and 6(k)). Figure 6(l) shows the results of this step.

In a printing process, each symbol is associated with its width, the amount of space it occupies when it appears in a text-line. In some fonts this spacing is constant, i.e., it does not vary from character to character. These fonts are called fixed pitch or monospaced fonts; they are used mainly for typewrite-style printing. Most fonts used for high quality typography, however, associate a different width with each character. Such fonts are called variable pitch fonts. In either case, most of alphabetic languages are usually printed in such a way that less spacing is put between consecutive char-



The plane formed by $\hat{v}_{i,j}$ and the focal point of the camera must include $\hat{v}_{i,j}$. Let this plane be designated by its normal $n_{i,j}$.

$$n_{i,j} = \bar{p}_{i,j} \times \bar{p}_{i,j+1} \qquad (1)$$

Since $n_{i,j}$ is perpendicular to $\hat{v}_{i,j}$

$$n_{i,j} \cdot \hat{v}_{i,j} = 0 \qquad (2)$$

In the case of purely translational motion, the direction of $\hat{v}_{i,j}$ is constant for all i. Therefore, Equation 2 can be rewritten as

$$n_{i,j} \cdot \hat{v}_j = 0 \qquad (3)$$

where $\hat{v}_j = \hat{v}_{i,j}$ for all i. This equation is linear with three unknowns, and can be solved using a least squares technique.

An error measure is used to evaluate the validity of the local translation approximation. The error measure we use is the average, taken over the local neighborhood, of the angle between each flow vector plane and the local translation. Using the normals $n_{i,j}$ from Equation 1, the error measure is defined as

$$\frac{1}{m}\sum_{i=1}^{m} \left| \sin^{-1}\left(\frac{n_{i,j} \cdot \hat{v}_j}{\|n_{i,j}\|\|\hat{v}_j\|}\right) \right| \qquad (4)$$

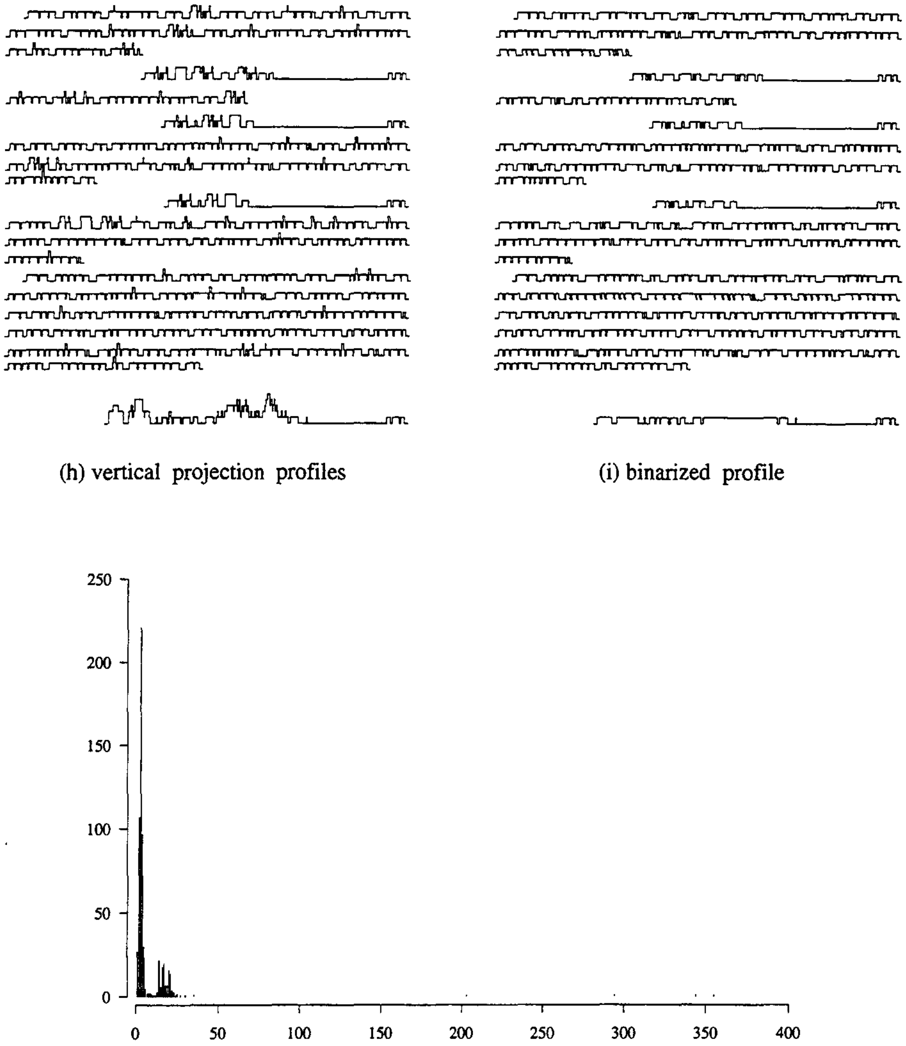
(m) text block bounding boxes

(n) superimposition

(o) superimposition

(p) superimposition

Figure 6: (continued) (m) text block bounding boxes, (n) superimposition of text line bounding boxes and the original image, (o) superimposition of text word bounding boxes and the original image, (p) superimposition of text block bounding boxes and the original image.

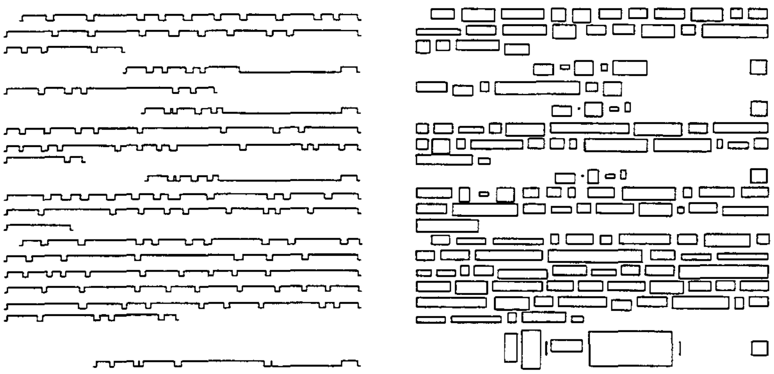


(h) vertical projection profiles

(i) binarized profile

(j) inter-character spacing distribution

Figure 6: (continued) (h) vertical projection profiles, (i) binarized vertical projection profiles, (j) inter-character spacing distribution (see Figure 3).



(k) closed binary vertical profiles

(l) word bounding boxes

Figure 6: (continued) (k) vertical projection profiles, (l) binarized vertical projection profiles.

acters in a word while more spacing is put between neighboring words on a text line. This fact can easily be observed from the vertical projection profile for each text line (Figure 6(h)).

Figure 6(j) shows the frequency distribution of inter-character spacings, which are actually used in the example text shown in Figure 6(a). As is expected from the fact that more spacing is usually placed between words rather than between characters, the inter-character spacing distribution is usually bimodal: The mode on the left side corresponds to the inter-character spacings and the mode on the right side to the inter-word spacings. In fixed pitch typefaces such as the Courier typeface the separation of two modes is outstanding while it is not so in variable pitch typefaces such as the "Times Roman" typeface. Anyway, some value between the two modes will function as a threshold value for word extraction.

Characters may be kerned to improve their visual spacing. Kerning is a modification to inter-character spacing of text. Kerning is applied with proportionally spaced typefaces for purely visual reasons: either because it looks better, or because it improves the legibility of the text [14]. When kerning occurs, the bounding boxes of the characters to which kerning is applied will overlap, which is reflected on the vertical projection profile (Figure 6(h)).

Sometimes, characters are *decorated* or *accented* as can be seen, for example, in é. Such an accent is usually found in a variety of foreign languages as well as in mathematical expressions. Accented symbols are composed of multiple components whose bounding boxes are aligned in the vertical direction. Also, accentuation is reflected on the vertical projection profile at a text line.

Step 5: Extraction of Paragraphs

In this step, the algorithm groups the bounding boxes of the extracted text-lines into bounding boxes of text-blocks.

There are four basic types of text line layout that are in common use: centered, flush left, flush right, and justified. These are also frequently referred to as centered, left-justified, right-justified, and fully-justified.

Though a paragraph is a unit in the logical hierarchy, its physical appearance is noticeable in a document image. Whatever justification has been used in the preparation of a document, paragraphs are usually made either by changing the justification of the current text line or by putting more space between two text lines, one of which is from the previous paragraph and the other of which from the current one. In the former case, one might usually indent the first line of a paragraph of text.

Extraction of paragraphs should be primarily based on the above two basic paragraph-breaking methods. When a significant change in text-line heights or in inter-text-line spacings occurs, we might say that a new paragraph begins. The distributions of text-line heights and inter-text-line spacings (Figure 6(f) and 6(g)) together with the horizontal projection profile (Figure 6(c)) give us a clue for this kind of paragraph breaking. If there is a change in the text-line justification, we might say that a new type of text block begins.

Let us take a look at Figure 6(a). The fourth line is a displayed math zone, which is right-justified in this example, and in the fourteenth text line, some amount of indentation was placed, and therefore, a new paragraph begins, and so on. Figure 6(m) shows the text block bounding boxes for our example text, which can be obtained merely by taking into consideration the above two basic text block breaking methods. In Figure 6(n)–6(p) are shown superimpositions of the bounding boxes obtained as above and the original image.

References

[1] O.T. Akindele and A. Belaid, "Page Segmentation by Segment Tracing," *2nd ICDAR*, Tsukuba, 1993, pp. 341–344.

[2] N. Amamoto, S. Torigoe, Y. Hirogaki, "Block Segmentation and Text Area Extraction of Vertically/Horizontally Written Document," *2nd ICDAR*, Tsukuba, 1993, pp. 739–742.

[3] H.S. Baird, "Background Structure in Document Images," *Advances In Structural and Syntactic Pattern Recognition*, World Scientific, Singapore, 1992, pp. 253–269.

[4] D.S. Bloomberg, "Multiresolution Morphological Approach to Document Image Analysis," *1st ICDAR*, Saint-Malo, 1991, pp. 963–971.

[5] A. Dengel, "Initial Leraning of Document Structure," *2nd ICDAR*, Tsukuba, 1993, pp. 86–90.

[6] X. Hao, J.T.L. Wang, P.A. Ng, "Nested Segmentation: An Approach for Layout Analysis in Document Classification," *2nd ICDAR*, Tsukuba, 1993, pp. 319–322. Atlantic City, 1990, pp. 464–468.

[7] Y. Hirayama, "A Block Segmentation Method For Document Images with Complicated Column Structures," *2nd ICDAR*, Tsukuba, 1993, pp. 91–94.

[8] F. Lebourgeois, Z. Bublinski, and H. Emptoz, "A Fast and Efficient Method For Extracting Text Paragraphs and Graphics from Unconstrained Documents," *11th ICPR*, The Hague, 1992, pp. 272–276.

[9] G. Nagy and S.C. Seth, "Hierarchical Representation of Opically Scanned Documents," *7th ICPR*, Montreal, 1984, pp. 347–349.

[10] G. Nagy, S.C. Seth, S.D. Stoddard, "Document Analysis With An Expert System," *Pattern Recognition in Practice II*, E.S. Gelsema and L.N. Kanal editors, North Holland, Amsterdam, 1986, pp. 149–159.

[11] L. O'Gorman, "The Document Spectrum for Bottom-Up Page Layout Analysis," *Advances In Structural and Syntactic Pattern Recognition*, World Scientific, Singapore, 1992, pp. 270–279.

[12] T. Pavlidis and J. Zhou, "Page Segmentation by White Streams," *1st ICDAR*, Saint-Malo, 1991, pp. 945–953.

[13] T. Saitoh and T. Pavlidis, "Page Segmentation without Rectangle Assumption," *11th ICPR*, The Hague, 1992, pp. 277–280.

[14] T. Saitoh, M. Tachikawa, T. Yamaai, "Document Image Segmentation and Text Area Ordering," *2nd ICDAR*, Tsukuba, 1993, pp. 323–329.

[15] S. Tsujimoto and H. Asada, "Understanding Multi-articled Documents," *10th ICPR*, Atlantic City, 1990, pp. 551–556.

[16] F.M. Wahl, K.Y. Wong, and R.G. Casey, "Block Segmentation and Text Extraction in Mixed Text/Image Documents," *Computer Graphics and Image Processing* Vol 20, 1982, pp. 375–390.

[17] A. Yamashita, T. Amasno, H. Takahashi, and K. Toyokawa, "A Model Based Layout Understanding Method for the Document Recognition System," *1st ICDAR*, Saint-Malo, 1991, pp. 130–138.

Page Decomposition and Related Research at the University of Maryland

Document Processing Group
University of Maryland, College Park, MD 20742

Abstract

Document image understanding at the University of Maryland has covered a number of areas in recent years. Most recently, work has been directed toward the application of page segmentation and classification to very large heterogeneous databases of document images.

In this report, we present results from work on several currently disjoint projects relating to page decomposition.

1 Introduction

This report overviews sponsored research in the document processing group performed over the past year and a half. In Section 2 we present an approach to layout independent page segmentation and in Section 3 a representation for generic document structure. In Section 4 we highlight several other projects of interest to the general community with extended abstracts.

2 Multiscale Layout Independent Page Segmentation

In this project we address the task of layout independent physical page segmentation. In an attempt to handle even the most difficult cases of segmentation, we make few assumptions about the document's textual and graphical attributes and layout structure. The system is designed so that as hypotheses about document components are generated and verified, more domain specific processing may occur.

Page segmentation and layout analysis methods described in the literature make use of well known image processing tools which can be broadly classified, depending on the nature of the precise application [1–6]. We see our approach as complementary, since we make additional layout independent assumptions.

In some applications it is desirable to have segmentation methods that do not assume a priori knowledge about the content and attributes of text, or about the boundaries of major blocks. Such approaches should be robust to skew, noise and other degradations. Some of the difficulties which are common in the general class of documents, and which make these goals hard to obtain include:

- Noise and degradations caused by copying, scanning, transmission or aging.
- Page skew and text regions with different orientations on the same page.
- Text regions touching or overlapping with image and graphics components.
- Combinations of varying text and background gray levels (e.g. inverted texts).
- Complex and irregular layout structures that are common especially in non-technical documents. Document objects may not have rectangular or even convex boundaries and one may be embedded in others.
- Curved lines or multi-column pages where text lines in the two columns pare not of the same size and/or are not aligned.
- Differences in language, font sizes and other textual attributes.

In our approach, text, image and graphics regions in a document image are described as three classes of textures. The approach can be justified by the fact that humans can identify document objects easily even from low resolution images or from distant views of document page. This shows that the physical segmentation of document is not detail or content sensitive and like texture segmentation is a low level vision process. Given the following considerations, some of the existing texture segmentation techniques [12–14] can be modified and used to identify these regions on the page.

Our method is based on the fact that there is some uncertainty associated with the local decisions over small windows, due to the limited view of signals and/or randomness and ambiguity inherent in the problem. The alternative is using large windows which is not recommended because over larger windows signals are highly non-stationary and the corresponding features are resulted from averaging over heterogeneous micro-structures and therefore are less reliable. Also larger windows provide less spatial resolution which is of great concern in segmentation schemes.

In the document domain, image sub-blocks may contain text, picture and graphics sub-regions adjacent to, or overlapping, one another. Such situations may occur on the boundaries, where for example, text lines come close to or touch image regions, or when there are major text regions in an image or on graphs. For such cases it is not appropriate, even for an optimally designed classifier, to make hard (binary) local decisions. This shows an inherent fuzziness of class membership which does not come from the noise or randomness and it supports our claim that soft local decisions are more realistic and efficient. The uncertainties reflected in soft decisions are then reduced by propagating and integrating decisions made independently in a neighborhood within and across scales. A flowchart of the system is shown in Figure 1.

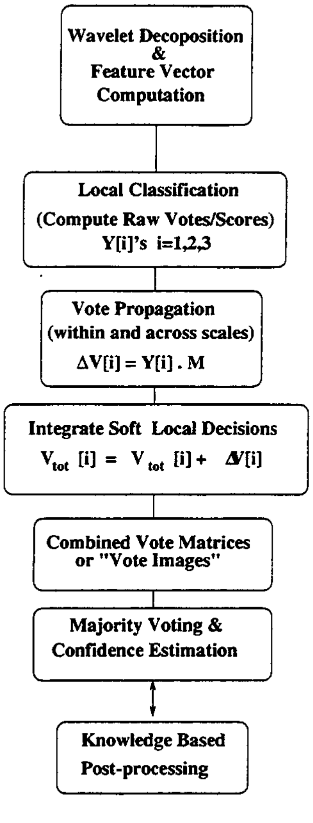

Wavelet Decoposition & Feature Vector Computation
Local Classification (Compute Raw Votes/Scores) Y[i]'s i=1,2,3
Vote Propagation (within and across scales) ΔV[i] = Y[i] . M
Integrate Soft Local Decisions V_tot [i] = V_tot [i] + ΔV[i]
Combined Vote Matrices or "Vote Images"
Majority Voting & Confidence Estimation
Knowledge Based Post-processing

Figure 1: Vote propagation and integration steps.

2.1 Wavelet Packet Basis

Wavelet packet analysis algorithms (wavelet transforms being special cases) allow us to perform adaptive Fourier windowing directly in the time domain by successive filtering of the signal into different frequency regions. The waveforms in the library are mutually orthogonal and each of them is orthogonal to all of its integer translates and dyadic rescaled versions. The full collection of wavelet packets (including translates and rescaled versions) provides us with a library of "templates" or "notes" which are matched "efficiently" to signals for analysis and synthesis.

We start with an exact Quadrature Mirror Filter (QMF) $h(n)$ satisfying

$$\sum h(n-2k)h(n-2\ell) = \delta_{k,\ell}\ ,\ \sum h(n) = \sqrt{2}. \quad (1)$$

We let $g_k = h_{k+1}(-1)^k$ and define the operations F_i on $\ell^2(\mathbf{Z})$ into "$\ell^2(2\mathbf{Z})$" by

$$\begin{aligned} F_0\{s_k\}(i) &= 2\sum s_k h_{k-2i} \\ F_1\{s_k\}(i) &= 2\sum s_k g_{k-2i} \end{aligned} \quad (2)$$

The map $F(s_k) = F_0(s_k) \oplus F_1(s_k) \in \ell^2(2\mathbf{Z}) \oplus \ell^2(2\mathbf{Z})$ results in an orthogonal "decomposition", and satisfies the alias cancellation and perfect reconstruction conditions [7]. We now define the following sequence of functions.

$$\begin{cases} W_{2n}(x) = \sqrt{2}\sum h_k W_n(2x-k) \\ W_{2n+1}(x) = \sqrt{2}\sum g_k W_n(2x-k) \end{cases} \quad (3)$$

A *Wavelet Packet Basis* [7] for $L^2(\mathbf{R})$ is any orthonormal basis selected from the functions $2^{k/2}W_n(2^k t - j)$. The waveforms $\{W_\gamma\}$ are induced by three parameters $\gamma = \{k, n, j\}$ with physical interpretations of scale, frequency (or sequency), and position. These waveforms constitute our tree-structured basis. Each node in the tree represents a subspace of its parent's space and each subspace is the orthogonal direct sum of its two children. In wavelet analysis of 2-D signals, for simplicity, $L^2(R^2)$ is typically considered as a separable Hilbert space. In filter bank implementation of wavelet packets the assumption about separability of the signal space results in separable filters along the row and column directions [7]. Depending on the application, the choice of a suitable wavelet packet tree can be based on different criteria (Figure 2). Energy and entropy based [7] as well as class separability [8, 9] based algorithms for basis selection have been suggested. In the following experiments, without any claim of optimality, a pyramidal wavelet transform and an energy based wavelet packet tree are used. In the following classification experiments, the wavelet-based features that are used are the second and third central moments (μ_2 and μ_3) of the image subbands computed over small windows W on each subimage of the transformed image.

2.2 Propagation and Integration of Local Soft Decisions

Many signal/image processing tasks consist of local processing of data followed by the combination of results obtained from the local windows. Image segmentation and boundary detection are examples of such tasks. On the other hand the recent success of neural networks and fuzzy systems has reconfirmed the fact that soft distributed decisions provide powerful and robust tools for dealing with uncertainty, ambiguity, and randomness [10, 11].

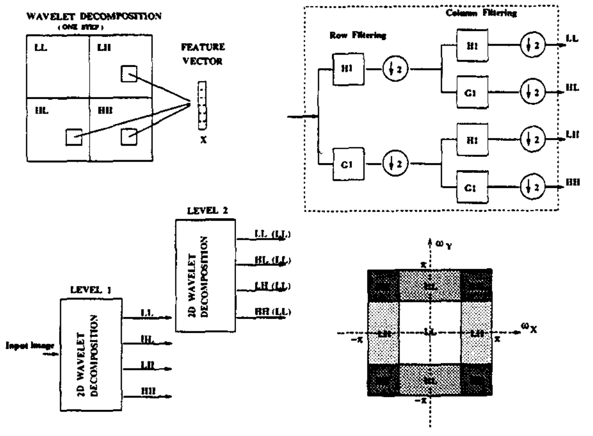


Figure 2: Two-dimensional Wavelet Transform 2D-WT (a) QMF filter bank structure for one level of 2D-WT, (b) corresponding subband decomposition, (c)two levels of 2D-WT, (d) feature vectors are computed at each scale.

Local decisions, which are inevitable in many signal processing schemes, suffer from such ambiguities and are not reliable on their own. There is therefore a need to devise methods of resolving the ambiguity and fuzziness of local decisions, in a consistent way, by incorporating more and more "evidence" or "contextual" information from the input signal or image. Since a reliable hard decision cannot be achieved by observation over a single small window, we suggest that local soft decisions/scores be propagated and integrated based on a weighting pattern in a neighborhood within each scale as well as across scales, and that the majority of the combined votes be used as a class identifier.

Assume that L classes of regions may be present in the image. For example, in the document domain we can set $L = 3$, corresponding to text, graphics and image regions. Our classifier is a map $F(.)$, typically non-linear and many-to-one, from the feature space $\{Y_s, s = (i,j)\}$ to the points in the "fuzzy" cube $[0,1]^L$. Thus

$$\begin{aligned} F : \Re^n &\rightarrow [0,1]^L \\ F(X_s) &= Y_s \end{aligned} \qquad (4)$$

where Y_s is the vote vector whose i^{th} element is the score or fit value associated with class i. This framework, along with proper training, provides the soft local decisions.

We also assume that the information contained in a block B about a region A is proportional to the intersection area of B and A. As the image is analyzed using windows of size W and window shift steps of size w, the area contained in each block B is also partially covered by neighboring blocks. Since analysis is performed at several scales, the area in B is also covered by windows at each scale [11]. Therefore, the pattern of decision flow in space as well as in scale has to be determined.

Based on our assumptions, we define a "Vote Propagation Matrix" (VPM) M which shows how much the vote of the window centered at $s = (x,y)$ should affect or contribute to its neighbors:

$$M_{i,j} = 1 - (|i| + |j|)/\delta + |ij|/\delta^2 \text{ for } -\delta < i,j < \delta \qquad (5)$$

$$(V)_{x+i,y+j} = (V)_{x+i,y+j} + Y_{x,y} \cdot M_{i,j} \qquad (6)$$

where $\delta = w/W$. The incremental vote $\Delta V_{x+i,y+j} = Y_{x,y} \cdot M_{i,j}$ is added to the vote of the corresponding principal block (centered at location (i,j) relative to the current block) to give its contribution to the sum of the votes in that block. After one complete scan of the image, the contributions of all neighboring blocks in support of matrix M have thus been accumulated, and matrix V_{tot} contains the combined vote for each block of size w. Note that the vote $(V)_{i,j}$ is a vector of L votes each corresponding to one class. All "decisions" thus far have been expressed as real numbers and no hard decision has been made.

To perform multi-resolution analysis we process the image at different scales and combine the resulting classifications. This is typically done from coarse to fine: we start at low resolution and use higher resolution where the confidence level is not satisfactory. The combination of decisions can make use of the fact that the windows of the low resolution image are projections of larger areas of the high resolution image. One can thus similarly define an "Across-Scales Vote Propagation Matrix"(ASVPM) based on overlapping areas and add the relevant weighted votes. The final majority votes and their confidence measures are based on the accumulation of all soft decisions within and across scales and the closeness of the best class candidates respectively:

$$\begin{aligned} (V_{\text{final}})_{i,j} &= V^{(l*)} = \max_l (V^{(l)})_{i,j} \qquad (7) \\ \text{Conf}_{i,j} &= V^{(l*)} - \max_{l \neq l*} (V^{(l)})_{i,j} \qquad (8) \end{aligned}$$

2.3 Knowledge-Based Biased Voting

In some applications we may wish to incorporate constraints into our decisions based on a priori or derived knowledge about the domain. Although such constraints are often considered at higher levels of

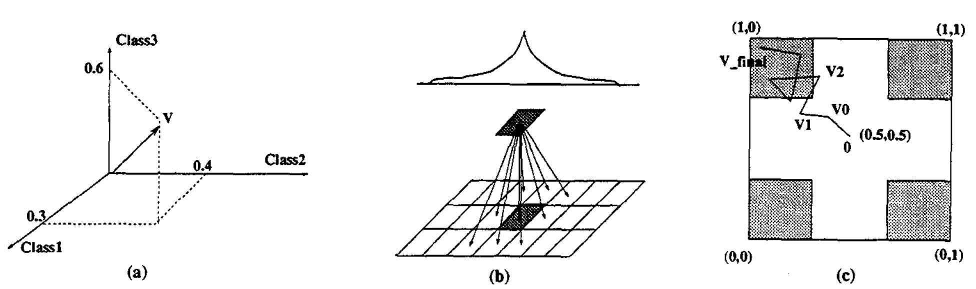


Figure 3: Decision integration: (a) Soft decision vectors, (b) Decision propagation profile in a neighborhood, (c) Weighted combination of soft decisions in the fuzzy decision square. Decisions outside the gray areas have low levels of confidence.

processing, one may also utilize them at the early stages of classification to get more reliable results. In the context of the described majority vote method this idea can easily be incorporated into the system, without increasing its complexity, by using a biased voting scheme. An expectation of observing a certain class of patterns in part of the scene is reflected in a biased vote in favor of a particular class within that region. In this case the system does not start from an all-zero vote matrix V_{tot}, rather, at each position, small non-zero initial votes are given to classes that have been frequently observed in that location. Thus we start from a non-zero vote matrix, on top of which the votes are accumulated. The initial vote may change during decision integration if the combined decision strongly favors another class. The method of "biased voting" based on prior expectations, can be viewed as not starting from the middle of the fuzzy decision cube (i.e. the most ambiguous point), but deviating from it toward one of the corners; see Figure 3c. Note that if a hard decision is made in each block, instead of a soft decision, the suggested combination of votes at each scale would be equivalent to a special form of median filtering on the vote matrix.

2.4 Post-Processing

The spatial patterns of combined soft decisions in the vote matrices directly reflect the locations, shapes and classes of major document blocks. In order to resolve ambiguity and obtain a more precise segmentation, additional post-processing based on knowledge about the structure of the document may be required. The types of processing which can be applied fall into two classes—class dependent and class independent.

If the class of document (e.g. correspondence, journal article, advertisement,...), can be identified, class dependent processing may proceed in a top-down manner using a hypothesize and test approach. We are able to use knowledge about the expected layout to refine the segmentation, as well as verify the document class hypothesis. As a general case, suppose a two-column structure is inferred from a simple projection profile. Rectangular blocks are then be fit to the text blocks and precise column boundaries are identified. If an advertisement is hypothesized, polygonal or non-vertical rectangles may be used as an initial estimate. In our current system we apply a collection of rules which are derived from class independent observations about documents. By applying structural rules to the different regions, one can hypothesize and eliminate "logically undesirable" gaps and small noise-like mis-classified regions. We establish a set of rules which suggest that a text region is typically uniform, with no small inset graphic or image components. Graphic or image regions are "large" relative to the line spacing and/or font size. Graphic and image regions can have subordinate text, but such strings are assumed to be on the order of several symbols long. If text regions are touching graphic components, they are assumed to be part of the graphic component, and will be separated later. Similarly, graphic and image regions will not overlap and graphic regions having uniform (possibly white-space) backgrounds. These properties are enforced by applying morphological closing operations each region to filter out noise which is too small to constitute a valid document region. For text regions, a 10–12 step operation is sufficient to eliminate regions smaller then the largest symbols in a 9pt font, scanned at 300 dpi. For graphic and image regions, a larger kernel is used, eliminating slightly larger regions. The resulting regions are examined, and regions corresponding to "closed" holes are eliminated. Since each region is processed separately, there is no risk of merging multiple larger regions. When the segmentation component is incorporated into a more complete system which may include logical analysis, higher level processing, based on the local hypothesis, will also be used to identify rectangular structures, lines of text, and subordinate text regions when appropriate.

2.5 Experiments

To show the effectiveness of the suggested soft decision integration method it has been applied to document page segmentation.

2.6 Input Representation and Training Set

In the following experiments both wavelet transform and wavelet packet decomposition are used as input signal representation. In the first two examples, features are computed from a two-level wavelet transform. At each level, only detail subbands are used and there is one classifier for each scale. The result of classification at the two scales are combined. In the other examples, features are selected using separability measure on wavelet packet decomposition. For these experiments, six features that contain the highest classification information, based on the previously used separability measure, have been selected and used.

The input data consist of several gray scale document pages scanned at 200 dpi and the input features are the second and third central moments (μ_2 and μ_3) of the image subbands computed over small windows W on the decomposed image:

$$
\begin{aligned}
\mu_n(W) &= \frac{1}{|W|}(\sum_{z \in W}(f(z) - \bar{f_W})^n)^{1/n} && (9)\\
X &= \{x_i = \mu_2(W_i), && (10)\\
& \quad x'_i = \mu_2(W_i)/\mu_3(W_i)\\
& \quad i = 0, 1, \ldots, N_{\text{subbands}}\}
\end{aligned}
$$

where W_i is the local window on the i^{th} subband. In each subband $f(z)$ and $\bar{f_W}$ are defined as the intensity value at location z and the average intensity on window W centered at z, respectively,

The training set consists of about 200 samples from each of the text, image and graphics sub-blocks. These 16 × 16 pixel sub-blocks are extracted randomly from several document pages. In order to avoid over-training, a "validation" set is used to test the performance of the network, after every 10 iterations, during the training stage. As training proceeds, errors on both the training and validation sets decrease. Training is suspended as soon as the error in the validation set starts increasing. If the desired performance is achieved, the process stops; otherwise, part of the validation set is included in the training set and training proceeds on the augmented training set.

2.7 Network Description and Training

In all of the experiments, multi-layer feed-forward neural networks [15] are used as the soft classifiers [16, 17]. The network consist of six input, eight hidden and three output units. The input units are linear, whereas the hidden and output units have sigmoid nonlinearities. A conjugate gradient method is used for fast convergence of the supervised learning algorithm [18].

The three outputs correspond to text, image, and non-text non-image classes. In other words, any subblock not identified as text or image is considered as "graphics". Blank regions are detected separately in a straightforward way. The outputs can take values in $[0, 1]$ and the network is trained in such a way that these outputs provide soft non-binary decisions about the class memberships of the input image blocks. This is essential because, as mentioned above, small regions may locally resemble more than one class, or the image sub-block may be composed of text, image or graphics subregions. In such cases, during the training, outputs corresponding to text, graphics and images are required to take target values roughly in proportion to the fraction of block area they occupy. Including such composite blocks in the training set results in better performance on the boundaries. If a decision integration stage is used the result will be much less sensitive to these adjustments.

Despite its significance, the effect of a suitable output representation is sometimes overlooked. In fact in some cases, such as design and training of soft decision based classifiers, choice of output representation can be as important as that of input representation. In this experiment, in order to provide the learning algorithm with a consistent set of input-output pairs the following procedure has been implemented: Assuming that the data in the training set is labeled correctly and consistently, for any macropixel w and any class l one can compute the desired soft decision for class-membership as:

$$\forall w \in V \;\; L_l^{(\text{Target})}(w) = \frac{1}{|w|} \sum_{x \in w} I(\text{Lab}(x) = l) \quad (11)$$

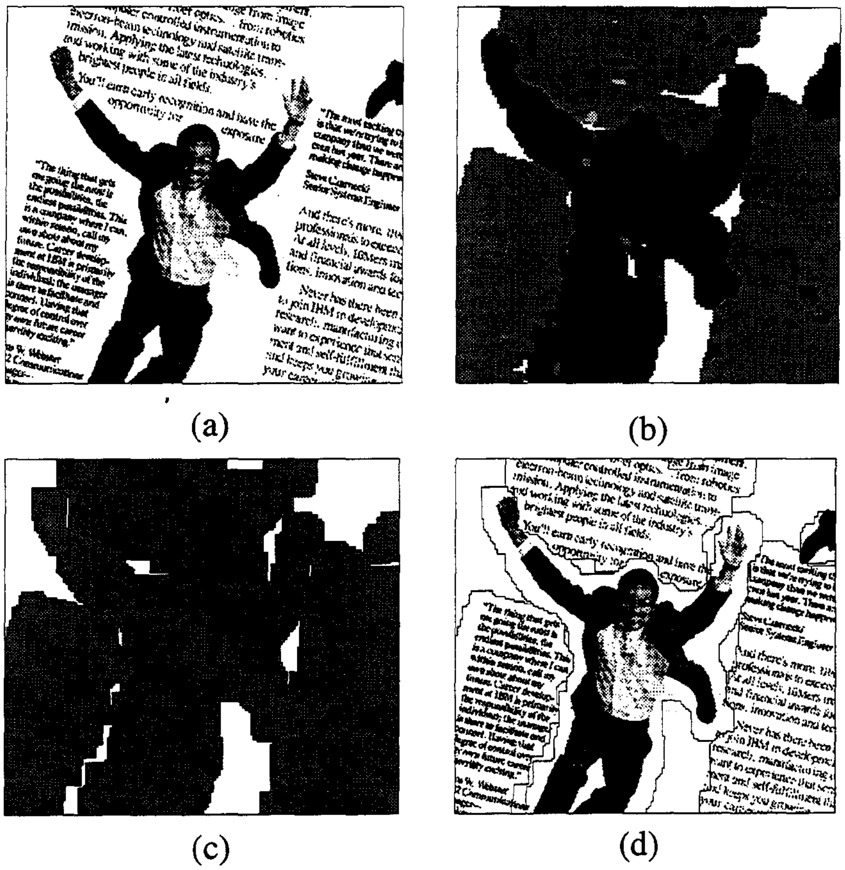
i.e., the relative number of pixels in the window labeled as l. This form of target value computation is consistent with our assumption in Equation 3.11 and is a suitable means of determining soft local decisions when mixed classes are present in the window, e.g. overlapped and adjacent text and image components in the area covered by w.

Figure 4: This example shows the effectiveness of suggested scheme for cases where image and text regions are very close to each other and regions do not have rectangular or even convex boundaries. For this example the results of prescribed post processing based on morphological operations are also illustrated.

Figure 5: This example shows a very difficult scenario where text is embedded in the image i.e. where different class of objects are overlapped. Even in this case the suggested method provides good results.

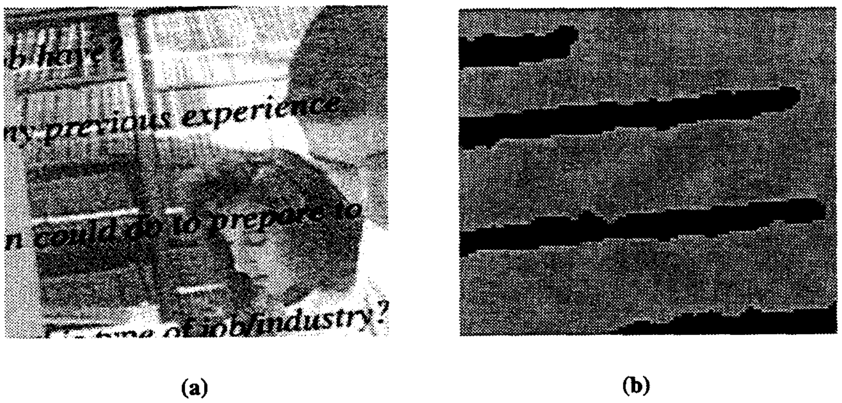
2.8 Results and Discussion

The decision integration scheme has been successfully used to identify text, images, and graphics areas on a document page. Improved performance is obtained by making soft decisions. The decision integration method is advantageous when confident hard decisions cannot be made because of poor features, resolution or windowing problems, noise, or the inherent fuzziness of some classification tasks. The scheme has yielded good results using simple feature sets and classifiers. It is based on a approach that can be applied to other segmentation and classification problems. Almost all the computations and decisions are made independently and in parallel, without iterative stages. Thus the scheme is


(a) (b)

(c) (d)

Figure 4: Irregular and non-convex image boundary very close to text.



(a) (b)

Figure 5: An example where text is embedded in image.

well adapted to fast implementation on distributed architectures.

References

[1] L.A. Fletcher and R. Kasturi, "A robust algorithm for for text string separation from mixed text/graphics images", *IEEE Trans. PAMI*, Vol. 10, pp. 910–918, 1988.

[2] D. Wang and S.N. Srihari, "Classification of newspaper image blocks using texture analysis", *CVGIP*, Vol. 47, pp. 327–352, 1989.

[3] M. Viswanathan and G. Nagy, "Characteristics of digitized images of technical articles", *Proc. SPIE*, Vol. 1661, pp. 6–17, 1992.

[4] F.M. Wahl, K.Y. Wong and R.G. Casey, "Block segmentation and text extraction in mixed text image documents" *CVGIP*, Vol. 20, pp. 375–390, 1982.

[5] T. Pavlidis and J. Zhou, "Page segmentation and classification", *CVGIP*, Vol.54, pp. 484–496, 1992.

[6] A.K. Jain and S. Bhattacharjee, "Text segmentation using Gabor filers for automatic document processing", *Machine Vision and Applications*, Vol. 5, pp. 169–184, 1992.

[7] R.R. Coifman and M.V. Wickerhauser, "Entropy based algorithms for best basis selection", *IEEE Trans. Information Theory*, Vol. 38, pp. 713–718, 1992.

[8] K. Etemad and R. Chellappa , "Separability based local basis selection for texture classification", *Proc. ICIP*, 1994.

[9] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd ed., Academic Press, 1990.

[10] B. Kosko, *Neural Networks and Fuzzy Systems*, Prentice Hall, 1992.

[11] K. Etemad, R. Chellappa and D. Doermann, "Document page decomposition by integration of distributed soft decisions", *Proc. ICNN*, 1994.

[12] J. Ghosh and A.C. Bovik, "Neural Networks for Textured Image Processing", in *Progress in Artificial Neural Networks and Statistical Pattern Recognition*, I.K. Sethi and A.K. Jain (eds.), pp. 133–154, North-Holland, 1991.

[13] A.K. Jain, *Fundamentals of Digital Image Processing*, Prentice Hall, 1989.

[14] P.C. Chen and T. Pavlidis, "Segmentation by texture using correlation," *IEEE Trans. PAMI.*, Vol. 5, pp. 64–69, January 1983.

[15] D. Rumelhart and J. McCelland, *Parallel Distributed Processing*, Vol. 1, pp. 322–328, MIT Press, Cambridge MA, 1988.

[16] K. Ramchandran and M. Vetterli, "Best wavelet packet bases in a rate-distortion sense", *IEEE Trans. Image Processing*, Vol. 2, pp. 160–175, April 1993.

[17] J.C. Bezdek and S.K. Pal (editors), *Fuzzy Models and Pattern Recognition*, IEEE Press, New York, 1992.

[18] R.L. Watrous, "GRADSIM: A connectionist network simulator using gradient optimization techniques", TR MS-CIS-88-16, University of Pennsylvania, March 1988.

3 Page Representation

Document understanding has attained a level of maturity that requires migration from ad-hoc experimental systems, each of which employs its own set of assumptions and terms, into a solid, standard frame of reference, with generic definitions that are agreed upon by the document understanding community.

The logical structure of a document conveys semantic information that is beyond the document's character string contents. To capture this additional semantics, document understanding must relate the document's physical layout to its logical structure. This work provides a formal definition of the logical structure of text-intensive documents. A generic framework using a hierarchy of textons is described for the interpretation of any text-intensive document logical structure. The recursive definition of textons provides a powerful and flexible tool that is not restricted by the size or complexity of the document. Frames are analogously used as recursive constructs for the physical structure description.

Papers covering all areas of document image analysis can be found in the Proceedings of the 1991, 1993, and 1995 International Conference on Document Analysis and Recognition [ICDAR]; the 1992, 1993, 1994 and 1995 Annual Symposia on Document Analysis and Information Retrieval, sponsored by the University of Nevada, Las Vegas; and the International Conferences on Pattern Recognition [ICPR]. For surveys of document image analysis and document image understanding see (Casey and Nagy, 1991) and (Tang et al. 1991).

3.1 Logical and Physical Document Description

In order to describe a document's physical and logical characteristics consistently, it is advantageous to first distinguish between the document's content and its structure.

3.1.1 Document Content vs. Document Structure

The *content* of a document is the information contained in the document, which is not bound to a particular representation format. At the lowest level, this information may include a stream of characters, and these characters make up successively higher-order objects built on top of each other such as words, sentences, paragraphs, etc., up to the complete document level.

The ***physical structure*** of a document is how the document's content is laid out on the physical medium. The same content can be organized in a variety of ways and therefore can have many physical layouts, which stem from different values of the attributes of the physical components (point size, line spacing, page size, etc.). The medium is traditionally paper, but may be any visual host, such as a computer screen or photographic film, which emulates the layout on the page. Bearing this extension in mind, we refer to paper documents as the classical representatives.

The *logical structure* of a document's content is how the content is organized prior to the enforcement of a particular physical structure. A text-intensive document, for example, typically consists of sentences, words and characters, and possibly higher-order constructs such as sections and chapters for an article, or address block, body and signature block for a business letter.

It is essential to note that the **same** content can be viewed with respect to both physical structure and logical structure. A major goal of this report is to describe the two types of structure and how they can interact.

3.1.2 Generic Document Structures

Text-intensive documents can be classified into many types including books (Over most text books, edited books, ...) technical papers, correspondence (business letters, memos, ...), newspaper articles, and conference proceedings. When attempting to describe the structural relationships between document components, it is essential to provide a level of abstraction, so as not to be caught up in the terminology associated with a specific type of document. To this end, we define terms that are generic at both the logical and physical levels. The *generic document structure* terminology allows us to define type-independent relationships among document components.

Following this rationale, we distinguish between the generic (both logical and physical) document structure and the instantiated generic document structure. An instantiation of a generic structure allows us to begin discussing a particular class of document, and the relationships between known entities. Such a distinction is in accordance with the basic concepts of object-oriented analysis (OOA), where objects are instances of their respective classes.

Terms such as column, line and symbol in the context of physical structure, or chapter, section, and subsection in the context of logical structure, are instances of the generic terms we define below. Some of them, such as the physical term "line" and the logical term "sentence", are applicable to a large variety of documents, while others, such as the logical term "session", are document-type-specific.

Finally, a *specific structure* (physical or logical) refers to the structure of a single given document, and describes the structure of that instance of the document.

3.1.3 Realizing Document Structures

The progression from generic to specific instances of a document is fairly straightforward, but the ability to reverse the process can be an important component of any analysis system. At the first level, the object class "Document" is instantiated as to its type, both physical and logical, according to the document content. The structure of each instance of the generic document is thus dependent on the document's type.

By making such distinctions, it becomes possible to prune the generic document tree to narrow the analysis to a given document type. For example, there is no use talking about a document's physical component called "volume" or the logical component "chapter" when the document under consideration is of type "business letter". Such distinctions can be made at an early point.

A second level of instantiation involves instantiating the object class of a particular document type, such as "Proceedings", to its specific structure, such as "Proceedings of Document Analysis Systems '94 Conference" (Dengel and Spitz, 1994).

We do not attempt to provide a full classification of all the text-intensive document types, let alone the particular logical or physical structure of each type. In the remainder of this section, however, we do provide a set of definitions and tools that enable this task to be done consistently and coherently, along with a few examples.

3.1.4 Generic Physical Structure

Following are the definitions related to the generic physical structure.

Frame – an area within a page of a document, which may consist of a collection of (lower level) frames and/or blocks.

Root Frame – a frame which encompasses the entire physical document.

Page – a frame which occupies a rectangular region of a page, on which the document, or part thereof, is physically recorded. A page is the basic physical document object.

Page Set – a frame which consists of multiple pages. An instance of a page set might be a single volume or chapter, for example.

Block or **Simple Frame** – a terminal, lowest-level frame, which, at the given level of granularity, need not or cannot be further decomposed.

A frame is defined to be a recursive component, as it may itself consist of one or more frames. A block is the terminal frame, which defines a region on the page and has content. Depending on the desired level of granularity, a block's content may correspond to a column, line, word, or character, for example.

Each type of frame has an associated set of attributes. Information about a frame's location, position on the page, justification, etc. is defined by its attributes. For example, a character is characterized by such attributes as font, boldness, point size, and inclination; a line, by length; etc.

Being the leaves of the tree, only blocks have content, while items higher in the hierarchy, which are compound frames, are lists of pointers to lower-level frames. The block's granularity must be at least as fine as the smallest logical component to be described. Thus, for example, if the lowest-level logical component is a word, then the block cannot be a line—it must be at least a word, and it may be a character. This data structure enables the reconstruction of the structure of a frame at each level.

Consider, for example, a paragraph which is split over two pages, or a word which is split over two columns. In both cases, a single logical component (paragraph or word) is a combination of two frames, which should be concatenated to yield the entire logical component.

If, on the other hand, a single physical component is found to correspond to more than one logical object, such as the string "092596", which corresponds to a date with three logical sub-components (month, day, year), then the physical representation may be subdivided to reflect the finer granularity. This is quite rare, however, as the physical structure is normally aimed at reflecting the logical one.

An instance of a text-intensive document may have blocks and frames corresponding to a

Character – a block containing the image of a symbol in the document's language.

Word – a frame containing a group of one or more aligned characters, separated by white space.

Word is both a logical object in a document (as we define below), which conveys a certain meaning, and a physical object—the frame (page area) containing the union of its constituent characters. If word is a block, then its content is the image of the word, else it is a frame consisting of characters (each of which is a block), with each character having its image. The specializations of word are as follows:

Subword – a frame containing a group of one or more aligned characters, which make up the first part or the last part of a word.

Preword – a subword containing the first part of a word. A preword is located at the end of a line and ends with a hyphen.

Postword – a subword containing the last part of a word. A postword is located at the beginning of a line and completes its preceding preword to a whole word.

Line – a frame containing 1) a collection of one or more aligned words and 2) at most one preword and at most one postword.

Stack – a frame containing a collection of one or more lines stacked on top of each other and possibly separated by a non-empty white space, such that its logical content is one paragraph[1] at most.

Column – a frame containing a collection of one or more stacks on top of each other. A page may contain one or more columns. In structured documents, this number is generally fixed throughout the document.

3.1.5 Generic Logical Structure

Like the physical structure, which is represented as a hierarchy of frames, the generic logical structure is similarly viewed as a tree, in which the leaves are typically the characters, or symbols, and the root is the entire document. This structure is depicted in Figure 6.

To be able to describe generically the logical document hierarchy that is not restricted by a particular number of levels and associated level names, such as "section" and "chapter", we define the term "texton" as the logical analog of "frame".

Texton – a logical component of a text-intensive document, which consists of one or more (lower level) textons or simple textons.

Root Texton – a texton which is the entire document.

Examples of root textons include book, encyclopedia, concordance, dictionary, journal, newspaper, magazine, report, scientific paper, cover letter and business letter.

Simple Texton – a logical component of a text-intensive document, which is not further divided.

Instances of a simple texton are paragraph, sentence, phrase, word and character. If, for example, word is the simple texton in a particular document, then any subcomponent such as a character is a *primitive texton* document.

Compound Texton – a texton consisting of a distinct header, body, and.optional trailer.

An example of a compound texton is a section of a document, which has a header (the section head), a body (the set of paragraphs), and no trailer. Another, less obvious example of a compound texton is a signature block in a letter, which contains a header (the closing), a body (the signature), and a trailer (the printed name).

As shown in Figure 6, scanning the logical structure from the top down, the entire document (encyclopedia, book, article, business letter, etc.) is the root texton—the root of the tree. Below it is a varying number of levels of textons. The black triangle along the paths connecting a whole to its parts in Figure 6 is the aggregation symbol (Dori, 1995).

Texton is the logical analog of the physical frame. Like frame, the definition of texton is recursive, and the halting condition is that the constituent texton

[1] Paragraph is a logical term defined below.

is a simple texton, i.e., the base logical unit, typically a character. The recursive definition of texton encompasses the entire spectrum of logical levels in any text-intensive document, just as the root frame encompasses all the physical levels.

Instances of a texton, in a text-intensive document, include:

Character – a texton which is a symbol in the document's language. Normally, character is a basic texton.

Word – a texton containing a sequence of one or more characters, which has some meaning in the document's language.

As we have noted, both character and word have logical as well as physical definitions. The difference between a logical character and a physical character is that a logical character is the symbol itself, while a physical character is the image representation of the logical character. Likewise, the difference between a logical word and a physical word is that a logical word is a semantic-conveying object, while a physical word can be considered either as the image representation of the logical word or as an ordered collection of its comprising physical characters. Note that there is no logical analog to the physical term subword, whose existence stems from spatial arrangement considerations.

We continue with the definitions of higher-level textons.

Phrase – a texton which is a meaningful collection of one or more words that do not necessarily form a complete grammatical sentence.

Examples of phrases include a title of a document or part thereof, a name of a person or an organization, an address, or a (possibly nested) itemized list of such entities.

Sentence – a meaningful collection of one or more phrases which correspond to a valid grammatical sentence, complete with punctuation.

Paragraphon – a texton which is a generalization of a paragraph. It consists of a group of one or more sentences and/or phrases.

Each one of the items above is an example of a paragraphon, where the title is a phrase, and it is followed by another phrase and optional sentence(s).

Unlike character and word, higher-level textons have different names than the corresponding frames, because the physical structure departs from the logical one, and the correspondence becomes more and more fuzzy as we climb up the two hierarchies. Thus, above word at the physical level is the frame called line, while the corresponding textons at the logical level are phrase and sentence. However, it is obviously unlikely that a single sentence occupies exactly one line. At the next level up, paragraphon is analogous to stack, but again, the correspondence is only partial, because a paragraphon may stretch across more than one stack, if it starts at the end of a column and ends at the first stack of the next column, or even across several whole columns and pages.

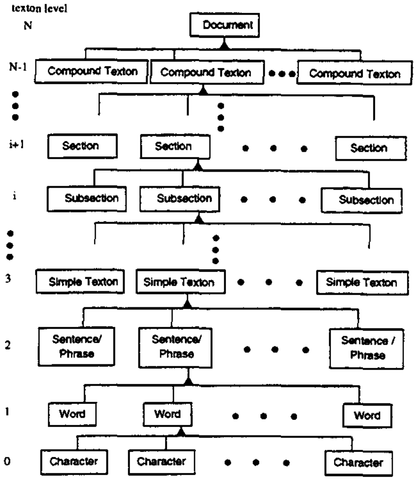


Figure 6: The logical structure of a document described as a tree

At yet higher levels, the relationships between textons and frames are type-dependent. For example, the texton chapter in a textbook normally starts at a new page, as does a paper in a proceedings.

A document may contain logical elements which are referenced from multiple independent points within the document itself. They are often self contained logical units (i.e., textons) and should be treated as such. To handle such components, we define a *referenced* texton.

Referenced Texton – a graphic or textual texton which is referenced from the document.

Examples of referenced components include figures, appendices, footnotes, citations, continuation text bodies (e.g. in newspapers) and even complete documents. The header of a texton is a label or identifier which is "referenced" by a pointer.

Pointer – a referencing texton pointing from the main text of the document to the referenced texton. This pointer is typically a phrase or a sentence, such as "continued on page 5, column 3", "see Figure X", or "(Author, 1995)".

Graphon – a referenced texton in a document whose nature is mainly non-textual, and whose function is to illustrate, explain or demonstrate the text. Examples of graphons are line drawings (engineering drawings or art-line), halftones, photographic (black and white or color) images, maps, diagrams, charts, tables, etc.[2]

In graphons, if text exists, it supplements or enhances the graphic. A graphon has graphic

[2] A table is a boundary case between text and graphon. We classify it as a graphon, because even though it contains text, the text normally does not have a definite linear reading order and it is normally enclosed within graphics—the lines that separate rows and columns.

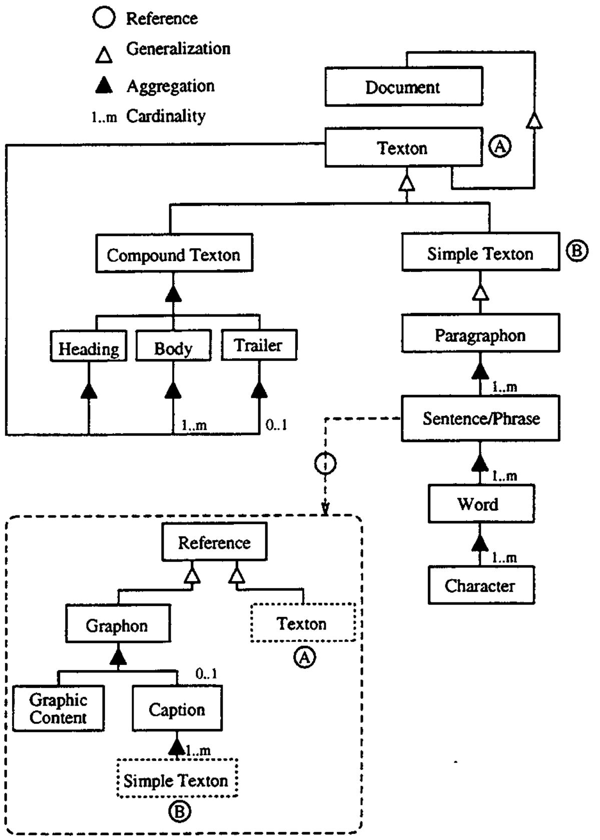


Figure 7: The object-process diagram of the generic logical structure of a text intensive document

(imagery or geometric) contents and an optional *caption*, which itself is a texton, and consists of a mandatory *caption header*(the graphon identifier), and an optional caption body(the textual title or explanation of the graphon). The caption header is mandatory, because it serves as a reference and is pointed to by the text.

Since a graphon, like the figures in this document, normally occupies a considerable portion of the page area, the physical location of a graphon is frequently allowed to *float* in the neighborhood of where it is referred to in the main text for the first time.

Finally, in addition to the hierarchical structure given by the recursive definition of the texton, the logical structure must also preserve the reading order.

Reading order – the order in which the characters or symbols in a text-intensive document must be traversed for the document to be correctly understood.

The reading order corresponds to a depth-first visit of the document's logical structure. Reading order normally makes sense only within and between the text-intensive components of a document. In the case of referenced textons, the texton appears physically only once, but may appear logically at many locations. A pointer denotes the logical appearance, so the reading order follows a round-trip "visit" to the referenced texton.

Graphons tend to "float" and can be referenced from multiple locations in the main text. Hence, like any referenced texton, the read-order is preserved by requiring a visit from the reference pointer (typically a phrase) to the graphon and back.

Document Complexity

We have seen that a text-intensive document has a hierarchy whose textons depend partially on the document's logical type and may represent chapters, sections, subsections, parts, etc. Hence, the number of texton levels in a document is finite, normally not greater than 10. This number depends on the nature of the document and indicates its structural complexity. The numbering of the levels is bottom up, with zero assigned to the character level.

The *logical complexity* of a document is the level number of the document's root texton.

Consider, for example, a journal paper, whose body consists of sections. The body of each section is a paragraphon. Assigning the level numbers 0, 1, and 2 to the character, word, and sentence levels, respectively, a paragraphon is a level 3 texton, and the entire document is a level 4 texton. Hence the complexity of this document is 4. If at least one of the sections is divided into subsections, and no subsection is divided into sub-subsection, then the document complexity is 5.

Although usually there is a relation between the document's size and its complexity, these two terms should not be confused. The size can be measured by the number of pages, words or characters. A dictionary, for example, may be a very large document, but its complexity is not necessarily high. Similarly, an outline may be relatively small, but may have much higher logical complexity.

Simple and Compound Textons

Having defined textons and their roles in the document logical structure, we turn to a more abstract and comprehensive description of logical document layout than the one given in Figure 6. Figure 7 is an object-process diagram, or OPD (Dori et al. 1995; Dori, 1995), which describes the structure of a document.

The object Document is a specialization of a Texton, which is the root of the structure. This is denoted by the generalization symbol—the blank triangle going from Texton to Document. Texton is a generalization of Compound Texton and Simple Texton. This is denoted by the blank triangle from Texton to both Compound Texton and Simple Texton in Figure 7. A *simple texton* is a generalization of a paragraphon.

A character is defined to be a *level 0 texton. A word* is a level 1 texton, as it consists of one or more characters, and a sentence is a level 2 texton. A simple texton in the main text of the document is therefore a level 3 texton. Below it in the main text reside the sentence or phrase (level 2 texton), the word (level 1 texton), and the character (level 0 texton). As we show below, these level numbers may vary for side text, such as the table of contents in a book.

Although in the simplest form, one may conceive of a primitive document consisting of a single character, perhaps conveying a coded message, a single-word document, a single-sentence/phrase document, or single-paragraph document, we consider the simplest document to be document which is a compound texton. Therefore, the minimal complexity of any document is 4. A simple document, such as a standard business letter, is an example of a level-4 document. It has a header (sender and recipient identification and subject), a body (one or more paragraphs: level 3 textons), and a trailer (salutation, signature, etc).

The black triangle between Compound Texton on one hand, and Header, Body, and Trailer on the other hand, is an aggregation (whole-part) relation, expressing the fact that a compound texton consists of these three parts. The default cardinality (participation constraint) of the aggregation symbol is (1..1):(1..1), i.e., exactly one (minimum 1 and maximum 1) part for exactly one whole.

Consider a texton of level n. The cardinality of the header of this texton is 1, i.e., there is exactly one texton of level $n-1$ which is the header of the level n texton. The cardinality of the texton's body is $1..m$, meaning that there are between 1 and many textons of level $n-1$ in the body of the level n texton. Finally, the cardinality of the (optional) texton's trailer is 0..1, i.e., there is at most one texton of level $n-1$ functioning as the trailer of the level n texton. The "0..1" next to Trailer indicates that Trailer is optional. In other words, a texton has either two or three parts and must have exactly one Header, one Body and at most one Trailer. In summary, Header has exactly one texton, Body has a number of textons between 1 and many (denoted "$1..m$" in Figure 7) textons, and Trailer, if it exists, has one texton.

For example, a section in a paper is a compound texton. It has a header (the section title); a body, consisting of one or more paragraphons; and no trailer. As another example, a textbook is a compound texton, whose header is everything from the beginning of the book to the beginning of the first chapter. The body of the book consists of a number of chapters and its trailer is everything from the end of the last chapter to the end of the book (appendices, glossary, index, etc.).

The Recursion in Texton Definition; the Body Path

Since Compound Texton is Texton, and Compound Texton has Header, Body and Trailer, each having at least one Texton, we get a recursive definition. As in any recursion, to avoid infinite looping, a halting condition must exist. The halting condition, as expressed in the object-process diagram of Figure 7, occurs when the textons of Header, Body and Trailer of the Compound Texton are all Simple Textons. When a texton is simple, the recursion stops, because from this level downward, we descend through the phrase level and the word level down to the character level. In the case of a referenced texton, the pointer—a Simple Texton in the main text—links it to preserve the reading order, while the referenced texton itself is a Compound Texton, whose header is the identifier the pointer points to.

Body Path – is the path in the tree structure going from the root node—the entire document—through successively decreasing levels of compound textons, all the way down to the simple texton (the paragraphon level), such that the path always visits the body of each texton. Since by definition any compound texton has a body, such a path is guaranteed to exist, and it is unique.

The level of a character, which is the last node—the leaf—along the body path, is defined to be zero. This implies that along the body path the level number of a word is one, the sentence/phrase level is two, and the level of the paragraphon—the simple texton—is three. Note that these numbers are not necessarily the same for characters, words, sentences and paragraphs which are not nodes along the body path. As we show in the example below, the level numbers may be higher or lower than the ones along the body path, depending on whether the path from the top texton (the document level) is longer or shorter than the length of the body path. The fact that of the three compound texton parts only two are mandatory gives rise to a 2-3 tree structure, as we demonstrate in the example in the next section.

DAS94 Proceedings—a case in point

To demonstrate the use of the concepts and terms presented above, and to show how document complexity is defined, consider the document *Proceedings of DAS94* (Dengel and Spitz, 1994). The structure of the document is described in Figure 8. The structure is detailed down to the Simple Texton level. Since a compound texton may have either three parts (Header, Body, and Trailer) or two parts (Header and Body), the resulting structure in Figure 8 is a 2-3 tree.

As indicated in the legend of Figure 8, the body path is marked by thick line segments. The level numbers are written in parentheses next to the corresponding textons along the path. The body path visits the nodes "Proceedings of DAS94", "Session", "Paper", "Section", "Subsection", and "Paragraph", in that order. Assigning the number 3 to the paragraph level and counting up we find that "Subsection" is at level 4, "Section" is at level 5, "Paper"

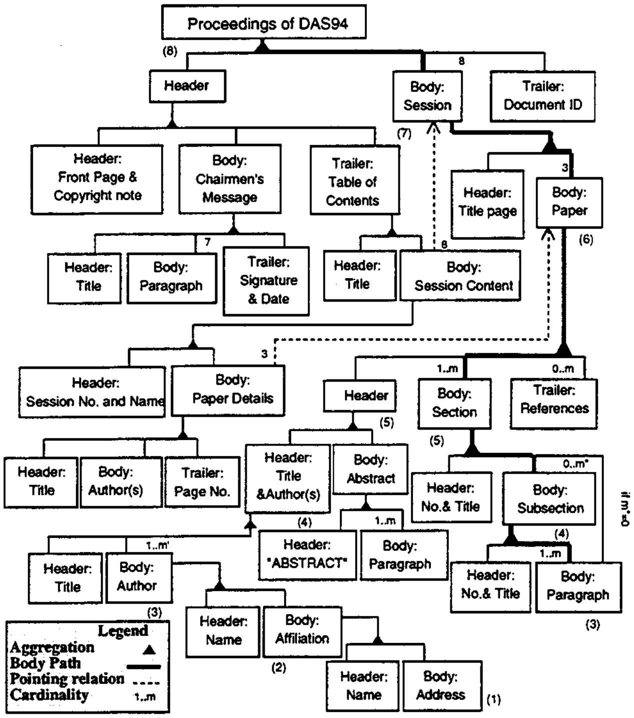


Figure 8: An OPD describing an instance of a generic logical structure—the Proceedings of DAS'94 (Dengel and Spitz, 1994)

is at level 6, "Session" is at level 7, and the entire document, "Proceedings of DAS94", is at level 8. Hence the complexity of this document is 8.

As a compound texton, "Proceedings of DAS94" has a header, a body and a trailer. The header is a level 7 texton, which, in turn, consists of three level 6 texton: a header—"Front Page" and "Copyright note", a body—"Chairmen's Message", and a trailer—"Table of Contents". Chairmen's Message consists of a level 5 header—the title "Chairmen's Message," a level 5 body, consisting of seven level 4 paragraphons, and a level 5 trailer, containing two level 4 paragraphons. The first phrase is "Kasierslautern, October 1994," and the second is the names of the two document editors. As we see here, both the paragraphs and the phrases, which are basic textons, are at level 4 rather than 3. The reason is that the path traversed here is not the body path. As already noted, a basic texton is guaranteed to be at level 3 only when it is on the body path. In other paths it may be more (as here) or less than 3. The path that ends with "Author", "Affiliation", and "Address", for example, is the longest one. It is longer by two edges than the body path. Therefore "Address", which is a simple texton, is a level 1 texton in this case, as shown at the bottom of Figure 8. Table of Contents is a level 6 texton, consisting of a header—the title "Table of Contents," a body, and no trailer. The body of the Table of Contents consists of eight items. Each item is a level 5 texton called Session Contents. It has a header—session number and name, a body—a phrase (itemized list) of three level 4 items, each called Paper Details, and no trailer. Each Paper Details item is a level 3 texton. It consists of three paragraphons, each containing a single phrase. The first phrase is the paper name, the second phrase is the author name, and the third phrase is the page number.

In most of the papers, Section consists directly of paragraphs, but several papers have subsections (see for example page 139 in the document). To accommodate this variability, we add the condition "if $m'' = 0$" along the aggregation link from Section to Paragraph in Figure 8, where m'' is the number of subsections in a section. This means that if there are no subsections in the section, then Section consists directly of paragraphs.

3.1.6 Relating Physical and Logical Structure

Having defined generic terminology for both logical and physical structure, it is straightforward to relate the two at the content level, in most cases. Figure 9 shows the structure of a simple document, a multiple page chapter. The chapter is a compound texton, with a header (title) and two body components (one abstract and one section). The abstract is a simple texton and the section is a compound texton, consisting of two simple textons (paragraphons).

The physical structure subdivides the document into rectangular blocks. The content is shared in both structures. Note that the structure allows logical components (i.e., the abstract) to be split over two pages. For most documents, the logical complexity will be higher.

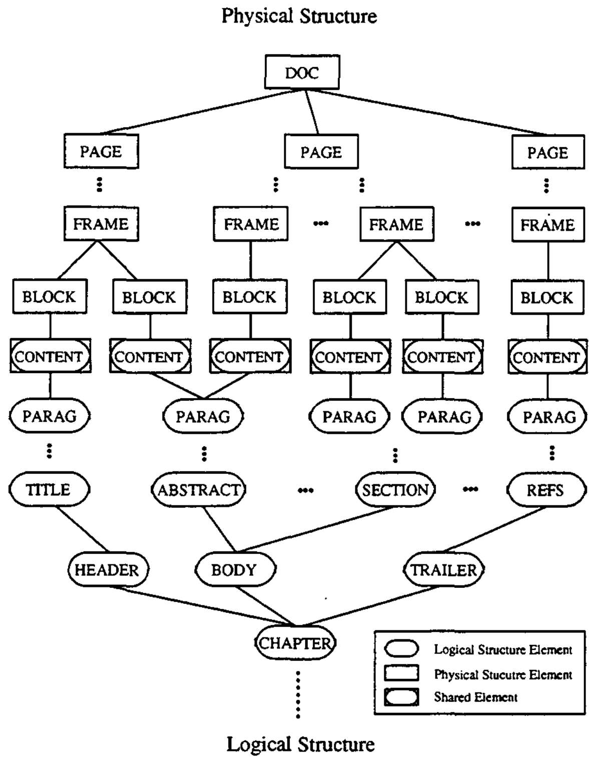


Figure 9: A sparse schematic representation of the physical and logical structure of the content of a document "chapter".

3.2 Summary

The structure of a document conveys semantic information that is beyond its character string contents. To capture this additional semantics, document understanding must perform a reverse encoding of the document and relate the physical layout to the logical structure. This work has proposed a formal generic framework for the definition and interpretation of any text-intensive document's physical and logical structure, that is not restricted by the size or complexity of the document. The physical document is described by a hierarchy of frames and the logical structure of text-intensive documents is described as a hierarchy of textons. The definition of textons provides a powerful and flexible tool for document logical structure analysis. We also propose a method for determining quantitatively, in an objective, reproducible, and unbiased way, the complexity of such documents.

We have also presented a description of a new and powerful document attribute format specification, DAFS, which provides mechanisms for representing and maintaining both physical and logical information during the reverse encoding process, and have shown how it can be used to relate logical and physical structure at the content level.

***This work done in conjunction with researchers at the Technion and the University of Washington.*

References

[1] R.G. Casey and G. Nagy, "Document Analysis—A Broader View," *1st ICDAR*, Saint-Malo, 1991, pp. 839–849.

[2] A. Dengel and L. Spitz (Eds.), *Proceedings of DAS94—Document Analysis Systems*, Kaiserslautern, Germany, October 18–20, 1994. DFKI Document D-94-06.

[3] D. Dori, "Object-Process Analysis: Maintaining the Balance Between System Structure and Behavior," *Journal of Logic and Computation*, 5, 2, April 1995, pp. 1–23.

[4] D. Dori, I. Phillips and R.M. Haralick, "Incorporating Documentation and Inspection into Computer Integrated Manufacturing: An Object-Process Approach". In *Applications of Object-Oriented Technology in Manufacturing*, S. Adiga (Ed.), Chapman & Hall, London, 1995.

[5] Y.Y. Tang, C.Y. Suen, C.D. Yan, and M. Cheriet, "Document Analysis and Understanding: A Brief Survey," *1st ICDAR*, Saint-Malo, 1991, p. p17–31.

4 Other Topics

Short descriptions of several recent projects are included below. Additional information and papers about these and other topics can be found on the WWW page http://documents.cfar.umd.edu.

4.1 Handwriting Analysis

The purpose of this research is to advance automated recognition of handwritten Latin text. Two key factors which significantly effect the performance of recognition systems are segmentation and classification. To achieve an acceptable level of performance, we are exploring ways to perform segmentation and classification concurrently.

The current approach uses model-based stroke detection. To simplify the problem, we assume we are given a model in the form of a set of strokes that are typical of a specific writer. Since the model represents uncorrupted strokes, including what may be classified in the image as retraced strokes, we have an added dimension to aid in the segmentation of the image. Using a model-guided approach, we classify strokes simultaneously based on the model. Once an understanding of the script writing of specific individuals is achieved we may extend this approach to generalized script writing.

The current method which identifies strokes is an energy-based line follower. This method follows an unbroken line segment until it arrives at an ambiguous region (or junction), then performs energy minimization to obtain the ideal traversal. This energy is calculated by considering a combination of factors which involve the image and model information. Matching of the traversed stroke with the model gives us an estimate of the direction. A successful segmentation results in a valid classification for unambiguous strokes. Our method has shown promise on simple numerals but needs improvement for generalized segmentation.

Snakes can be used to further integrate the segmenter with the recognizer. Snakes are adaptive splines which allow graceful degradation with increased deviation from a model. Minimum degradation can be tuned to allow for a set of known deviations over other unlikely deviations, in an attempt to maximize overall performance. The set of snakes were model-based only in terms of the allowed set of deviations. We plan to introduce a model spline along with a tunable degradation model.

4.2 Hybrid Thinning

One difficulty with many pixel-wise thinning algorithms is that they produce unacceptable results at junction points, or in the presence of contour noise. The primary advantages are speed and simplicity, but such approaches suffer greatly from their myopic view of the data. Non-local algorithms, on the other hand, can perform much better on intersecting or noisy strokes, but can also be complex and prohibitively slow.

In response, we present a novel approach to detecting ambiguous regions in a thinned image. The method uses the reconstructability properties of appropriate thinning algorithms to reverse the thinning process and automatically detect those pixels which may have resulted from more then one stroke in the image. The ambiguous regions are then interpreted and reconstructed using domain specific or derived contextual information. The approach has the advantage of using local methods to rapidly identify strokes (or regions) which have been thinned correctly and allowing more detailed analysis based on non-local methods in the remaining regions.

We have developed a hybrid approach to extracting the skeleton of a binary pattern that has two advantages: First, the ability to quickly identify those areas of the pattern that were successfully thinned using local methods and those where ambiguity exists. Second, the ability to perform a more detailed evaluation, using context, in areas that were not successfully thinned. This algorithm can detect and reduce the number of erroneous "spurs", reduce sensitivity to thresholding, and reduce sensitivity to local noise.

The hybrid algorithm uses local methods to perform an initial skeletonization of the image. Strokes which are elongated and have slowly varying width are thinned correctly. By growing the thinned strokes back to their original width, again using a local process, we can identify regions which may have resulted from more then one stroke, and label them as ambiguous. Non-local methods are then used for only these regions.

Figure 10 shows three examples.

4.3 Synthetic Data for Text Understanding

In this project, we describe work on a system for modeling errors in the output of OCR systems. The project is motivated by the desire to evaluate the performance of various text analysis systems under varying, yet controlled conditions. We have described a set of symbol and page models which are used to degrade an ideal text by introducing errors which typically occur during scanning, decomposition and recognition of document images. A first generation of the software implements the page models and allows the use of transition probabilities, either extracted from real data or generated synthetically, to corrupt text.

It is often advantageous to use synthetic data for any component of a large system whose input data may be noisy or otherwise unpredictable. For such systems, the amount of data necessary to cover a sufficient cross section of the possible inputs may be prohibitive. For example, in document image understanding systems, the input is typically a scanned image of a physical document. The types of degradations which appear in the input data may be introduced during the production, manipulation or imaging of the original "ideal document". Although examples of degradations from any given source can typically be obtained, obtaining a given combination of degradations with known magnitude may be difficult. In many domains, the cost of data capture alone can be enormous, and the results still may not represent a useful distribution of expected inputs. The document understanding community is benefiting from a number of approaches to modeling character image and page image distortions and degradations [1-3, 7]).

We are developing a system which makes use of enhanced synthetic data for the development and testing of information retrieval (IR) and related text analysis systems. There have been several recent papers which explore the effects of OCR errors on IR accuracy. Croft et al. [4] have considered the effects of different commercially available OCR systems on retrieval accuracy. They concluded that devices with high word OCR rates have minimal effect on retrieval, but as word recognition rates decrease, a significant decrease in retrieval is observed, especially in short documents. More specifically, Taghva et al. [5] evaluated the effect of OCR errors on the vector space model for IR. They found that although the *average* precision and recall from the vector space model are not significantly affected by common OCR errors, when different weighting schemes are applied, the rankings observed between the OCR version and a "corrected" version are affected. Finally, Taghva et al. [6] showed that the difference in performance of a set of queries on an OCR database and a corrected version is not significant.

We explore the development of models for OCR output, independent of the ultimate analysis task. We use the current perceived state of OCR and doc-

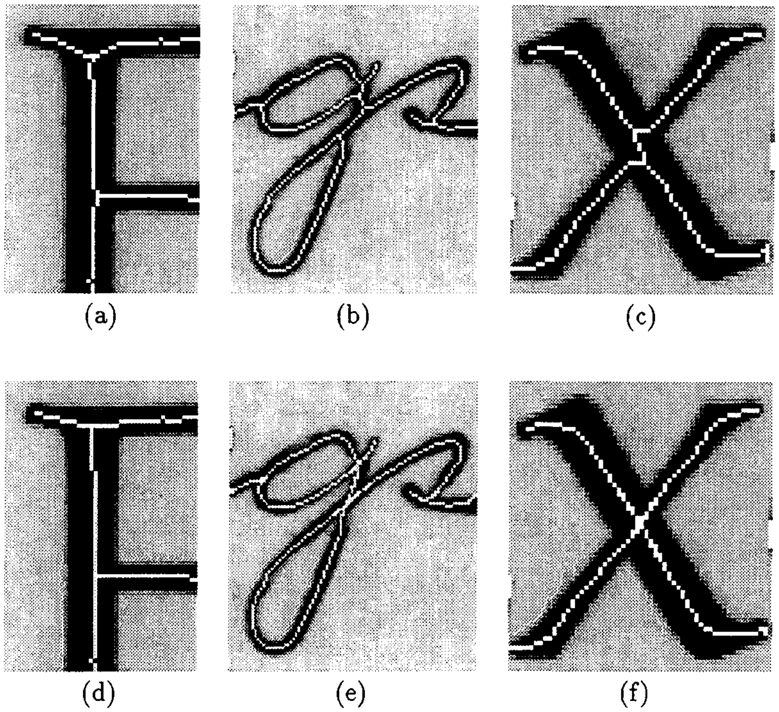


Figure 10: Examples – Row 1: pixelwise method, Row 2: hybrid method

ument analysis technology, including types of errors, to define the classes of models. We present a method for defining symbol set independent classifier models and give several examples, and we describe several page models. We provide a software tool-box (available upon request) which can be used to model a series of symbol level and page level events which are common and may result in errors being produced by the OCR system.

4.4 Text Image Compression

In ongoing research, we are developing SCODI, a system for the compression of text intensive document images. The primary objective of the system is to provide an encoded version of either a single or multi-page document image. A secondary objective is to provide an encoding which allows near random access of regions within the image and facilitates traditional document processing operations.

We describe an encoding procedure which makes use of the redundancy in the symbol bitmaps. The system uses an efficient representation which allows algorithms for tasks such as skew detection, segmentation, block classification, and keyword searching to be performed without requiring a complete decoding of the image.

It is generally agreed that maintaining a scanned document in image form is necessary to avoid both the high cost of manual conversion and the introduction of a large numbers of errors during automatic segmentation, OCR and/or graphics interpretation. In addition, if we require that we be able to regenerate the original image with arbitrary precision, even partial OCR with an ASCII representation is not an option.

Text intensive document images typically have a great deal of redundancy in the bitmap representation of symbols. We attempt to make use of this redundancy by encoding sets of similar symbols once, and representing symbol by the encoded representative and a difference. Typically the differences will represent noise on the symbol boundary due to quantization. If this noise can be characterized, we can provide a lossy scheme which ignores these extraneous pixels.

In our approach, small regions are believed to correspond to textual symbols are identified in the image. They are clustered and a template is created for each appropriate cluster. The regions are then represented uniquely by a combination of this template, its location in the original image and an encoding of the difference between the template and the original image. Tiles are used to encode non-symbol-like regions.

Clearly, the compression ratio is dependent on having documents which are large enough to contain multiple instances of the same symbol, and on having noise levels low enough so that difference encoding is minimal. We anticipate the largest compression to occur on multi-page documents in which the relative overhead is reduced.

References

[1] *University of Washington English Document Database I CDROM: Distortion Software—DDMPHOTO*, 1993.

[2] H.S. Baird, "Document image defect models". In *Structured Document Image Analysis*, pages 546–558. Springer-Verlag, 1992.

[3] M. Buchman. DISTORT *Software*, 1993. Available via www at documents.cfar.umd.edu.

[4] W.B. Croft, S.M. Harding, K. Taghva, and J. Borsack. "An evaluation of information retrieval accuracy with simulated OCR output. In *Symposium on Document Analysis and Information Retrieval*, pages 115–126, April 1994.

[5] K. Taghva, J. Borsack, and A. Condit. "Effects of OCR errors on ranking and feedback using the vector space model". Technical Report 94-06, Information Science Research Institute, University of Nevada, Las Vegas, August 1994.

[6] K. Taghva, J. Borsack, A. Condit, and S. Erva. "The effects of noisy data on text retrieval". *Journal of the American Society for Information Science*, 45(1):50–58, January 1994.

[7] C.H. Tung, Y.J. Chen, and H.J. Lee. "Performance analysis of an OCR system via an artificial handwritten Chinese character generator". In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 315–318, 1993.

Document Classification and Functional Decomposition

Suzanne Liebowitz Taylor Mark Lipshutz Roslyn Weidner Nilson
Loral Defense Systems-Eagan
Paoli, PA 19301-0517
{suzanne,mark,roslyn}@vfl.paramax.com

Abstract

Document classification and functional decomposition are key components for processing heterogeneous batches of documents. We describe our approach and recent results in both these endeavors. Classification is performed in two steps: first, the document is sorted by the number of columns and second, functional landmarks are detected to determine the class. Two methods for functional decomposition are also described: one, using content (or text-based) clues and the other using only geometric clues. Results for classification and functional decomposition are provided for business documents.

1 Introduction

This paper describes extensions to the Intelligent Document Understanding System (IDUS)[11] in the areas of document classification and functional decomposition. Our development goal is to find single-page business documents from heterogeneous document batches and decompose these documents into their functional components. This decomposition enables sophisticated retrieval operations in which one can pose queries (such as the one in Figure 1) based on content and/or document objects and the roles they play.

Documents in the business domain (such as business letters and memoranda) are rich in meta-information such as sender, recipient, date, page numbers, etc., as well as the main body content which usually contains thc topical information. Although the business domain is constrained in the sense that there is a basic underlying organizing principle, there is enough variation in layout due to writer's style, non-native authors and cultural differences, that the problem is quite challenging.

Several other systems have addressed the issue of functional decomposition of documents [1, 5, 6], some specifically for business letters [3–5, 7, 8]. In general our environment seems to be more "unconstrained" than in other systems. We are quite liberal in our definition of "business document" as we are not tied to any particular method or result of physical segmentation, do not rely on a database of anticipated recipients, and do not expect there to be a one-to-one correspondence between physical segments and functional components. Again, this is due to the wide range of style and cultural differences.

The document classifier and functional decomposition modules were developed within the framework of the IDUS system [10, 11]. IDUS consists of components for layout analysis: physical, logical and functional decomposition. We distinguish between logical and functional decomposition in the following way. Logical organization groups physical components appropriately (e.g., into articles on a newspaper page) and sequences them in the correct reading order for further text analysis. Functional decomposition discovers the underlying functional role of each component (e.g., title, inside address, date) within an article or document. After processing, articles/documents may be stored in a corpus for subsequent retrieval.

The classifier is discussed in Section 2 and Section 3 details the functional analysis component. Results and system performance are presented in Section 4.

2 Document Classifier

A document classifier plays two important roles in a document analysis system: one, to find particular class or classes of documents and two, to sort all documents into classes by type (such as newspapers, business letters, or technical journals).

The classifier architecture (Figure 2) is a two-layer hierarchy, designed to classify one-page documents (or classify multiple page documents from their first pages). Full details are provided in [12] and summarized here.

The document is first sorted by the number of columns in the document. In some instances (such as technical journal articles) a class may have more than one column configuration, but for many classes (such as business letter or memorandum as a one column document or a newspaper page as a "more than two" column document) it is unlikely for the document to fall into more than one class.

After column detection, the document is passed to all classification engines which fall under the same

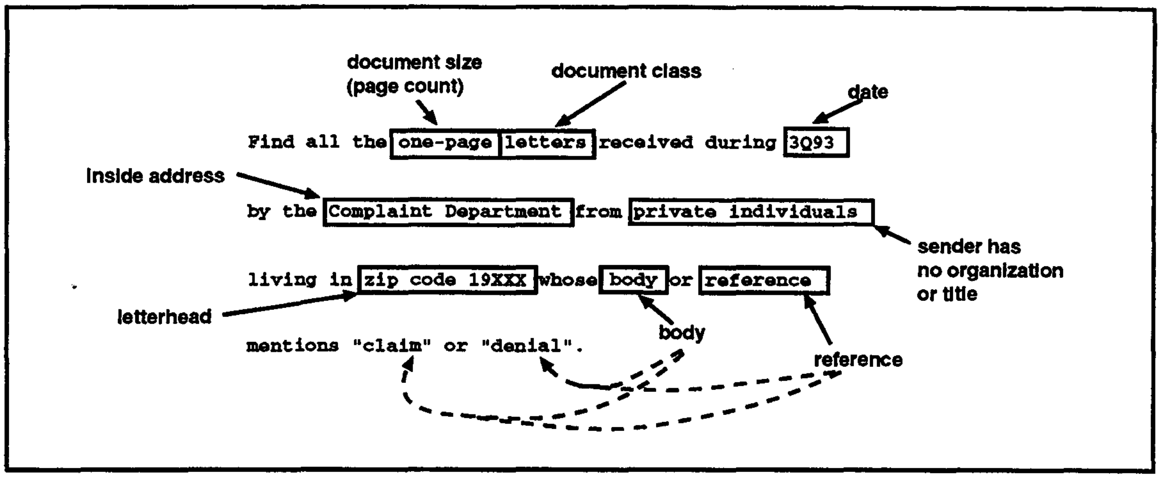

document size
(page count)
document class
date
Find all the one-page letters received during 3Q93
Inside address
by the Complaint Department from private individuals
sender has no organization or title
living in zip code 19XXX whose body or reference
letterhead
body
mentions "claim" or "denial".
reference

Figure 1: Document database query using functional decomposition.

column configuration. Each classification engine is based on the presence of key functional components or *landmarks* for a particular class. The output of each classification engine is either *yes* or *no*. If none of the functional classification engines produces an output, we implement a backtracking mechanism which tests all remaining classes (regardless of column configuration). After this second pass, if none of the remaining functional classification engines produce a positive output, the document is classified as *unknown*.

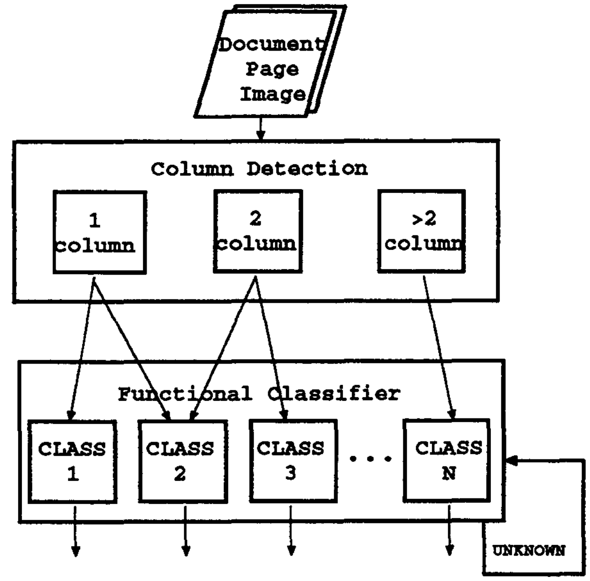


Figure 2: Two-layer document classifier.

In the classifier, if a document contains a particular number of the *landmarks* for a class (this number can vary for the class), then it is categorized as that class. Currently, it is possible to label a document as belonging to more than one class, although plans to implement conflict resolution are in progress. We found one or two cases where a document could reasonably be classified as either a memorandum or a business letter. For example, the document contained a *salutation* typical of a business letter (e.g., `Dear J. Doe`), but also header fields typical of a memorandum (e.g., `Re: Meeting on Friday`). Because of this type of ambiguity and the anticipation of adding more classes, conflict resolution will become imperative in the future.

The lower layer of the classifier uses the presence of functional components as discriminating features. In order to determine the most likely features for each class, we ran the test data through our content-based functional analysis module (Section 3.1) and scored the frequency of each component's occurrence for the true class and false class data. We chose landmarks that were very likely to occur in the true class and much less likely to occur in the false class.

The rules to compute these functional components were decoupled from the functional module and applied separately for classification. However, no positional relational information (such as "the title block in a journal article is above the author block") was used in order to keep the decision schemes as simple and fast as possible. If the original rules as implemented in the functional analyzer contained any inter-component dependencies, these dependencies were removed from the classifier version of the rules.

The classifier has been designed for English-, French- and German-language business letters and English-language memoranda.

3 Functional Decomposition

For our main thrust – English-language business letters – two approaches to functional decomposition were undertaken. Initially, we worked on a content-based approach, which while also taking positional knowledge into account, focused on string matching of keywords associated with particular components and thus required OCR. This had the advantage of being able to find functional components embedded within physical segments, but bore the disadvantage of being computationally expensive. Fuller coverage to this approach is given in [9] and a summary is provided in Section 3.1.

Our current pursuit is a geometric-based approach, which is OCR independent and executes quickly, but completely relies on the physical segmentation. Intra-object knowledge is implemented via fuzzy logic; inter-object knowledge with standard predicates. Section 3.2 provides coverage of this method.

3.1 Content-based Approach

After scanning, our initial step is a physical segmentation to obtain objects on which to test functional labeling rules. As we will show, we do not assume a one-to-one correspondence between physical segments and functional components. In fact, this method seeks functional components even if they are "buried" inside physical segments or span multiple physical segments. Functional labels are mapped to physical segments (or portions) via a process whose rules include intra- and inter-object knowledge of various kinds.

Inferencing is carried out in four phases as follows:

- In Phase 0 we perform initialization including determination of column boundaries [13], which gives us the width of the *content fill* (as opposed to the *stationery frame*) and allows us to preliminarily identify and set aside likely body blocks. In addition we seek certain "landmark" components. By exploiting the positional relationships of blocks to these landmarks, we are able to greatly reduce the search space for functional components associated with these blocks.
- In Phase 1 we find components which are coincident with physical segments. This search is driven by the list of unfound components, not the list of unlabeled blocks.
- For those blocks not yet labeled, we assume in Phase 2 that a block may contain more than one functional object. Using bipartite divisions of blocks, we examine each member of each partition to see whether it is coincident with a functional component.
- As in Phase 1, Phase 3 is predicated on having each physical block contain exactly one functional object, but in this case the rules are based on relative position rather than content. Thus, we exploit the partial order of physical segments and the partial order of labels in the quest to find functional components. Merging operations are also performed in Phase 3 to locate contiguous blocks with the same label.

Each rule is aimed at labeling a particular functional component, although it may contain references to others.

3.2 Geometric-Based Approach

Principally to avoid dependence on OCR and secondarily to reduce the computational expense of string matching, we have more recently pursued a content-free approach. We use only the features derived from the geometry of the physical segmentation and the basic datatype of the block (*text* or *image*).

The features employed are:

- X origin of the block relative to that of the page
- Y origin of the block relative to that of the page
- Block width relative to that of the widest (and presumably only) column on the page
- Line count (estimate)

The intra-object knowledge is applied using fuzzy logic [2], combinatoric methods based on fuzzy sets. Unlike an ordinary (or "crisp") set, a statement about the membership of an element in the fuzzy set is not usually a predicate. That is, there is a degree of membership associated with each element.

Reasoning on such entities aligns nicely with human intuition in many domains, for in real life decision making is often imprecise and must deal with partial or conflicting information. Fuzzy logic is an extension of classic logic, augmented with fuzzy implication and inference.

The functional components we seek are the *body*, *date*, *inside_address*, *salutation*, *sender*, *closing*, *letterhead_top*, *letterhead_bottom*, *signature* and *administrative*, of which the last two actually are sought chiefly by inter-object means. As for the *administrative* component, we formerly sought separate components for the *secretary*, *distribution* and *enclosure*. However, without content information, we do not feel we can reliably find those individual components and have lumped them together into one *administrative* unit. Similarly, we do not look for a *reference* component at this time.

The goal of the labeling process here is to find the best functional labeling of the given set of physical blocks. Notice that we are constrained by the physical segmentation in the sense that without content analysis we can not get inside a block; therefore, we have no way of knowing whether a block contains (portions of) multiple functional components. Accordingly, each block can be assigned at most one functional label and no splitting will occur. However, this does not preclude reasoning about larger-grained entities in that we can merge blocks with the same label when it makes sense to do so, e.g., the *body*.

3.2.1 Fuzzy Labeling Rules

There is one basic fuzzy rule per functional component plus variations to accommodate the fact that it is reasonable for certain components to appear at more than one location. For example, the *date* might be left justified or somewhere in the right half of the page, so we have a fuzzy rule for each possibility. For this component only one block could be assigned the label. However, in the case of *body* or *letterhead_top*, multiple assignments of the functional label are allowed.

Each rule assigns a score in the range [0,1] to the block under test as a candidate to bear a particular functional label. Given a raw score, experimentation has shown that simple numeric filtering should be applied to determine whether it should be retained. These filters are applied both at the component level (a block must score at least 0.70 to be a candidate for labeling as *body*) and overall (any score less than 0.20 is deemed insufficient to justify assignment of a label).

We have found that just two types of fuzzy functions suffice for our rules: the "Pi" function, which tests whether a feature is around a certain value and the "S" function where the distribution of a feature's values corresponds to a growth (or decline) pattern. In Table 1 we show the correspondence between features and fuzzy function types in our system.

A rule's overall score results from a combination of the fuzzy scores derived for the individual features contributing to the rule. Following convention we "AND" together the individual feature scores by taking the minimum of the set. Convention also dictates that we "OR" together the scores of the variations of a rule corresponding to the different physical locations at which some components may be found. The process of evaluating expressions of fuzzy functions is called *fuzzification*. Later we will discuss the corresponding process of *defuzzification*.

A sample fuzzy rule for the *inside address*, stated in English, is:

```
If relative X coordinate is near beginning
     of widest column
  and relative Y coordinate is near 0.20
  and relative block width is likely less
     than half that of the widest column
  and line count is near 4.5
  then the segment is the inside address
     with a score of 0.DDD.
```

3.2.2 Conflict Resolution

Our earlier method, centered on monotonic labeling, was contrary to a generate and test approach. One reason we chose monotonic labeling was the relative expense of the character-by-character comparisons at the heart of content-based labeling process, meaning we severely needed to prune the search space to maintain efficiency.

By contrast the fuzzy rules execute so quickly that it is efficient to apply the rule for each functional component to each physical block, and then perform conflict resolution. When all the rules have fired in a fuzzy logic system, the result is the definition of a fuzzy output space. The process of mapping back to a representative solution value is known as *defuzzification*. In our application conflict resolution plays that role by determining the "best" functional label for each block (which may be no label) from the set of fuzzy scores.

Conflict resolution proceeds as follows. A priority has been established for the functional components where the ones whose location is thought to be the most independent of other components head the list. For each component other than the *body*, *letterhead_top* or *letterhead_bot* we find the block with the highest score. If this block has been consumed (i.e., labeled), we take the block with the next highest score for this component. If this block has not been consumed, but there is another component which has the highest score for this block, again we try the block with the next highest score for this component. This could eventually result in a label's not being assigned to any block. Finally, some inter-object constraints are applied such as ensuring that a block to be labeled as *sender* is below the block labeled as *body*.

We then martial additional inter-object knowledge to make certain that the labeling is consistent with intuitive expectations. For example, we enforce that a block labeled as *sender* is horizontally aligned with and closely below the one labeled *closing*. If there is an *image* block between them, then we assign that the *signature* label. At this point we also try to identify the letterhead components, which must be above or below every other labeled component; merge *body* and letterhead blocks where appropriate and go after the *administrative* component, which involves more fuzzy function calculations.

4 Results

The current implementation of IDUS runs on a SPARC^{TM}20 with the UNIX^{TM}operating system. The IDUS system is written in the C programming language except for its reasoning component of the layout analysis module (for logical and functional analysis) which is written in Quintus^{TM}Prolog. OCR and segmentation are performed with the Xerox Imaging Systems ScanWorX^{TM}Application Programmer's Interface toolkit. The core functionality is accessible via both an X-Windows^{TM}Motif^{TM}user interface and a command user line interface.

The data set used for these experiments consisted of one page documents. Our training and test data have been culled from a variety of sources, including individuals, law firms, banks, insurance companies, industrial corporations, universities and (U.S.) government agencies. We were also provided with a set of documents by the DOD. These were letters written to the U.S. Library of Congress from other countries, i.e., meaning that English was not the native language of the authors. In general, the data represent a wide variation in layout, which has a concomitant effect on segmentation and the subsequent labeling. The documents were scanned as binary images at 200 dpi (dots-per-inch). This scanning resolution was chosen because of a requirement to minimize the reliability on OCR. Therefore, there was no reason to use the extra storage space or greater processing times required for higher resolution images.

Table 1: Features employed in fuzzy functions

	Origin Relative X	Origin Relative Y	Relative Block Width	Line Count
Pi	x	x	x	x
S		x	x	x

4.1 Classifier Results

Training images were used in the design of the classifier, either for finding the frequency of features from the functional module or for refinement of component rules for the classifier. The test data were used for blind testing of the classifier. The false class data ("other") consisted of one column documents which were neither business letters nor memoranda, and their corresponding classification should be *unknown*.

The business letter classification engine was developed using a training set (manually sorted for class) consisting of 92 business letters (true class) and 23 additional one-column documents from an assortment of other domains. The functional module was run on all of the training data to determine the *landmark* components. The functional components available for the letter class are: *letterhead*, *reference*, *date*, *inside address*, *salutation*, *body*, *closing*, *sender*, *distribution*, *enclosure* and *secretary*. The following landmarks were used to design the classifier for the business letter class: *salutation*, *closing*, and *inside address*. Functional analysis for memoranda consists of finding three components: *header*, *body*, and *trailer*. We found that subcomponents of the *header* were the most discriminating for that class. Therefore, to classify the the memoranda class, we identify two or more specific *header* subcomponents, e.g., *to*, *from*, *subject*, by looking for specific key words at the beginning of lines. The decision to require more than one *header* component resulted from the observation that many times lines in any document will begin with the word *to* or *from* (common key words in memorandum header material). In addition, we restrict the search for these *header* subcomponents to the upper portion of the document.

The classification results for both the training and test sets are summarized in Figure 3 and Table 2. The business letter class results are presented by written language in Figure 4. We scored the "other" data as if they should be classified as *unknown*. Correct classification indicates that the document was classified properly (either as *business letter*, *memorandum* or *unknown*, in the case of the "other" data). Undetermined classifications are the number of business letters or memoranda that were classified as *unknown*. False classifications are the number of *business letters* that were classified as *memoranda* or vice versa, or unknowns (from the "other" data base) that were classified as either a *business letter* or a *memorandum*.

On a sample set of 40 images, the classifier took an average of 0.2 cpu seconds/document to complete (see performance comparison in Section 4.3).

4.2 Evaluation of Functional Decomposition

4.2.1 Scoring Methodology

Performance is measured in terms of *precision* and *recall*, standard measures in the field of Information Retrieval. These were computed with respect to what a knowledgeable human would achieve in carrying out the functional labeling task on sets of single-page, English-language business letters, more fully described below.

In our context *recall* becomes the ability of the system to find and label the functional components which are actually present, while *precision* measures how many of the labels the system applies are correct. In arithmetic terms *recall* for a given document is the ratio of correct labels to all actual labels and *precision* is the ratio of correct labels to labels output by the system.

Scoring is based on the goal of finding the above specified list of target components, not on labeling all blocks in the physical segmentation. The raw scores for the components are based on the following:

- Components are weighted according to their importance in a document class or their importance to the user. The default weight is 1.0 for all components at present.
- The degree and nature of the overlap between a labeled physical region and an actual functional component govern partial credit. In general a score for a labeled region is based on the relative area of the region to that of the actual component, where the actual component has been so specified by a knowledgeable human being.

  In particular, if a physical segment envelops two (or more) functional components, but only one is labeled, full credit is given for the component found; zero credit is given for the component(s) not found; no false positives are scored. Thus, *precision* is not affected in this situation.
- Note that a text component may be correctly labeled even though it is resistant to OCR. In this case full credit is given even though a database entry containing its ASCII contents would not be useful for later searching.

4.2.2 Results for Functional Decomposition

We trained our content-based functional labeling module on a set of 38 single-page, English-

Table 2: Recognition rates for classifier.

	Training Set				Test Set				
	letters	memoranda	other	subtotal	letters	memoranda	other	subtotal	total
Recognition rate	0.96	0.90	0.96	0.95	0.88	0.93	1.00	0.92	0.93
Undetermined	0.04	0.07	0.00	0.04	0.12	0.07	0.00	0.98	0.06
Incorrect	0.00	0.03	0.04	0.01	0.00	0.00	0.00	0.00	0.01

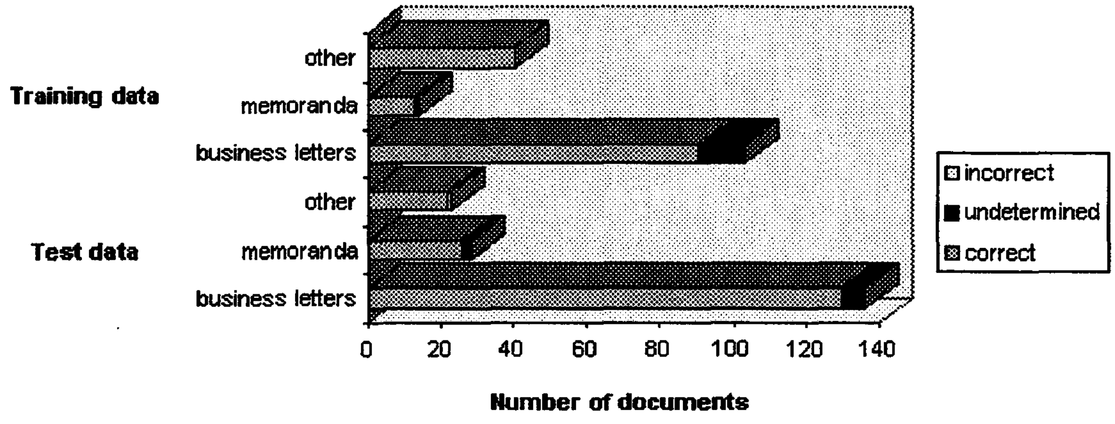


Figure 3: Training and test set classifier performance by class.

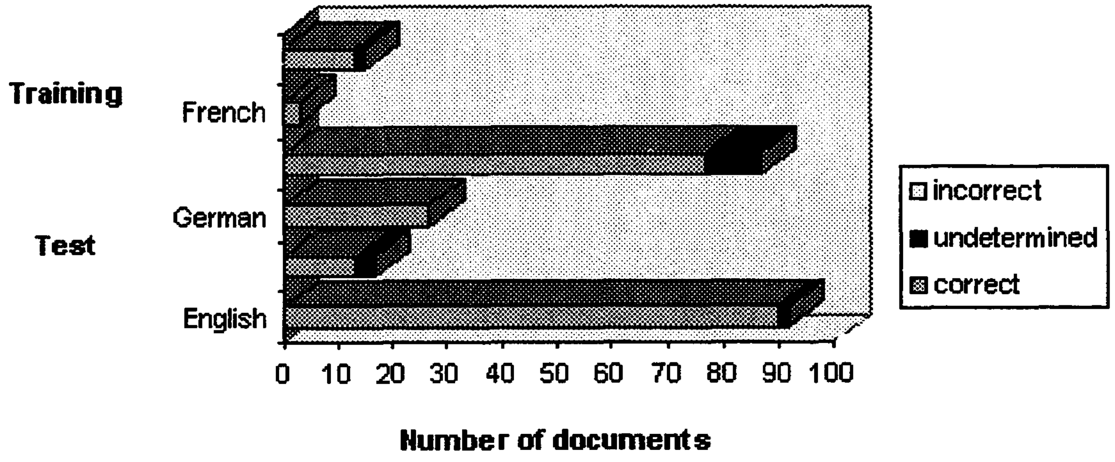


Figure 4: Classifier performance on business letter data base.

Table 3: Comparison of results for functional decomposition.

Description	Source	Approach	Images	Recall	Precision
Test Set A	Native/English	content	49	0.78	0.97
Test Set B	Nonnative/English	content	22	0.36	0.76
Test Set B	Native/English	geometric	22	0.52	0.77
Test Set C	NonNative/English	geometric	56	0.71	0.92

language business letters. For the most part these images, which can fairly be described as "clean," were scanned from the original documents at 400 DPI (dots per inch). Overall *recall* was 0.87 and *precision* was 1.00.

Training for the geometric-based approach took part on a set of about 90 images. These images were relatively "clean" single-page, English language business letters, although, as in our earlier work [9], the notion of business letter was interpreted rather informally, requiring our system to be fairly robust.

In all there were three blind test sets (A,B and C) of which only Test Set B was evaluated under both methods, making four test scenarios. A summary of the results for the four scenarios is given in Table 3. Blind testing for the content-based method was initially performed on the set of 49 letters from Test Set A. Again, we used only single-page, English-language documents in which "business letter" was interpreted rather loosely, for they represented a wide range relative to what might be found in a secretary's traditional style manual. The chart in Figure 5 shows *recall* and *precision* for each type of functional component and those values overall. Figure 6 depicts the raw data which underlie Figure 5. The graph in Figure 5 has a gap at the position of the *distribution* component because given the raw data shown in the table of Figure 6, we see that *recall* is zero and *precision* is undefined for that component. These results were also reported in [9].

Test Set B, containing 22 images, consisted principally of letters written to the U.S. Library of Congress from other countries. In some cases the image quality was poor. Also these letters tended to have more manual annotations than the training set or either of the other two test sets. In general they did not observe so well layout conventions one would expect from a writer whose first language was (U.S.) English. For comparison purposes, Test Set B was evaluated using both the content-based approach and the geometric-based approach.

For Test Set B the overall scores under the content-based approach were 0.36 for Recall and 0.76 for Precision. A graph of these statistics for each component is shown in Figure 7 and a table of the underlying raw scores is given in Figure 8. Under the geometric-based approach Test Set B scores were somewhat better at 0.52 for Recall and 0.77 for Precision. The corresponding graph and table for this scenario are Figure 9 and Figure 10, respectively.

The results were lower in both cases from those of Test Set A, largely because of the lack of consistency in layout conventions and keyword phrasing, the second which especially affected the content-based approach. Recall that the knowledge base for both approaches was geared to U.S. English. These scores indicate the potential usefulness of a hybrid approach which is discussed later.

Test Set C, comprised of 56 images, was very similar to the training set and produced higher scores. In Test Set C the overall scores for precision and recall were 0.92 and 0.71, respectively, as illustrated and detailed in Figure 11 and Figure 12. Precision remains high, so our discussion will focus on recall. As might be expected, the recall scores for components above the body were better than for those below. Because the size of the body varies from document to document, placement of components which occur below the body will not be as consistent as it is for those above. Given the content-free nature of this method, it is entirely dependent on the physical segmentation. Thus, a component like the *signature*, which is rarely encapsulated in its own block, is not likely to be found. In fact, frequently, the *closing*, *signature* and *sender* occur in the same physical segment. Along the same lines, if the *date* gets merged into the *letterhead_top*, it will not be separately identified. In several cases the *letterhead_bot* was not in any physical segment, so it could not be found.

4.3 System Performance

Figure 13 shows the relative speed of the different components of the IDUS system. There is a 2:1 difference in the processing speeds for the two functional decomposition methods. These times are separate from the actual OCR execution; the difference in speed is due to heavy reliance on the content-based approach to dictionary lookup. However, OCR, in general is a huge time sink. By using techniques that are not reliant on OCR and only using OCR when absolutely necessary, we can obtain a huge savings in processing time.

5 Conclusions

Future work includes modifying the classifier so that it is less reliant on content-based clues (OCR) and more on geometric clues. Our success using a non-ocr-based approach to functional decomposition leads us to believe that we will be able to adopt a non-OCR-reliant approach.

In this paper, we have discussed two different approaches for functional analysis. The content-based affords us the luxury of using textual clues to label functional components; however, it is at the expense of computational speed. The geometric based approach is much faster; however, we are unable to

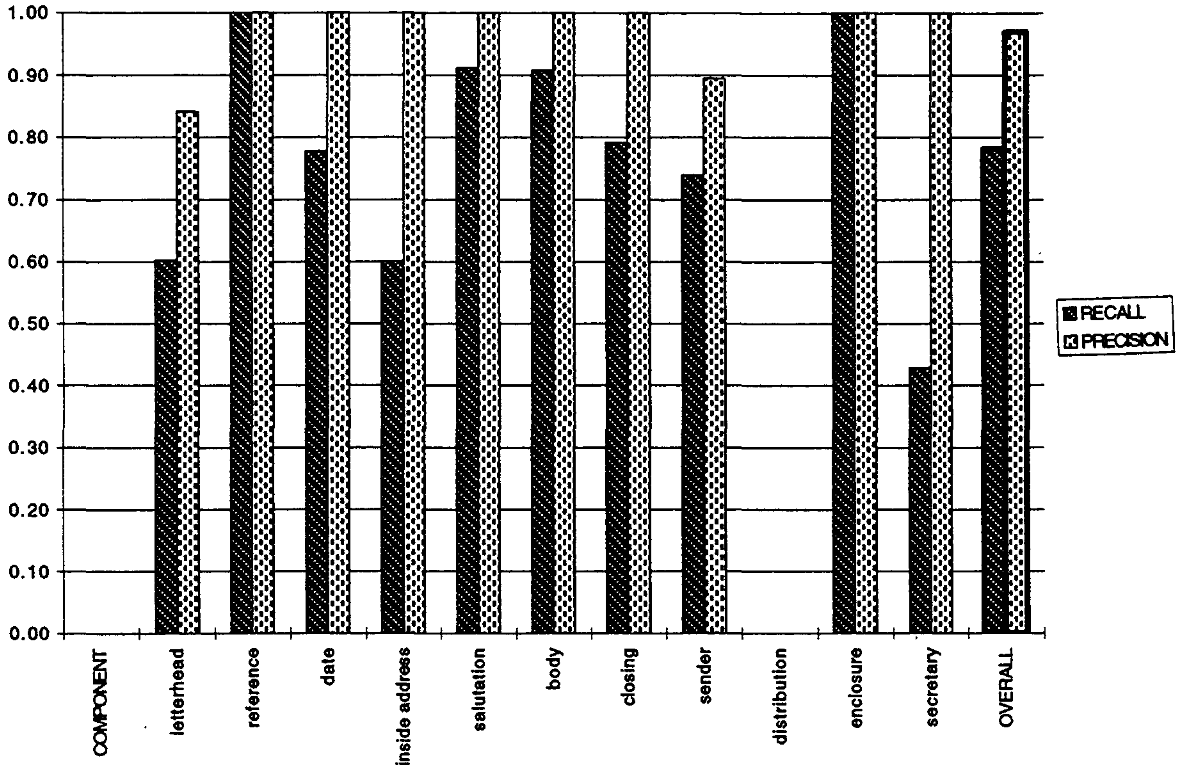


Figure 5: Test set labeling results by functional component for content-based functional analysis on data set (A).

COMPONENT	TRUE POSITIVE	FALSE POSITIVE	PRESENT
letterhead	26.50	5.00	44.00
reference	4.00	0.00	4.00
date	28.00	0.00	36.00
inside address	18.00	0.00	30.00
salutation	41.00	0.00	45.00
body	44.47	0.00	49.00
closing	38.00	0.00	48.00
sender	34.00	4.00	46.00
distribution	0.00	0.00	1.00
enclosure	8.00	0.00	8.00
secretary	6.00	0.00	14.00
TOTALS	**247.97**	**9.00**	**325.00**

Figure 6: Test set raw scores by functional component for content-based approach on data set (A).

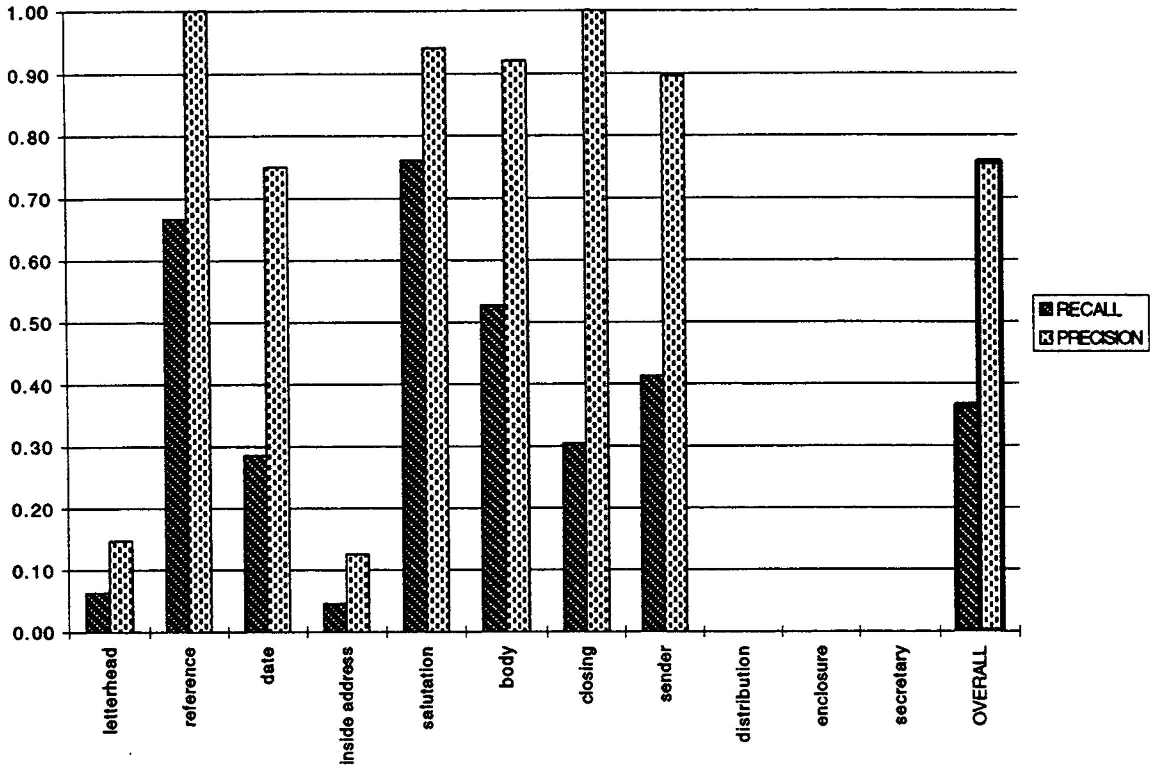


Figure 7: Test set labeling results by functional component for content-based functional analysis on data set (B).

COMPONENT	TRUE POSITIVE	FALSE POSITIVE	PRESENT
letterhead	1.38	8.00	22.00
reference	2.00	0.00	3.00
date	6.00	2.00	21.00
inside address	1.00	7.00	22.00
salutation	16.00	1.00	21.00
body	11.61	1.00	22.00
closing	7.00	0.00	23.00
sender	8.68	1.00	21.00
distribution	0.00	0.00	1.00
enclosure	0.00	0.00	1.00
secretary	0.00	0.00	0.00
TOTALS	**53.67**	**20.00**	**157.00**

Figure 8: Test set raw scores by functional component for content-based approach on data set (B).

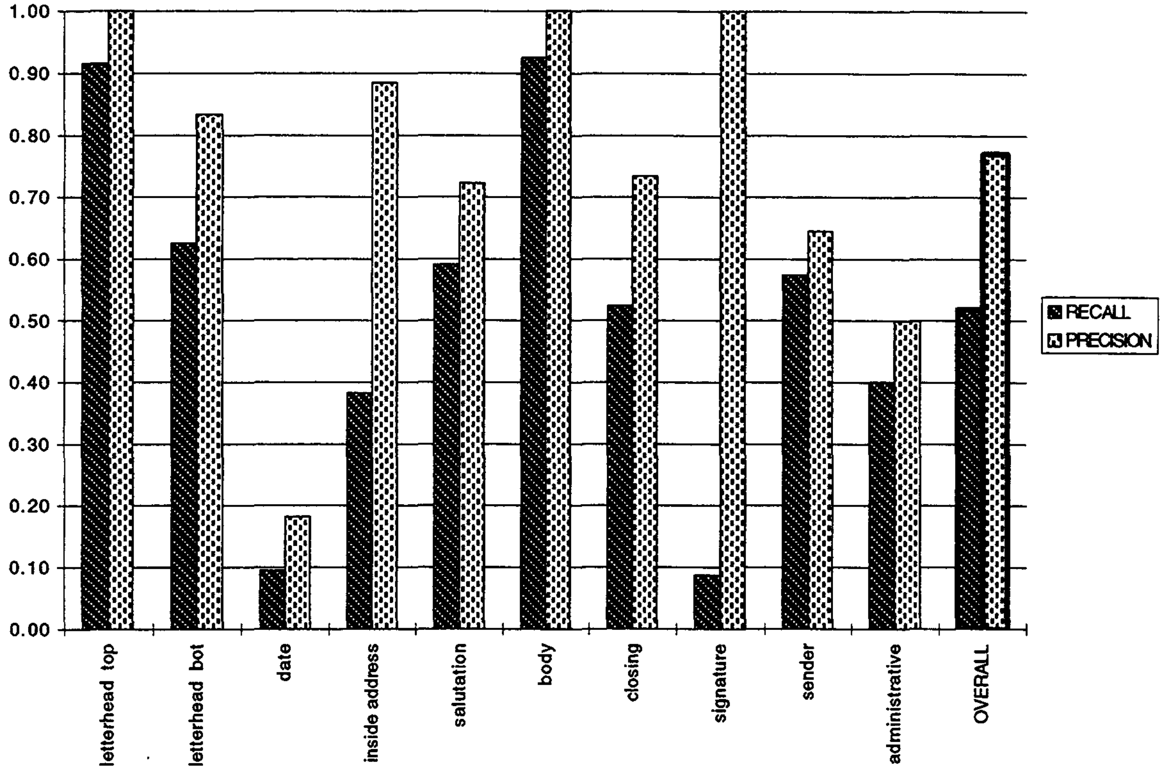


Figure 9: Test set labeling results by functional component for geometric-based functional analysis for data set (B).

COMPONENT	TRUE POSITIVE	FALSE POSITIVE	PRESENT
letterhead top	20.14	0.00	22.00
letterhead bot	5.00	1.00	8.00
date	2.00	9.00	21.00
inside address	7.65	1.00	20.00
salutation	13.00	5.00	22.00
body	20.34	0.00	22.00
closing	11.00	4.00	21.00
signature	1.83	0.00	21.00
sender	10.90	6.00	19.00
administrative	2.00	2.00	5.00
TOTALS	**93.86**	**28.00**	**181.00**

Figure 10: Test set raw scores by functional component for geometric-based approach on data set (B).

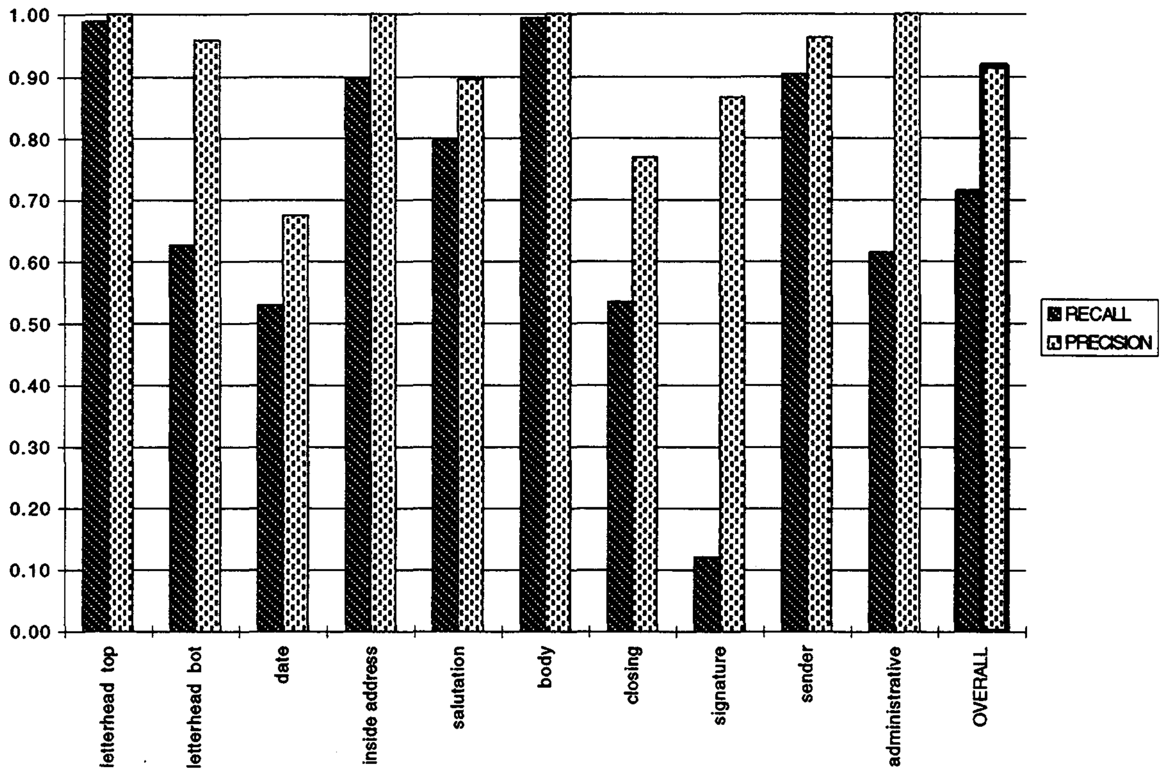


Figure 11: Test set labeling results by functional component for geometric-based functional analysis for data set (C).

COMPONENT	TRUE POSITIVE	FALSE POSITIVE	PRESENT
letterhead_top	55.36	0.00	56.00
letterhead_bo	22.60	1.00	36.00
date	27.00	13.00	51.00
inside address	44.87	0.00	50.00
salutation	43.00	5.00	54.00
body	55.63	0.00	56.00
closing	30.00	9.00	56.00
signature	6.50	1.00	54.00
sender	50.57	2.00	56.00
administrative	8.00	0.00	13.00
TOTALS	**343.53**	**31.00**	**482.00**

Figure 12: Test set raw scores by functional component for geometric-based approach on data set (C).

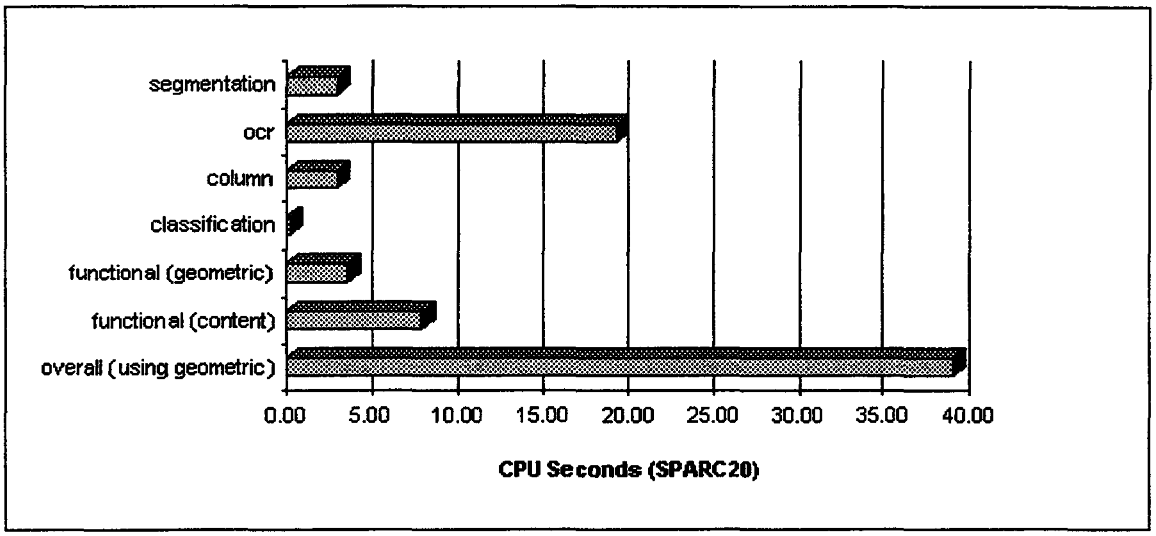


Figure 13: System component and overall performance.

split physical segments to find functional components occupying only a portion of them. To achieve really good performance what is really needed is a way to compensate for the physical segmentation via the judicious incorporation of OCR and content analysis. OCR should not be used indiscriminately, as it is computationally expensive. Rather it should be "surgically" applied as a verification tool or when a component is deemed important enough that it needs to be excised from the physical region in which it is embedded. We feel that this hybrid approach, combining geometric and content knowledge, would be extremely beneficial, recognizing that the key development issue would be the fashioning of a control structure which summons OCR only when it is appropriate.

Acknowledgements

This work was funded by the U.S. Department of Defense.

References

[1] T.A. Bayer. Understanding structured text documents by a model based document analysis system. In *Second International Conference on Document Analysis and Recognition*, pages 448–453, Tsukuba Science City, Japan, 20–22 October 1993. IEEECSP.

[2] Earl Cox. *The Fuzzy Systems Handbook.* AP Professional, Cambridge, MA, 1994.

[3] A. Dengel, R. Bleisinger, F. Fein, R. Hoch, F. Hones, and M. Malburg. OfficeMAID - a system for office mail analysis, interpretation and delivery. In *Proceedings of the International Association for Pattern Recognition Workshop on Document Analysis Systems*, pages 253–312, Kaiserslautern, Germany, 18–20 October 1994.

[4] Andreas Dengel. *ANASTASIL: A System for Low-Level and High-Level Geometric Analysis of Printed Documents.* Springer-Verlag, New York, NY, 1992.

[5] Hiroko Fujihara, Elmamoun Babiker, and Dick B. Simmons. Fuzzy approach to document recognition. In *Second IEEE International Conference on Fuzzy Systems*, pages 980–985, San Francisco, CA, March 28–April 1 1993.

[6] Hiroko Fujihara and Amit Mukerjee. Document recognition using qualitative reasoning. In *Symposium on Document Analysis and Information Retrieval*, pages 316–331, Las Vegas, NV, 16–18 March 1992. University of Nevada, Las Vegas.

[7] Joachim Kreich. Robust recognition of documents. In *Second International Conference on Document Analysis and Recognition*, pages 478–483, Tsukuba Science City, Japan, 20–22 October 1993. IEEECSP.

[8] Joachim Kreich, Achim Luhn, and Gerd Maderlechner. An experimental environment for model based document analysis. In *First International Conference on Document Analysis and Recognition (ICDAR 91)*, pages 50–58, Saint-Malo, France, 30 September – 2 October 1991. AFCET-IRISA/INRIA.

[9] Mark Lipshutz and Suzanne Liebowitz Taylor. Functional decomposition of business letters. In *Fourth Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas, Nevada, 24 – 26 April 1995.

[10] Suzanne Liebowitz Taylor, Deborah A. Dahl, Mark Lipshutz, Lewis M. Norton, Roslyn Weidner Nilson, and Marcia C. Linebarger. Integrating natural language understanding with document structure analysis. *Artificial Intelligence Review*, 8:163–276, December 1994.

[11] Suzanne Liebowitz Taylor, Mark Lipshutz, Deborah A. Dahl, and Carl Weir. An intelligent document understanding system. In *Second International Conference on Document Analysis and Recognition*, pages 107–110, Tsukuba City, Japan, October 1993. IEEE Computer Society Press (Los Alamitos, CA).

[12] Suzanne Liebowitz Taylor, Mark Lipshutz, and Roslyn Weidner NIlson. Classification and functional decomposition of business correspondence. In *Third International Conference on Document Analysis and Recognition*, pages 563–566, Montreal, Canada, 14-16 August 1995.

[13] Suzanne Liebowitz Taylor, Mark Lipshutz, and Carl Weir. Document structure interpretation by integrating multiple knowledge sources. In *Symposium on Document Analysis and Information Retrieval*, pages 58–76, Las Vegas, Nevada, 16–18 March 1992. University of Nevada, Las Vegas.

A Complete Environment for Ground-Truthing and Benchmarking Page Segmentation Algorithms

Luc Vincent **Berrin Yanikoglu**
Xerox Desktop Document Systems
9 Centennial Drive, Peabody MA 01960, USA
lucv@xis.xerox.com berrin@xis.xerox.com

Abstract

We describe a new approach for the automatic and objective evaluation of page segmentation (zoning) algorithms. Unlike techniques that rely on OCR output, our method is region-based: the segmentation output, described as a set of regions together with their types, ordering relationships, child-parent relationships, etc, is compared to a pre-stored set of ground-truth regions, and an assessment of segmentation quality is derived. In order for this comparison not to be fooled by slight variations in region shapes, it is achieved using region maps in which ON-pixels are painted according to the region(s) they belong to. From the "ground-truth map" and the "segmentation map", error maps are derived, in which segmentation mistakes and their locations are encoded. Error maps enable an accurate assessment of segmentation quality regardless of the number and intricacy of the regions on the page. Misclassifications, splittings and merging of various types, are among the zoning mistakes that are derived from these error maps.

Our segmentation benchmarking system can be finely tailored to a user's needs: region types, subtypes, attributes, etc, can be customized, the weights of errors of different types are user-defined, and several method are provided for combining errors into a global quality measure. This approach is also of great interest for benchmarking specific aspects of segmentation, such as headline detection, text/image separation, reverse-video text detection, etc.

The ground-truth data files needed by the system can be easily generated and updated using GroundsKeeper, *an X-window based tool we developed.* GroundsKeeper *allows one to view a document image, manually draw regions (rectangular or arbitrarily shaped) on top of it, and specify information about each region (type, attributes, etc). Just like the benchmarking system itself,* GroundsKeeper *is completely customizable: region types, sub-types, attributes, etc, are specified by the user in a startup file.*

1 Benchmarking Page Segmentation: Introduction

Page segmentation is the process by which a document page image is decomposed into its structural and logical units, such as images, rulings, paragraphs, headlines, tables, line-art regions, etc. This process, often refered to also as "zoning", "layout analysis", or "page decomposition", is critical for a variety of document image analysis applications. First and most importantly, commercial OCR packages need page segmentation to understand the layout of the page, read the text in the correct order (without attempting to read the halftones and graphics), and ultimately be able to *recompose* the document in a word processing environment. Similarly, reading machines for the blind need accurate page segmentation to be able to deal with multicolumn documents and read the text in an adequate order. Even digital copiers and related software/hardware systems rely on document segmentation techniques to correctly identify the various types of regions present on a page (e.g., text, continous tone images, halftones, line-art, ...), and render them appropriately [8].

A very large number of page segmentation algorithms have been developed in the past couple of decades, some described in literature [9] and some prorietary [20]. These algorithms use many different aproaches to the problem, some of which include: rule-based systems [4], use of connected component bounding-boxes [5], use of background and "white streams" information [12, 2, 1], etc. Some of these algorithms are specifically designed to work on particular types of document, whereas others are meant to be completely general. Furthermore, some algorithms are designed for very specific sub-tasks of what is generally refered to as page segmentation: for example, some algorithms may simply detect the graphics in a document page image, others may only care about the text. Some algorithms generate coarse segmentations (e.g., at the galley level) while others decompose pages down to the paragraph level, or even down to the line level. For some applications, a page segmentation algorithm may need to distinguish between halftones and continuous tones, for others, the distinction is irrelevant.

Faced with such a diversity of methods and such a wide variety of goals, a question arises: how good is a particular segmentation algorithms at performing a particular type of page segmentation? How should we assess the quality of a given algorithm for a given task? The accuracy of a page segmentation system, is unfortunately very difficult to evaluate objectively. Typically, assessing the quality of a system involves running it on a large number of document images and "eyeballing" the results, a very tedious and subjective process.

For all these reasons, systems that make it easier to automatically and objectively evaluate the adequacy of a zoning algorithm for a given segmentation task are becoming more and more important. Obviously, a good segmentation benchmarking tool, would be very helpful in deciding between commercial systems for a given application. But beyond that, such a benchmarking environment would also enable developers of page segmentation systems to regularly test their algorithms against large databases of documents, thereby speeding up the process of testing new ideas and fine-tuning existing segmentation algorithms.

Two main approaches have been proposed in literature for benchmarking page segmentation. The first one, developed at UNLV, is purely text-based [17, 6, 7], whereas the approach we have chosen is region-based instead [15, 21]. These two approaches are briefly described below, and we explain why we believe our region-based approach to offer significant advantages over the UNLV system.

The rest of the paper is organized as follows: in Section 2, we describe our ground-truthing methodology, the tool we developed to easily create segmentation ground-truth files, as well as the file format used to represent the greound-truth information. In section 3, we describe the benchmarking itself: the use of region maps by the system is first described, then we deal with error identification and combination. Finally, we show how the system can be customized to virtually any user's need, and conclude.

1.1 Text-Based Approaches to the Evaluation of Page Segmentation Algorithms

The Information Science Research Institute (ISRI) of the University of Las Vegas (UNLV) has designed an interesting technique to evaluate the quality of page segmentation algorithms working within OCR software packages: schematically, this method works as follows:

1. Run page segmentation (including region ordering) *and* character recognition on a document page, and output the result as an ASCII string;
2. Calling c_i the cost of an elementary insertion operation in a string, and c_m the cost of a block move, use string matching algorithms derived from [19] to determine the number of insertions n_i and the number of block moves n_m that are needed to turn this output string into the ideal *ground-truth* string while minimizing the associated page cost $C = n_i \times c_i + n_m \times c_m$;
3. From this number, *subtract* the cost of errors that are purely due to OCR mistakes, as opposed to segmentation mistakes; this is done by running the OCR system on the same page, using *manual zoning* information, and subtracting the resulting error cost (expressed in numbers of insertions times cost c_i of an insertion) from the page cost C.

The big advantage of this technique is that it is purely text-based, and therefore does not require the page segmentation sub-system to specifically output zoning results in any particular format. In addition, although its underlying string-matching algorithms are rather elaborate, the overall approach is fairly straightforward: for one thing, ground-truth files for use with this system are very easy to create. Therefore, the UNLV zoning evaluation system has been relatively well accepted by the document recognition community.

Nonetheless, this system has a few severe limitations:

- Although this may not be true any longer, the UNLV segmentation quality metric only takes *insertions* and *block moves* into account, *deletions* are assumed to have zero cost. In practice, this means that an algorithm will not be penalized for finding extra text regions where there are none, or worse, for detecting images as text.
- With this method, only one zone order, or a small set of zone orders, are considered acceptable. If a zoning algorithm produces a perfect segmentation, but does not output regions in one of these acceptable orders, it will be penalized. This is sometimes unfair, since in many cases, region order is not uniquely defined (see Section 2.1). For example, how should such text regions as captions, headers, footers, or insets, be ordered? Clearly a range of options should be considered equally correct.
- Similarly, this approach requires running the OCR system twice: once with automatic segmentation and once with manual zoning, so that errors purely due to the OCR can be subtracted. The underlying assumption here is that OCR accuracy is unchanged regardless of scanning order. This is often not true in practice: a messy segmentation generally results in poorer

OCR quality (for example, if the segmentation splits or joins two galleys, hyphens are incorrectly paired, so the OCR engine can rely on fewer words to be lexically verified...). Therefore, this benchmarking approach tends to give lower segmentation scores than it would if the OCR engine was perfect.

- The method requires the zoning algorithm being benchmarked to be part of of an OCR system. Without OCR, there is no way to benchmark segmentation! Specifically, this means that the segmentation of documents in "exotic" languages (not supported by the OCR engine) cannot be benchmarked. More annoying, the segmentation of document images containing no text cannot be evaluated with this system.
- Along the same lines, the UNLV segmentation benchmarking system can only deal with text regions: the accuracy measure produced only reflect accuracy on text zones!
- Lastly, the output of the UNLV system is merely a set of numbers (number of insertions, block moves, and perhaps deletions). It therefore provides very little information on the types of mistakes that were actually made (Were regions split or merged? Were images mistaken for text? Were headline regions messed up more than regular text regions? etc). Such information would be very valuable to the segmentation algorithm developer.

1.2 Our Region-Based Approach to Automatic Segmentation Benchmarking

For all the reasons listed above, we started working on a region-based approach to page segmentation benchmarking. As will be shown in the rest of the paper, the system we developed is more complex and cumbersome to use than a text-based system, but offers much more flexibility, and avoids most of the shortcomings of the UNLV approach.

In our system, the output of the segmentation algorithm to be benchmarked is modeled as a set of regions, a region being a polygon (representing the outline of a page zone) together with such attributes as a type (text, image, etc) and a parent zone. This set of segmentation regions is matched against a pre-stored set of *ground-truth regions,* and such problems as erroneous region merging and splitting, misidentified region types, or extra "noise" regions are flagged.

A tool, *GroundsKeeper,* was developed to help the creation of ground-truth files. This tool, described in Section 2.2, allows the user to bring up a document image, draw a set of regions (enclosing image features such as halftones, paragraphs, captions, etc), specify ordering and "attachement" relationships between regions, and save the result in a simple ASCII format called *RDIFF (Region Description Information File Format).* This format is described in Section. 2.1. As will be emphasized later, *GroundsKeeper* can be customized to a user's needs.

During the actual benchmarking, the set of ground-truth region represented in the RDIFF file is matched against a set of *segmentation regions.* This region matching step is actually performed using *sets of pixels* instead of polygonal representations. In other words, each region is modeled as the set of ON-pixels its associated polygon contains: this way, the shape of region polygons is not taken into account, only their ON-pixel contents matters. Errors can therefore be accurately flagged, and insignificant differences between ground-truth and segmentation polygons are automatically ignored. This overall scheme is efficiently implemented through the use of *region maps* and *error maps,* as described in Section 3.

As we stress later, both the ground-truthing tool (GroundsKeeper) and the benchmarking tool (currently called *cluzo*) are completely customizable. At ground-truthing time, region types, subtypes, and attributes are user-defined. Additionally, in the benchmarking, some region types can be ignored while some can be made equivalent to other (example: ignore all `halftone` regions, and make `caption` type equivalent to `regular-text`), the types of errors that are important can be specified by the user, together with the error weighting method. All this information is stored in `init` files used by the ground-truthing and benchmarking tools.

2 Creation of Ground-Truth Files

2.1 Representation of Ground-Truth Information

To represent ground-truth information as well as segmentation results, the RDIFF format is used. For simplicity, this is an ASCII format, containing some header information such as the name of the associated document image, the resolution, the page size, and then a list of regions. Each region has an associated type and sub-type, may refer to a "parent" region, and may have any number of attributes. Region shape itself is encoded as a simple polygon (list of coordinates).

Providing the complete specifications of the RDIFF format would be beyond the scope of this paper. Let us however point out a few interesting features of this format:

- RDIFF is a Tag-Value based format, meaning that each entry in the format starts with a tag, the rest of the entry being the value of this tag. The advantage of this approach is its flexibility: an RDIFF reader typically only knows about the tags that are relevant to its purpose, and ignores the rest. In addition, even as new tags are added with each new release of the format, the new RDIFF files remain backwards compatible with previouly written RDIFF readers.

- Region shape is currently encoded using a polygon. In most applications, this is sufficient. However, if it became important to support region with holes, or disconnected regions, the RDIFF format could easily be expanded. New region representations, using for example lists of polygons, or run-length encoding, could be added, and a new tag would be used to specify the representation scheme chosen for each region (the default scheme being the current one: single polygon).
- Note that there are no restrictions on region location: regions can overlap, be included into one another, etc.

An example of RDIFF file is given below. Fig. 3 shows the *GroundsKeeper* window with the corresponding set of regions overlayed on the document image they were created from.

```
BEGIN_IMAGE
FILENAME /data/tif/h/halftone21.tif
IMAGE_WIDTH 2560
IMAGE_HEIGHT 3300
IMAGE_XRES 300
IMAGE_YRES 300
END_IMAGE

BEGIN_SUMMARY
INIT_FILE_VERSION 1.8 1995/04/18
TOTAL_REGIONS 25
TEXT_REGIONS 25
IMAGE_REGIONS 0
V_RULE_REGIONS 0
H_RULE_REGIONS 0
ORDER_FLAGS ANY
END_SUMMARY

BEGIN_REGIONS

R_TYPE IMAGE
R_SUBTYPE HALFTONE
R_NUMBER 1
R_ATTACHMENT 0
R_PARENT 0
R_DRAW_TYPE Box
R_NAME Zone1
R_ATTRIBUTES SHADED_BACKGROUND
REGION_POLYGON 301 692 71 681

... ...

R_TYPE TEXT
R_SUBTYPE CAPTION
R_NUMBER 9
R_ATTACHMENT 1
R_PARENT 0
R_DRAW_TYPE Box
R_NAME Zone9
REGION_POLYGON 722 838 78 266

R_TYPE TEXT
R_SUBTYPE CAPTION
R_NUMBER 10
R_ATTACHMENT 2
R_PARENT 0
R_DRAW_TYPE Box
R_NAME Zone10
REGION_POLYGON 1510 1584 82 348

... ...

R_TYPE IMAGE
R_SUBTYPE HRULE
R_NUMBER 15
R_ATTACHMENT 0
R_PARENT 0
R_DRAW_TYPE Box
R_NAME Zone15
REGION_POLYGON 136 150 0 1920

R_TYPE IMAGE
R_SUBTYPE VRULE
R_NUMBER 16
R_ATTACHMENT 0
R_PARENT 0
R_DRAW_TYPE Box
R_NAME Zone16
REGION_POLYGON 306 3034 1142 1156

...

R_TYPE TEXT
R_SUBTYPE REG_TEXT
R_NUMBER 18
R_ATTACHMENT 0
R_PARENT 0
R_DRAW_TYPE Box
R_NAME Zone18
REGION_POLYGON 304 426 722 1110

R_TYPE TEXT
R_SUBTYPE REG_TEXT
R_NUMBER 19
R_ATTACHMENT 0
R_PARENT 0
R_DRAW_TYPE Constrained-Polygon
R_NAME Zone19
REGION_POLYGON 428 1086 376 1116
COUNT 14
774 428
982 428
982 464
1116 464
1116 1048
954 1048
954 1086
718 1086
718 1048
376 1048
376 714
722 714
722 466
```

```
774 466
... ...
R_ORDER_RULE 18 < 19 < 20 < 21 < 22 < 23 < 24 < 25
```

Note the R_ORDER_RULE at the end of this file. This represents a set of ordering constraints between regions. Indeed, as mentioned in Section 1.1, we firmly believe that one complete region order is inadequate. Instead, a number of ordering constraints (or rules) should be used. For example, ordering constraings for regular text regions should be separate from ordering constrains among insets or side-bars. The latter regions are indeed not part of the main flow of text. Similarly, having several ordering constraints makes it possible to deal with ambigous cases such as the one shown in Fig. 1.

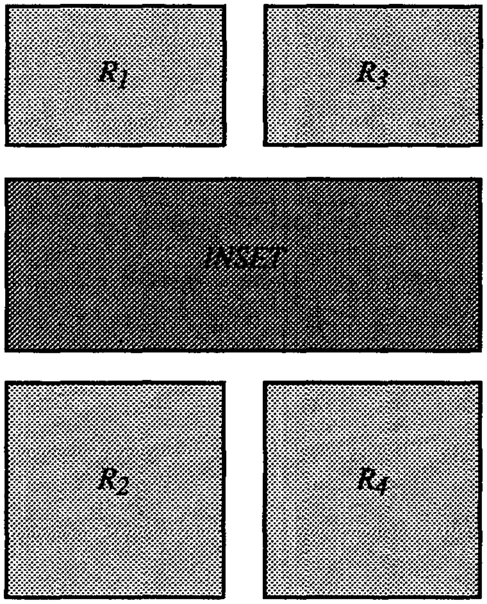


Figure 1: Ambiguous case for region ordering. The correct order between text regions R_1 through R_4 could be R_1, R_2, R_3, R_4 or R_1, R_3, R_2, R_4.

Accordingly, in the RDIFF format, any number of partial ordering rules can be specified using the R_ORDER_RULE tag. An example where several of these tags would be needed is shown in Fig. 2. Note that ordering constraints are not always the correct solution. Sometimes, *attachements* should be used instead. For example, a caption is "attached" to a picture, it does not come before or after this picture. Attachements can also be specified in the RDIFF language.

To conclude this section on RDIFF, let us point out that region ordering relationships are often considered as not being part of page segmentation itself. In fact, the R_ORDER_RULE tags are currently used only by a very small subpart of our benchmarking system. We made them part of our ground-truth format mainly in an effort to be complete.

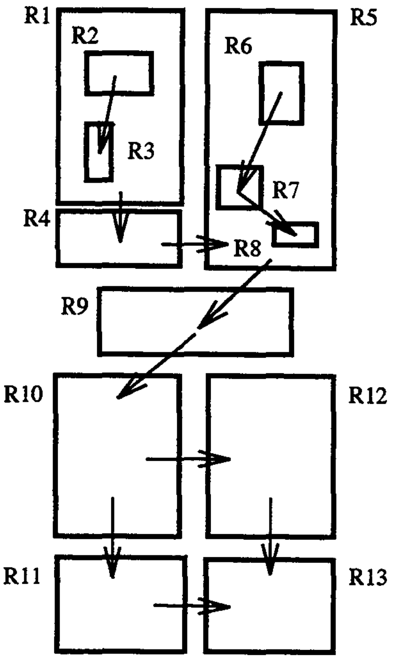


Figure 2: The order between regions in this "page" can be specified using four partial orders: $R_1 < R_4 < R_5 < R_9 < R_{10} < R_{11} < R_{13}$, $R_2 < R_3$, $R_6 < R_7 < R_8$, and $R_{10} < R_{12} < R_{13}$.

2.2 X-Window Based Tool to Assist Ground-Truth Creation: GroundsKeeper

To easily create test suites of segmentation ground-truth files in the RDIFF format, we developed an X-Window based tool called *GroundsKeeper* (see Fig. 3). For convenience, this tool provides three different region drawing methods: rectangles, that are sufficient for most cases, constrained polygons, i.e., polygons whose edges are bound to be vertical or horizontal, and arbitrary polygons, that are useful for complex layouts, or skewed pages. After their creation, regions can be deleted, moved, or modified at will. Previously created files can also be read in and updated. "Sequences" (i.e. region ordering rules) can be specified by simply clicking on regions one after the other. The functionality described in this paragraph is available under the "Zone" menu shown in Fig. 4.

Different display modes are also available. By default, the image is shown, with the regions and the sequences drawn over it, but one can also choose to display only the regions, or only the image (see "Display" menu in Fig. 5. Sequences, normally shown as a succession of arrows, can also be ommitted from the display if desired. More importantly, different

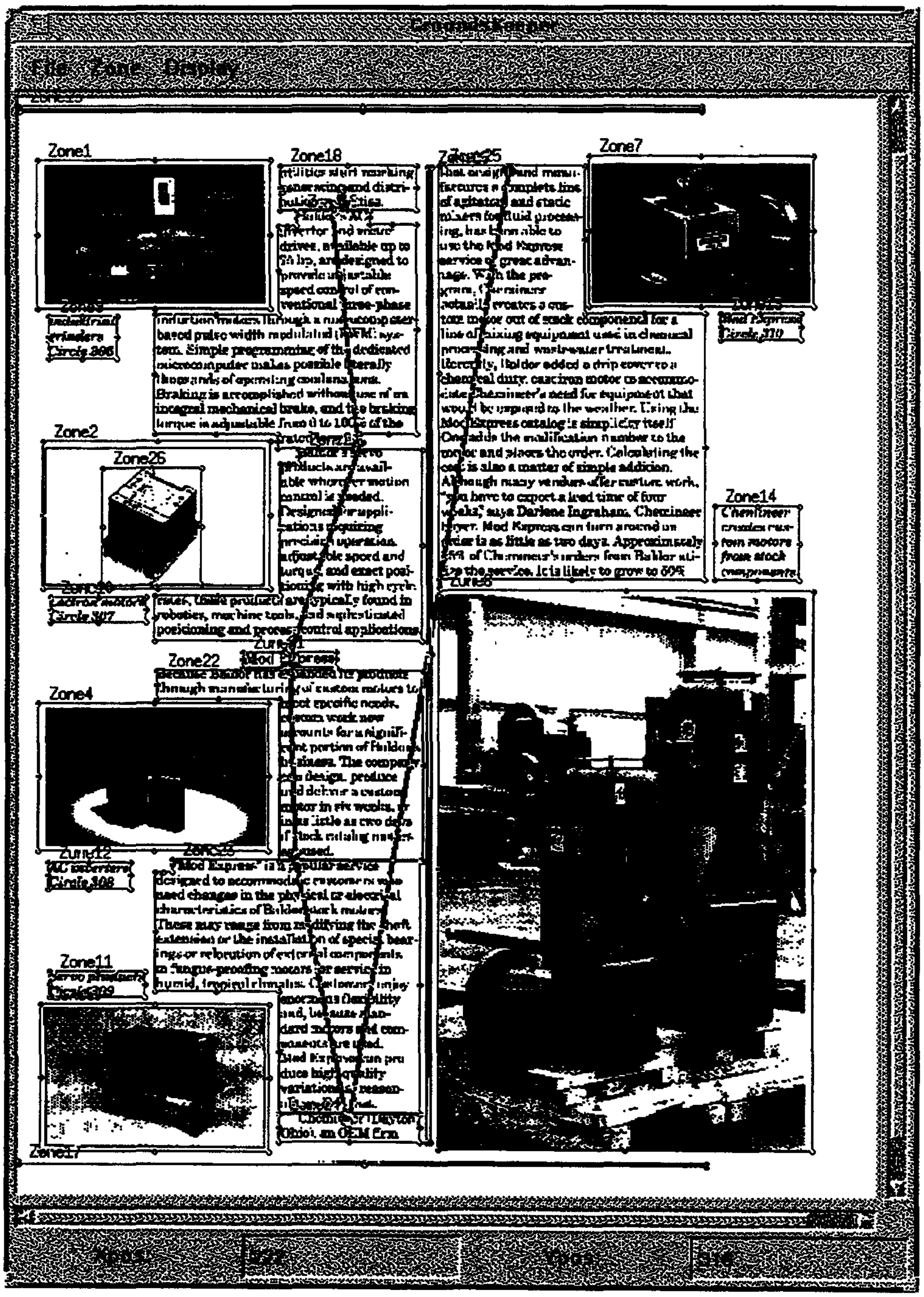


Figure 3: *GroundsKeeper* window showing document image `halftone21.tif` together with the complete set of ground-truth regions created. Ordering constraints between regions are shown as arrows (currently selected).

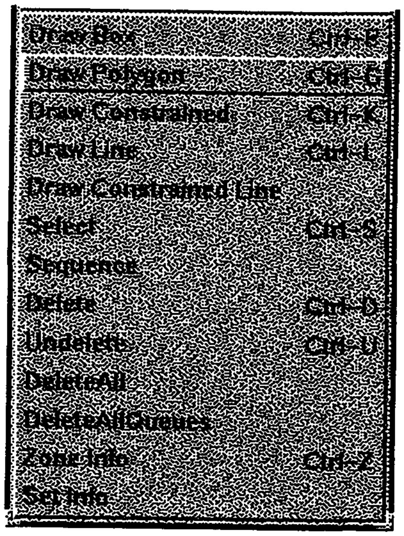


Figure 4: *GroundsKeeper*'s "Zone" menu.

zoom levels are available. In fact, a good way to use *Groundskeeper* is to first draw the regions coarsely at low-resolution, then zoom-in and adjust the polygons created.

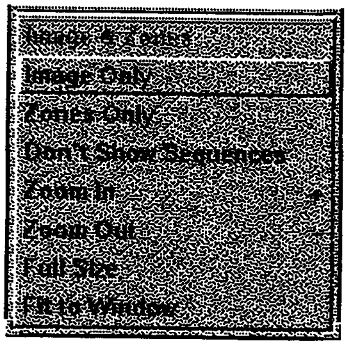


Figure 5: *GroundsKeeper*'s "Display" menu.

The polygons created are given unique identifying numbers, and can be tagged with their type, their sub-type, plus any number of ***attributes.*** Types, subtypes, and attributes are defined in an `init` file that *GroundsKeeper* reads at startup: they are therefore completely customizable for any particular application. In house, we use the `init` file shown in Fig. 6. Note that we chose to create very detailed ground-truth files: indeed, one can always choose to use less information that these files contain, but if the files were not detailed enough for a given application, they would be much less useful.

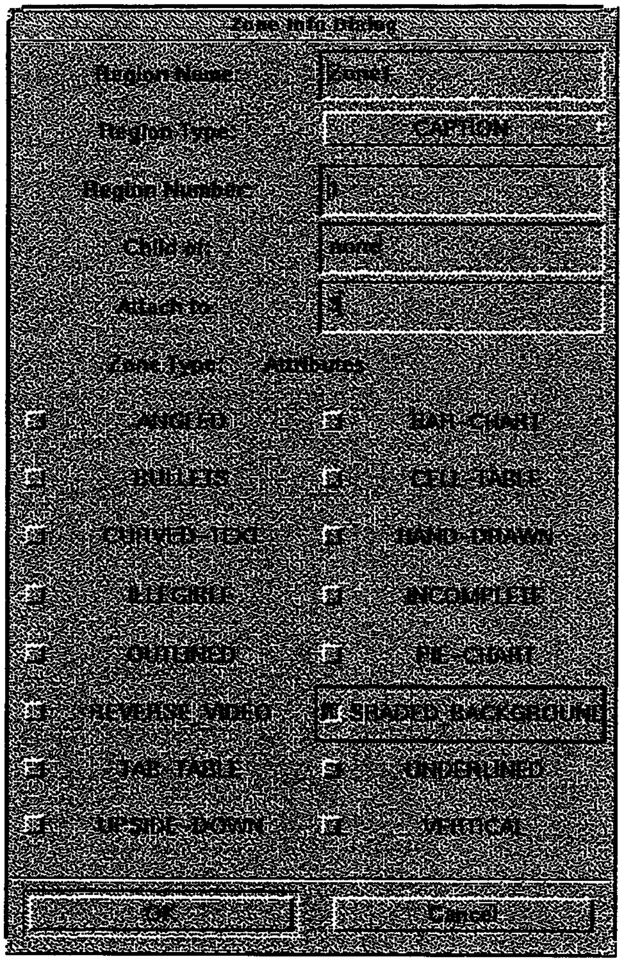
Groundskeeper has a number of additional features designed to ease and speed-up the creation of RDIFF files. Depending on the complexity of the input document, it takes between 2 and 30 minutes to create a complete detailed ground-truth file. We have created half a dozen test suites of ground-truthed

```
ATTRIBUTE_NAME ANGLED
ATTRIBUTE_NAME BAR-CHART
ATTRIBUTE_NAME BULLETS
ATTRIBUTE_NAME CELL-TABLE
ATTRIBUTE_NAME CURVED-TEXT
ATTRIBUTE_NAME HAND-DRAWN
ATTRIBUTE_NAME ILLEGIBLE
ATTRIBUTE_NAME INCOMPLETE
ATTRIBUTE_NAME OUTLINED
ATTRIBUTE_NAME PIE-CHART
ATTRIBUTE_NAME REVERSE_VIDEO
ATTRIBUTE_NAME SHADED_BACKGROUND
ATTRIBUTE_NAME TAB-TABLE
ATTRIBUTE_NAME UNDERLINED
ATTRIBUTE_NAME UPSIDE-DOWN
ATTRIBUTE_NAME VERTICAL

IMAGE_TYPE_NAME CAPTION
IMAGE_TYPE_NAME CAPTION_or_HEADLINE
IMAGE_TYPE_NAME CAPTION_or_REG_TEXT
IMAGE_TYPE_NAME CELL
IMAGE_TYPE_NAME DROPCAP
IMAGE_TYPE_NAME FOOTER
IMAGE_TYPE_NAME FOOTNOTE
IMAGE_TYPE_NAME FRAMEBOX
IMAGE_TYPE_NAME GRAPHICS
IMAGE_TYPE_NAME GRID
IMAGE_TYPE_NAME HALFTONE
IMAGE_TYPE_NAME HEADER
IMAGE_TYPE_NAME HEADER_or_HEADLINE
IMAGE_TYPE_NAME HEADLINE
IMAGE_TYPE_NAME HEADLINE_or_REG_TEXT
IMAGE_TYPE_NAME HRULE
IMAGE_TYPE_NAME INSET
IMAGE_TYPE_NAME LOGO
IMAGE_TYPE_NAME RAISEDCAP
IMAGE_TYPE_NAME REG_TEXT
IMAGE_TYPE_NAME SIDEBAR
IMAGE_TYPE_NAME SUBTITLE
IMAGE_TYPE_NAME TABLE
IMAGE_TYPE_NAME TABLE_COLUMN
IMAGE_TYPE_NAME TABLE_LINE
IMAGE_TYPE_NAME TIMESTAMP
IMAGE_TYPE_NAME UNKNOWN_TYPE
IMAGE_TYPE_NAME VRULE
```

Figure 6: Example of *GroundsKeeper* init file.

documents (magazines, newspapers, business letters, etc), for a total of about 500 RDIFF files. Currently, these suites are primarily used for testing and improving the segmentation benchmarking algorithms in *cluzo*, but progressively, we will use them more and more for benchmarking in-house and external segmentation algorithms.



Zone Info Dialog

Region Name: Zone1

Region Type: CAPTION

Region Number: 0

Child of: none

Attach to:

Zone Type/ Attributes

□ ANGLED □ BAR-CHART

□ BULLETS □ CELL-TABLE

□ CURVED-TEXT □ HAND-DRAWN

□ ILLEGIBLE □ INCOMPLETE

□ OUTLINED □ PIE-CHART

□ REVERSE VIDEO ☒ SHADED BACKGROUND

□ TAB-TABLE □ UNDERLINED

□ UPSIDE-DOWN □ VERTICAL

Figure 7: *GroundsKeeper*'s "Zone Info" dialog box. The attributes shown correspond to the list present in the `init` file of Fig. 6.

3 Segmentation Benchmarking Algorithms

3.1 Use of Region Maps

One of the key features in our region-based approach to benchmarking segmentation is that regions are *not* compared based on their shape, but based on their ON-pixel contents. Two regions that enclose the same image features (text, image, etc) should indeed be considered as identical regardless of how tightly they enclose these features.

In order to efficiently deal with regions as sets of ON-pixels, the first step in our system is to create *region maps.* A region map is created for the ground-truth regions, and another one for the "segmentation regions". These two maps are called ground-truth map and segmentation map respectively. Each one is a low-resolution representation of the original document image, in which each ON-pixel is tagged according to the region(s) it belongs to.

3.2 Region Matching and Identification of Errors

The region maps are used for identifying the following types of segmentation errors:

- missed region
- extra region
- erroneous region type
- vertical split
- horizontal split
- vertical merge
- horizontal merge

To this end, both maps are scanned concurrently in order to detect which segmentation regions intersect with each ground-truth region, and which ground-truth regions intersect with each segmentation region.

One-to-one matches are first considered: if the region types involved are identical (or equivalent for the purpose of the segmentation being benchmarked), then they correspond to zones that have been correctly segmented. Otherwise, mislabeling errors are identified.

All other cases correspond to segmentation errors. For example, if a ground-truth region does not match with any segmentation region (i.e. does not intersect with any segmentation region), it means that the corresponding page object has been missed by the segmentation algorithm.

Take now the case of a segmentation region S that matches with several ground-truth regions G_1, G_2,..., G_n. This means that regions have been erroneously merged. In this case, depending on region types, and depending on the type of error analysis desired, it may be useful to detect whether the merge was *vertical* or *horizontal*. For example, if the segmentation is to be used in the context of an OCR system, vertical merging of text regions is usually not a big deal, especially if the regions are consecutive within the same ordering rule. On the other hand, horizontal merging of text regions should be strongly penalized. To detect whether a merging is horizontal, we scan the pixels of the segmentation region S and see for each scanline if the corresponding ON-pixels in the ground-truth map belong to several regions. When this is the case, the entire scanline corresponds to an erroneous merge and is flagged as such. With this method, we are able to only flag those pixels that are actually affected by the horizontal merge. This is illustrated by Fig. 8. The same analysis can be done for vertical merging as well if needed.

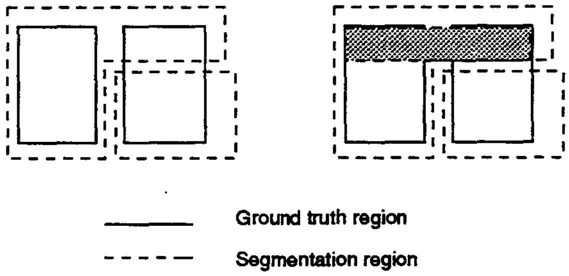


Figure 8: When benchmarking the segmentation of an OCR system, only the pixels in the shaded area will be found to be part of an erroneous horizontal merge of text regions.

Similarly, when a ground-truth region is found to match several segmentation regions, this is a sure sign that this region has been erroneously split by segmentation. The same kind of analysis as described in the previous paragraph can be done in order to decide whether the split was vertical or horizontal. For more details on how all the different cases are handled, refer to [21].

3.3 Weighting and Combination of Errors

When a segmentation error is detected, how should it be weighted? How should the seriousness of this error be assessed? How should errors be combined if they affect the same ground-truth region? For example, if (part of) a ground-truth region is found to have been split twice by segmentation, should this be considered twice as bad? Similarly, if this region is found to have been both merged and split, should one of these error types have precedence over the other one, or should both error be charged?...The answers to all those questions depend on the type of segmentation being benchmarked, and should be specified in the `init` file used by our benchmarking tool (see next section).

To deal with all the possible ways one may want to weight amd combine errors, we use a scheme based on *error maps*. Each time an error is detected, the ON-pixels involved are flagged in this error map. Whenever a pixel is found to be involved in several errors (for example, an erroneous split and an erroneous merge), we can decide to flag this pixel as being involved in all these errors, or only in the most serious one (the seriousness of each error type is specified in the `init` file).

Eventually, once all the errors have been detected, the resulting error map is analyzed and the segmentation quality is then characterized by a few numbers, such as: percentage of mislabeled pixels, percentage of pixels involved in horizontal merges, etc. The set of numbers used to characterize segmentation accuracy is derived from the `init` file used by the system.

As an example, consider the set of ground-truth regions shown in Fig. 9: 8 text regions and 2 image regions are present. The double-spaced text was zoned into 5 different paragraphs, according to our ground-truthing guidelines, but for most applications, zoning it into 2 galleys would have been sufficient. The automatic segmentation algorithm we used here did not see the two different galleys present, and made a mess of this page, as shown in Fig. 10. The corresponding error map created by our benchmarking algorithm is shown in Fig. 11. Note that in this map, only the pixels where horizontal merging occured are flagged (black pixels in the map). Indeed, the `init` file we used specified that vertical merge/split of text should not be penalized.

The output of the benchmarking tool for this example is given below. Note that for "historical" reasons we are still distinguishing between labelling errors and all other error types. The `init` file used to produce this output specified 6 different types of mislabelling, but only one type of vertical merge, horizontal merge, etc. Also recall that the cost of vertical merges and splits was set to 0.

```
Groundtruth file:          bhng11.sgt
Segmentation file:         bhng11.rdiff
Image file:                bhng11.tif

ZONING:

Missed regions              =     0.00 %
Noise regions               =     0.00 %
Horizontal merges           =    49.22 %
Horizontal splits           =     0.00 %
```

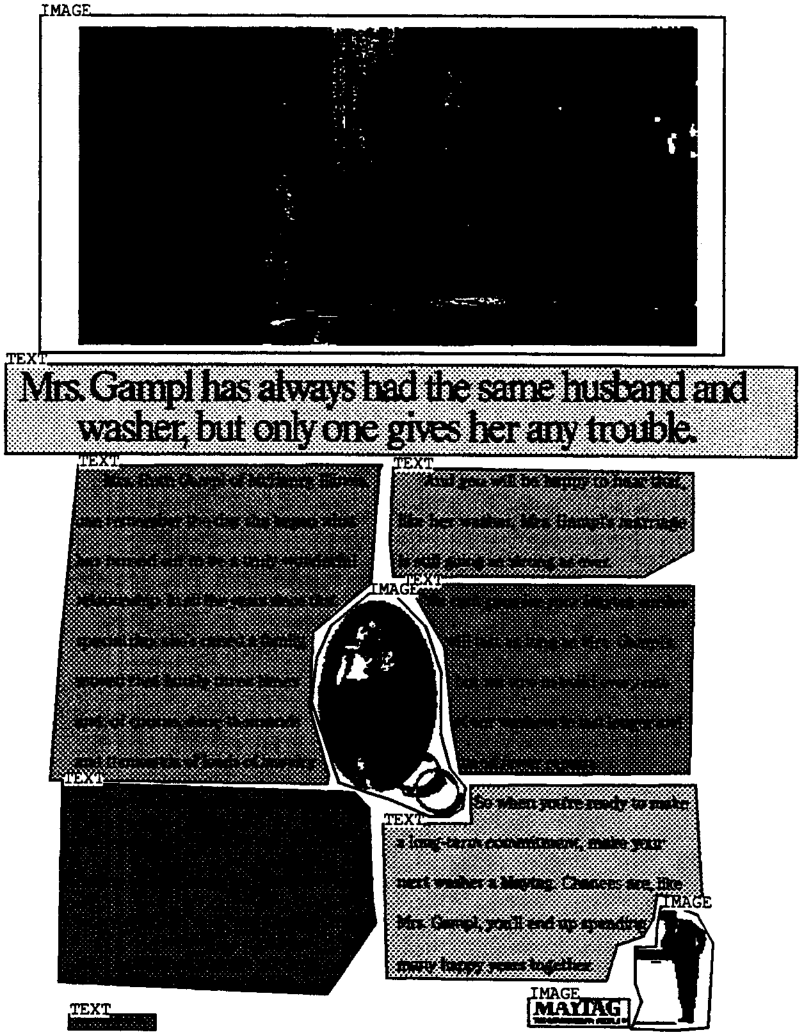
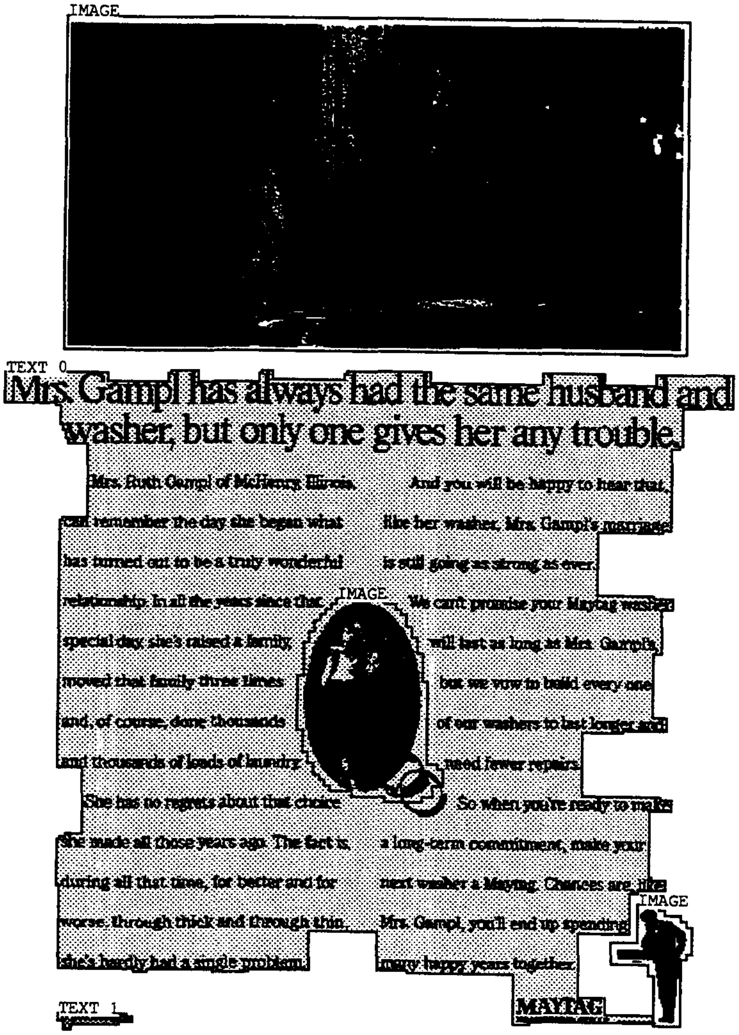


IMAGE
TEXT
Mrs. Gampl has always had the same husband and washer, but only one gives her any trouble.
TEXT
TEXT
And you will be happy to hear that, like her washer, Mrs. Gampl's marriage is still going as strong as ever.
IMAGE
TEXT
TEXT
TEXT
So when you're ready to make a long-term commitment, make your next washer a Maytag. Chances are, like Mrs. Gampl, you'll end up spending many happy years together.
IMAGE
IMAGE
MAYTAG
TEXT

Figure 9: Display of the ground-truth regions for a document image.



Mrs. Gampl has always had the same husband and washer, but only one gives her any trouble.

Mrs. Ruth Gampl of McHenry, Illinois, can remember the day she began what has turned out to be a truly wonderful relationship. In all the years since that special day, she's raised a family, moved that family three times and, of course, done thousands and thousands of loads of laundry.

She has no regrets about that choice she made all those years ago. The fact is, during all that time, for better and for worse, through thick and through thin, she's hardly had a single problem.

And you will be happy to hear that, like her washer, Mrs. Gampl's marriage is still going as strong as ever.

We can't promise your Maytag washer will last as long as Mrs. Gampl's, but we vow to build every one of our washers to last longer and need fewer repairs.

So when you're ready to make a long-term commitment, make your next washer a Maytag. Chances are, like Mrs. Gampl, you'll end up spending many happy years together.

MAYTAG

Figure 10: A segmentation result for the same image as in Fig. 9. Note the horizontal merging of text galleys.

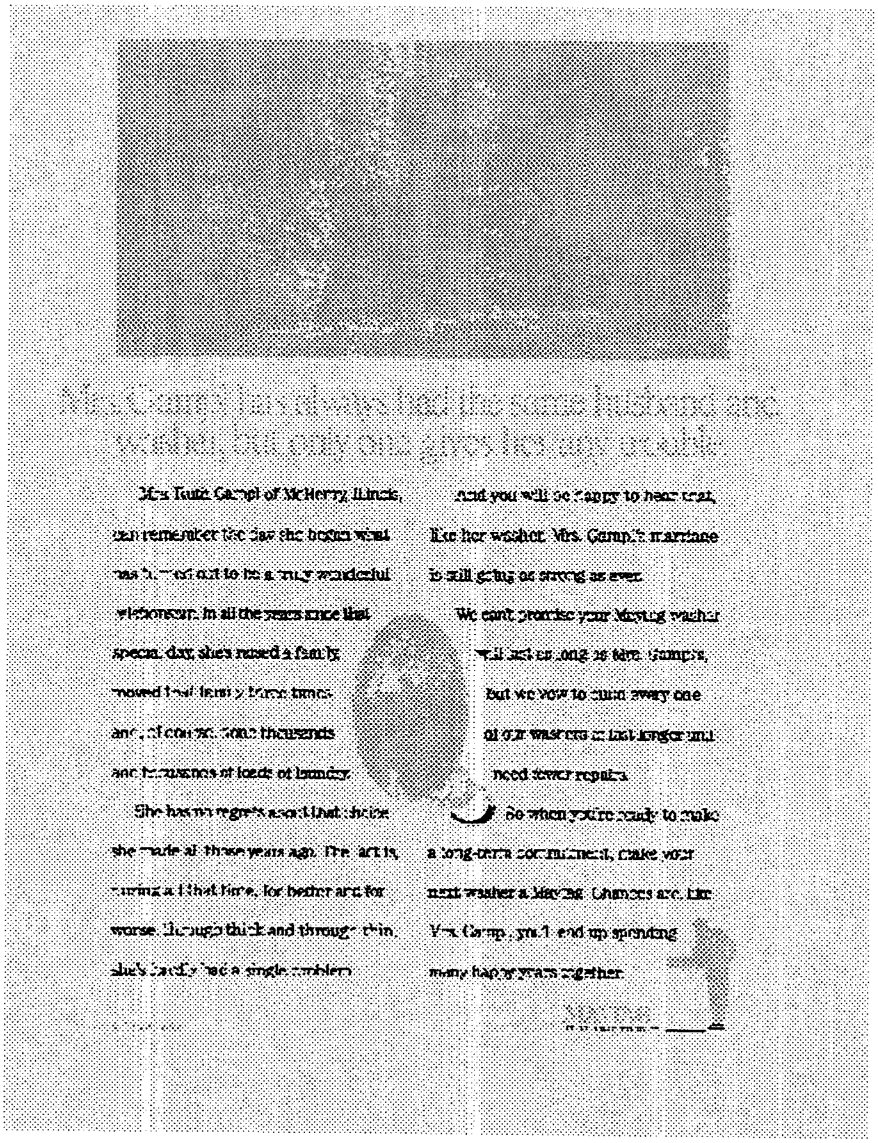


Figure 11: Error map corresponding to the segmentatino result displayed in Fig. 10. Pixels involved in a merging error are black in this map.

```
Vertical merges          =    0.00 %
Vertical splits          =    0.00 %

Overall zoning quality   =   49.22 %

LABELING:

Text recognized as image  =   0.00 %
Img. recognized as text   =   0.45 %
Text recognized as noise  =   0.00 %
Img. recognized as noise  =   0.00 %
Noise recognized as text  =   0.00 %
Noise recognized as image =   0.00 %

Overall labeling quality  =  99.55 %
```

3.4 Customizing the Benchmarking

Just like *GroundsKeeper*, our benchmarking tool *cluzo* can be customized to a particular application. The init file required by the program is decomposed into several sections:

- Listing of all the region types and attributes that can be encountered in RDIFF files
- Description of region equivalences: in this section, a user can specify that type A is in fact equivalent to type B, and that type D is equivalent to type B as well, etc. For example, if the user did not care about distinguishing between different types of image regions (such as graphics, halftones, line-art, etc) and text regions (such as captions, headlines, footers, etc), the total number of region types could be reduced to 2.
- Listing of all the different error types considered, together with their weights. This can be done for generic regions as well as for particular region types. For example, the init file we currently use contains the following:

  ```
  Weight_of Vertically_split  REGION   0
  Weight_of Vertically_split  GRAPHICS 3
  Weight_of Vertically_split  HALFTONE 3
  ```

  This specifies that a vertical split is of cost 0, except when it involves a GRAPHICS region or a HALFTONE region, in which case the associated cost becomes 3.
- Error weighting method to use. By default, each error is weighted by its intrinsic weight (as defined in the previous section) multiplied by the number of pixels involved. In some cases however, one may want to be able to specify that an error of a given type is of nominal cost, regardless of the number of pixels involved. Similarly, when text regions are involved, it may make sense to use the *height* (≈ proportional to the number of text lines) of the identified error region as weight. We are currently working on techniques to enable all these different weighting method to coexist harmonioulsy.
- Definition of the error combination method: when an area of the page is involved in multiple segmentation mistakes, should all these mistakes been taken into account, or should the most serious mistake (the one with highest associated weight) prevail?

The format of this init file is still evolving, as we are working on better and simpler ways to describe a user's benchmarking needs in this file.

4 Conclusions, Future Work

We described the region-based page segmentation benchmarking environment currently under development at Xerox. Every aspect of the system has been developed with flexibility in mind, and we always tried to think of all the different ways this system could be used in practice. Instead of computing a single number/curve characterizing segmentation accuracy, our system is able to keep track of an arbitrarily large number of error types, and therefore can provide very detailed information on segmentation quality. For these reasons, and because the system does not require any OCR, we believe that it is a compelling alternative to text-based segmentation benchmarking approaches.

Most of the functionality has been implemented, and we are currently in an incremental improvement phase. In particular, we are working on better and simpler ways to specify error types and weights in the system. We are also considering a few options for extending our benchmarking system to deal with grayscale images: this would potentially open a whole new range of new applications for this system, beyond document analysis.

The actual error numbers reported are still a bit arbitrary at times, and this makes it still difficult to use our system to compare very different segmentation algorithms. However, as is, the system is already invaluable for segmentation algorithm developers: indeed, absolute error figures are less useful to developers than relative numbers. Using *cluzo* on a large enough test suite makes it easy to automatically flag degradations and improvements in different aspects of segmentation, and to fine-tune the parameters of a given zoning algorithm.

References

[1] A. Antonacopoulos and R. Ritchings. Flexible page segmentation using the background. In *12th Int. Conf. on Pattern Recognition*, pages 339–344, Jerusalem, Oct. 1994.

[2] H. S. Baird. Background structure in document images. In *Advances in Structural and Syntactic*

Pattern Recognition, volume 5, pages 253–269. World Scientific, 1992.

[3] O. Desforges and D. Barba. Segmentation of complex documents multilevel images: a robust and fast text bodies-headers detection and extraction scheme. In *Int. Conf. on Document Analysis and Recognition.*, pages 770–773, Montreal, 1995.

[4] J. Fisher, S. Hinds, and D. D'Amato. A rule-based system for document image segmentation. In *Int. Conf. on Pattern Recognition*, pages 567–572, Atlantic City, NJ, June 1990.

[5] J. Ha, R. Haralick, and I. Phillips. Document page decomposition by the bounding-box projection technique. In *Int. Conf. on Document Analysis and Recognition.*, pages 1119–1122, Montreal, 1995.

[6] J. Kanai, T. A. Nartker, S. V. Rice, and G. Nagy. Performance metrics for document understanding systems. In *Int. Conf. on Document Analysis and Recognition.*, pages 424–427, Tsukuba, Japan, Oct. 1993. IEEE Computer Society Press.

[7] J. Kanai, S. V. Rice, T. Nartker, and G. Nagy. Automatic evaluation of OCR zoning. *IEEE Trans. Pattern Anal. Machine Intell.*, 17(1):86–90, Jan. 1995.

[8] Y.-W. Lin. Digital image processing in the Xerox docutech document processing system. In L. Vincent and T. Pavlidis, editors, *SPIE/SPSE Vol. 2181, Document Recognition*, San Jose CA, Feb. 1994.

[9] M. Nadler. A survey of document segmentation and coding techniques. *Comp. Vis., Graphics and Image Processing*, 28:240–262, 1984.

[10] G. Nagy and S. Seth. A prototype document analysis system for technical journals. *IEEE Computer*, 25(7):10–22, July 1992.

[11] L. O'Gorman. The document spectrum for bottom-up layout analysis. *IEEE Trans. Pattern Anal. Machine Intell.*, 15(10):1162–1173, Oct. 1993.

[12] T. Pavlidis and J. Zhou. Page segmentation by white streams. In *Int. Conf. on Document Analysis and Recognition.*, pages 945–953, Saint-Malo, France, 1991.

[13] T. Pavlidis and J. Zhou. Page segmentation and classification. *Comp. Vis., Graph. Im. Proc.: Graphical Models and Image Processing*, 54(6):484–496, Nov. 1992.

[14] S. Randriamasy and L. Vincent. Benchmarking page segmentation algorithms. In *IEEE Int. Computer Vision and Pattern Recog. Conference*, Seattle, WA, June 1994.

[15] S. Randriamasy, L. Vincent, and B. Wittner. An automatic benchmarking scheme for page segmentation. In L. Vincent and T. Pavlidis, editors, *SPIE/SPSE Vol. 2181, Document Recognition*, San Jose CA, Feb. 1994.

[16] S. V. Rice, F. Jenkins, and T. A. Nartker. The fourth annual test of OCR accuracy. Technical report, ISRI, 1995.

[17] S. V. Rice, J. Kanai, and T. A. Nartker. A preliminary evaluation of automatic zoning. Technical report, ISRI, 1993.

[18] S. V. Rice, J. Kanai, and T. A. Nartker. The third annual test of OCR accuracy. Technical report, ISRI, 1994.

[19] E. Ukkonen. Algorithms for approximate string matching. *Information and Control*, 64:100–118, 1985.

[20] L. Vincent. Page segmentation in Textbridge. Unpublished Document, Xerox Desktop Document Systems, 1993.

[21] B. Yanikoglu and L. Vincent. Ground-truthing and benchmarking document page segmentation. In *Int. Conf. on Document Analysis and Recognition.*, Montreal, Aug. 1995.

Systems

OCR: A Five Year Retrospective

Mindy Bokser
Caere Corp.
100 Cooper Court
Los Gatos, CA 95030

Abstract

The paper will take a retrospective look at how the state of the art in OCR has changed over the last five years, by looking at how the distribution of errors has changed for one commercial system. The purpose is to better understand the current state of the art and help guide future research efforts.

The WILDFIRE FPGA-Based Custom Computing Machine

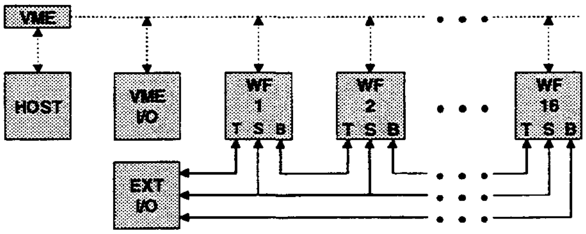
Bradley K. Fross
Annapolis Micro Systems, Inc.
190 Admiral Cochrane Drive, Suite 130
Annapolis, MD 21403
(410) 841-2514

Abstract

Modern computer technology has been evolving for nearly fifty years, and has seen many architectural innovations along the way. One of the latest advances in computer architecture is the reconfigurable processor-based custom computing machine (CCM). CCMs use field programmable gate arrays (FPGAs) as their processing cores, giving them the flexibility of software systems with performance comparable to that of dedicated custom hardware. The Splash 2 attached processor developed at the Center for Computing Sciences in Bowie, MD (formerly the Supercomputing Research Center) is an example of a CCM. The WILDFIRE architecture is based on the Splash 2 technology transferred from the Department of Defense and the Institute for Defense Analyses, Supercomputing Research Center. This architecture has been proven to effectively solve many applications related to document image understanding technology. Several image processing kernels have been developed at the Virginia Polytechnic Institute and State University for the Splash 2 and Wildfire architectures that can be used to develop document image understanding applications. Other Splash 2 applications that relate to the concept of document image understanding include textual keyword searching, DNA sequence comparison, and fingerprint image understanding. How some of these applications relate to document image understanding and an implementation of text searching using WILDFIRE are discussed in this paper.

1 Introduction

Advances in microelectronics have once again provided the means to achieve advances in computer architecture. A device called the *field programmable gate array* (FPGA), combines the flexibility of software with the high-performance characteristics of custom-built hardware [14]. The FPGA is not only ideal for rapid prototyping of digital logic, but has proven to be a very effective computational core of a new type of computer architecture called a *custom computing machine* (CCM).



HOST	: Host machine (Sun Sbus or PCI based system)
VME I/O	: VME-32 (or -64) I/O subsystem
EXT I/O	: External I/O subsystem (i.e. serial digital or video I/O)
WF1-WF16	: WILDFIRE array boards 1 through 16
T (on WF)	: Top systolic connector on WILDFIRE array board
S (on WF)	: SIMD connector on WILDFIRE array board
B (on WF)	: Bottom systolic connector on WILDFIRE array board
◄·········►	: 32-bit data path
◄————►	: 36-bit data path

Figure 1: WILDFIRE system configuration.

CCMs use FPGAs as their main processing resources in order to take advantage of their speed and the ability to change internal hardware configuration. Some examples of research-oriented CCMs include Splash [6], Splash 2 [1], Ganglion [5], and PAM [3]. An example of a commercially available CCM is the WILDFIRE system, developed by Annapolis Micro Systems, Inc. WILDFIRE, shown in Figure 1, is based on the Splash 2 CCM architecture that was developed at the Center for Computing Sciences (formerly the Supercomputing Research Center).

The CCM architecture on which WILDFIRE is based has proven to be an effective solution for many applications in several different areas. Applications in image processing [2], genetic engineering [7], digital signal processing [13], and document understanding [9] have all been shown to be good fits for the WILDFIRE architecture.

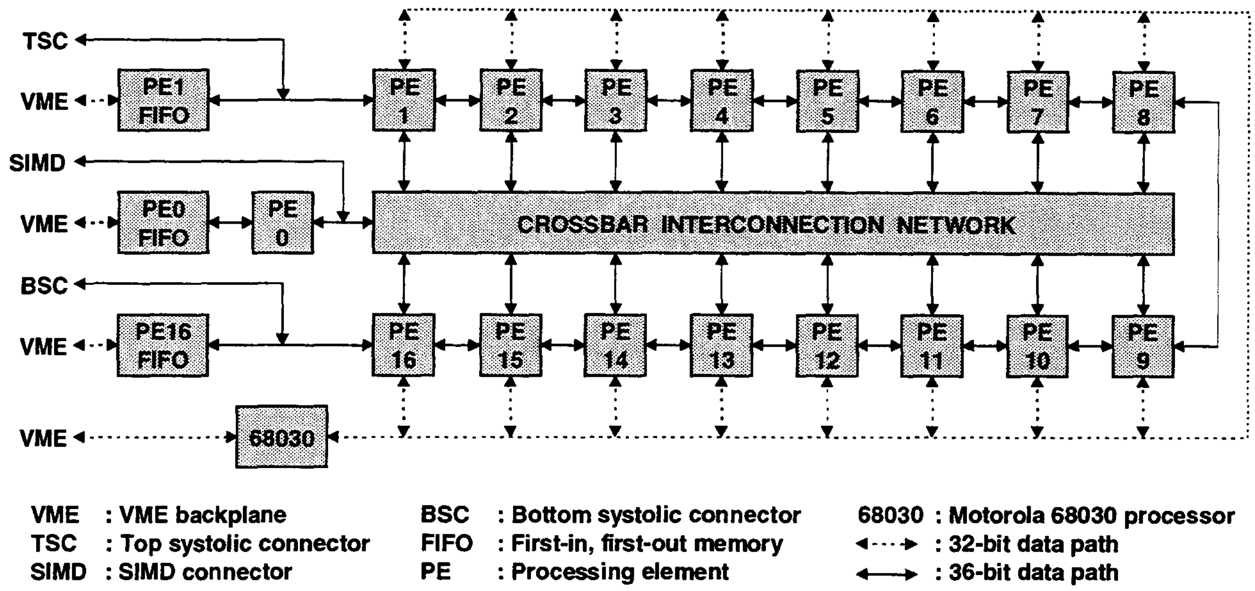


Figure 2: WILDFIRE processor array block diagram.

2 The WILDFIRE system

The WILDFIRE system is an attached processor that can be composed of one to sixteen processor array boards. The system depicted in Figure 1 shows how several WILDFIRE processor array boards can be used with a host processor and external I/O to form a complete high-performance computing system.

2.1 Architectural description

The architecture of a WILDFIRE system is divided into three parts: the host system, I/O subsystems, and the processing array itself. The host system can be a Sun Sbus-based workstation, Intel Pentium PCI-based personal computer, or a VME-based SPARC card. The VME I/O subsystems can consist of virtually any VME-based cards, including high-speed disk or tape subsystems, memory cards, or bus-translation cards. The external I/O subsystem can consist of custom high-speed serial or parallel devices such as network cards, digitized voice channels, video frame-grabbers, or other WILDFIRE systems.

The WILDFIRE processor array is shown in more detail in Figure 2. Each array consists of sixteen processing elements (PEs one through sixteen) that are attached to their nearest neighbors via direct connections. These direct connections between PEs form a one-dimensional systolic array (or linear array) bi-directional data path. PEs one through sixteen are also connected to each other via a full crossbar interconnection network. The crossbar network allows for other interconnection network topologies to be created, such as ring, star, toroid, mesh and other networks. All network data paths connecting PEs together are 36 bits wide.

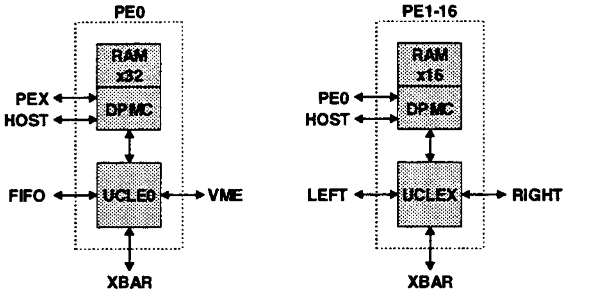
An additional processing element called PE0 is present on each array to provide a means to control the configurations of the crossbar, to control the data flow between the host-accessible FIFOs and the array, and also to perform processing tasks.

The bi-directional FIFOs on the WILDFIRE array board are used to buffer the data to and from the host. The PE1 and PE16 input and output FIFOs are each 16k 36-bit words long, while the PE0 input and output FIFOs are each 32k 36-bit words long. The systolic and SIMD connectors on the array board can be used to connect multiple boards together in various configurations, or can be used to connect an array board to external I/O subsystems.

The configuration shown in Figure 1 uses the top and bottom systolic connectors to create a linear array of 256 PEs that is connected to external I/O. Also, a SIMD broadcast network can be created by having each array board receive instructions from an external source via the SIMD connectors and sending these instructions to each of the PEs through the crossbar network.

The host can monitor or send data to the arrays through the VME backplane and 68030 processor on the processor array board. The 68030 also performs on-board diagnostics, sets up data transfers to/from the host, processes on-board interrupts, and handles PE configuration and state readback.

The PEs on the processor array are made up of three components: a *user-configurable logic element* (UCLE), a memory component, and a *dual-port* *memory controller* (DPMC). Figure 3 shows a block diagram of the different PE structures.



UCLE0: User configurable logic element
DPMC : Dual-ported memory controller
RAMx32: 20 ns 256k x 32-bit SRAM module
PEX : Port to/from PE1-PE16
HOST : Port to/from HOST
FIFO : Port to/from PE0 FIFO
VME : Port to/from VME
XBAR : Port to/from crossbar

UCLEX: User configurable logic elements 1 - 16
DPMC : Dual-ported memory controller
RAMx16: 20 ns 256k x 16-bit SRAM module
PE0 : Port to/from PE0
HOST : Port to/from HOST
LEFT : Port to/from PE or FIFO on left
RIGHT: Port to/from PE or FIFO on right
XBAR : Port to/from crossbar

Figure 3: Processing elements PE0 and PE1-16.

The PE0 processing element is sometimes referred to as the control element because it is used to control the configuration of the crossbar and the flow of data through the FIFOs. The UCLE0 portion of PE0 is composed of a Xilinx XC4025 FPGA device, which has 25,000 usable gates and 2560 flip/flops, that can be used to process data as well as control the array. PE0 has a 256k by 32-bit SRAM attached to it through a DPMC. This memory is twice as wide as the memory of devices PE1 through PE16, and can be used for buffering data or as a lookup table for performing various computations. Data and control paths between PE0 and the host, PE1 through PE16, the FIFOs, the crossbar, and other WILDFIRE arrays can be used to implement various communication patterns.

Processing elements PE1 through PE16 (denoted as PEX) are the processing core on a WILDFIRE array card. Each PEX can communicate with any other PEX through the three 36-bit LEFT, RIGHT, or XBAR data paths shown in Figure 3. Other data paths are available for communicating with PE0 or the host machine. The UCLEX part of PEX consists of a Xilinx XC4010 FPGA device, which has approximately 10,000 usable gates for implementing combinatorial logic, and 1120 bits of state (i.e. flip/flops). The memory component attached to each UCLEX via a DPMC is a 256k by 16-bit SRAM device.

Each PE is programmed using a mix of behavioral and structural VHDL. The VHDL code used to program the PEs can be attached to a VHDL simulation of a WILDFIRE system in order to verify the correct behavior of the system. Once the correct behavior is achieved, the VHDL code for each of the PEs can then be synthesized into separate PE configuration bit-streams that can be downloaded into the actual PE devices. A C-language program is used to control the interaction and data flow between the WILDFIRE arrays and the host machine.

2.2 System performance

The general performance of the WILDFIRE system depends on the performance of the user-configurable logic elements. More efficient UCLE designs result in higher performance. However, the peak performance of the WILDFIRE array board depends mainly on processor clock rates and I/O throughput.

The maximum WILDFIRE processor array clock frequency is 25 MHz. This means that the maximum data throughput along the any one of the linear array or crossbar data paths is approximately 113 Mbytes per second. At the same clock rate, the maximum aggregate memory bandwidth of a single processor array board is approximately 900 Mbytes per second. This does not include the FIFO memories, which can achieve sustained aggregate transfer rates of up to 50 Mbytes per second. The FIFOs are limited to VME backplane transfer rates, but the front panel connectors can easily achieve the maximum throughput rates of 113 Mbytes per second.

Typical processor clock rates are between 10 and 20 MHz. These rates greatly depend upon the nature of the WILDFIRE application. The next section describes several WILDFIRE applications and describes their performance.

2.3 WILDFIRE applications

Many applications that have been written for the Splash 2 CCM can be ported directly to the WILDFIRE CCM with minimal modifications. As stated earlier, many of these applications fall into the categories of image processing, digital signal processing, and several types of searching algorithms. Several of the image processing applications that are discussed in [2] are similar to document image understanding applications such as multi-lingual OCR and document layout analysis. Performance results of several of these applications are shown in Table 1.

Another application that has been proven to be a successful implementation on the architecture used in the WILDFIRE CCM is document database text searching [9]. The remaining sections of this paper are used to illustrate how text searching can be implemented on WILDFIRE, and how it can be used to perform document database searching and document language identification.

Table 1: Approximate performance of image processing tasks on Splash 2 and WILDFIRE (in millions of operations per second) [2].

Image Processing Task	Memory Accesses	Arithmetic/ Logical	Floating Point
Histogram	20	120	N/A
Region Labeling	30	190	N/A
Median Filter	60	280	N/A
Hough Transform	20	390	N/A
Pyramid Generation	50	380	N/A
Morphological Operations	20	480	N/A
8x8 Convolution	30	650	N/A
2-D FFT	260	200	220

3 Text searching

As the information age descends fully upon us, efficient methods of searching through the vast amounts of data being generated daily become essential. Many types of information databases, such as legal and medical document databases, are highly structured and can therefore be searched using inverted file methods [11]. However, many other types of document databases, such as news story databases, are not structured with indexes and full text searches must be performed [12], [10].

The text based keyword searching algorithm described in this paper was originally designed for operation on Splash 1, then modified for Splash 2. For more information on these Splash 1 and Splash 2 implementations of this algorithm, refer to [8] and [9], respectively.

Text searching can be used to search a document database for certain words that are contained in a keyword dictionary. One algorithm designed for this purpose uses hashing to determine the presence of keywords in document database. Words of the document database are converted into hash functions which are used as addresses into hash tables which test for the inclusion, or "hit", of a word in the search list. Once a hit has been found, the word is marked as a keyword and its location in the document database is reported.

The traditional von Neumann architecture of general purpose computers limits them to slower sequential implementations which cannot compete with CCM or ASIC parallel implementations of these algorithms. CCMs, unlike their ASIC counterparts, have the flexibility to change hashing functions for different types of databases. The next section describes one such algorithm designed for the Splash 2 CCM.

3.1 Splash 2 algorithm description

The algorithm described in [9] that is designed for text searching on Splash 2 streams words from a document database through the linear array of processing elements. Each processing element calculates a different hash function for each word of the database and checks its hash table for a hit. Only if each of the processing elements records a hit will the occurrence of a keyword in the database be recorded.

This algorithm consists of three stages: input character formatting, hash function generation and table lookup, and keyword hit recording. The first stage is fairly simple and involves the formatting of a 32-bit data stream of words into sets of two characters each.

In the English language, a space delimits the separation of two words, and therefore signifies the beginning or end of a word. For instance, an occurrence of the word "the" in a document database would be denoted by one of the two four-character sequences, "the_" or "_the" (where "_" denotes a blank space). If the algorithm is case-insensitive, each of the 62 English lexico-numeric characters can easily be represented by an 8-bit quantity. Therefore, two-character sequences can be represented as 16-bit quantities. The first form of the word "the_" would be separated into the 16-bit sequences "th" and "e_", while the second word would be separated into "_t" and "he". The first stage will then signal to the second stage when once an entire 16-bit formatted word has been transmitted.

The second stage is responsible for generating the hashing functions that will be used as addresses into the hash tables. Each processing element is used to calculate a different hash function for the same word. The hash function is calculated by first clearing the 22-bit hash register, then inputting a 16-bit character sequence. The hash function mask is then applied to the character sequence, and the result is circular shifted to the right and loaded into the upper 16-bits of the hash register. The next 16-bit character sequence is then loaded, and the process is repeated until the end of the word is reached. The value stored in the 22-bit hash register is then used to check the hash table for a hit.

Since the PE memories are 256k by 16-bits, they contain exactly 2^{22} bits. If the bit at the location in the hash table referenced by the hash function contains a one, then there is relatively high probability that a match has been found. Figure 4 shows an example of how the word "the_" is found in the hash table.

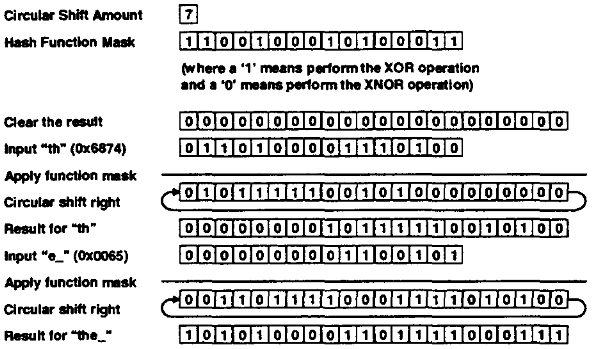


Figure 4: Hashing example for the word "the_".

An example of the probability of having a false hit can be illustrated by using the entire English language as the list of keywords. The list of case-insensitive keywords would number around 2^{18} words. Since each word can start on either an even or odd byte boundary (as shown above using the word "the"), this means that $2 * 2^{18}$ words can be found in the hash function. Given that a hash table has 2^{22} locations, only ($2 * 2^{18} / 2^{22}$), or 1/8, of these locations will contain ones.

If *n* relatively independent hash functions are chosen for each of the processing elements, the resulting probability that a false hit occurs is equal to the chance of getting only one false hit among all PEs. Therefore, the probability of getting a false hit is $(1/8)^n$, or 2^{-48} for n = 16.

The final stage of this algorithm finds the location of the matching word in the input stream and reports it to the host. The hit location is kept by counting the number of words in the input stream and marking the words that result in a match.

Figure 5 shows a block diagram of the Splash-2 implementation of this algorithm. Stage one is performed by XL, stage two is performed by X1 through X16, and the last stage is completed by XR.

3.2 WILDFIRE implementation

The WILDFIRE implementation of the text search algorithm described above is very similar to the approach used for Splash 2. The major differences in implementation are mainly due to slight architectural differences and added features of WILDFIRE.

The major architectural difference between Splash 2 and WILDFIRE is that the Splash 2 system has only one interface to the host. The WILDFIRE board has one interface to the host *per array board*, so one can take advantage of broadcasting data to multiple boards to exploit higher parallelism of execution. Another architectural difference is that Splash 2 has the I/O processors XL and XR, while each WILDFIRE array card has only one PE0 to perform the same I/O functions. However, since PE0 is an XC4025 and XL and XR are only XC4010s, PE0 has 2.5 times more logic available than both XL and XR combined. Also, PE0 has a 32-bit wide memory attached to it that it can use as additional I/O between the array and the host.

The differences between the Splash 2 and WILDFIRE implementations stem from the architectural differences. The basic data flow in the WILDFIRE implementation is no longer through the linear array data path. The 16-bit character sequences are broadcast from PE0 across the crossbar to eliminate any latency due to the linear pipeline used in

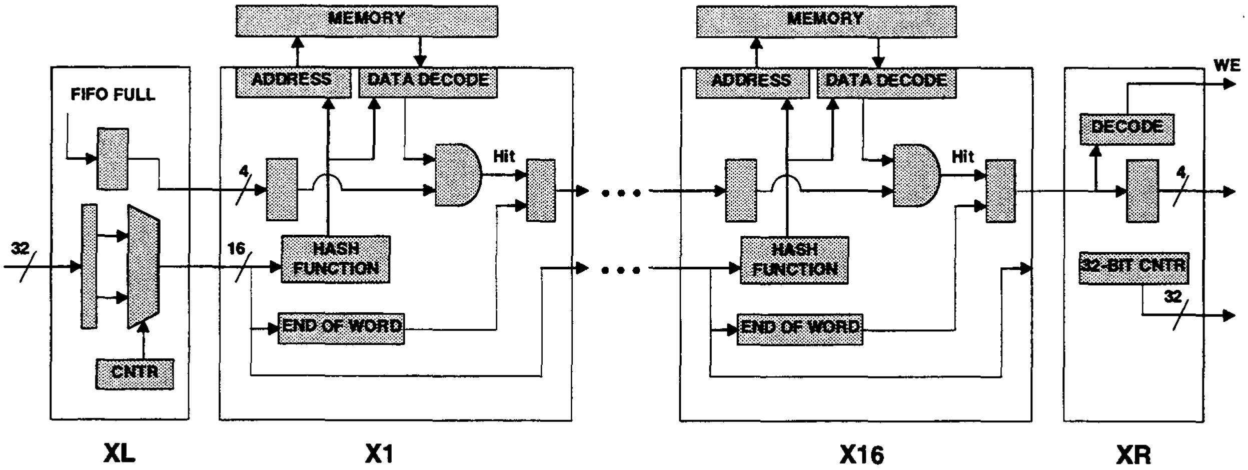


Figure 5: Splash 2 implementation of text search algorithm.

the Splash 2 approach.

The results of the individual hash table lookups are passed back to PE0. If multiple boards are performing the algorithm, the PE0 components on the additional array boards pass their "hit" results to the first array board's PE0 via the VME backplane. The PE0 of the first board then calculates whether or not a "hit" has occurred.

The remaining portions of the WILDFIRE implementation of the algorithm are basically the same as the one described above, except for the change in data flow. The next section compares the results between the Splash 2 and WILDFIRE implementations of the text search algorithm.

3.3 Algorithm performance

The Splash 2 implementation of the algorithm is able to process incoming data at the rate of approximately 20 million characters of text per second [4]. This rate is based on the fact that the average number of characters per word in the English language is 4.7 [11]. The WILDFIRE implementation should be able to run at the maximum processor clock rate of 25 MHz. This corresponds to an estimate of approximately 50 million characters per second, including blank characters. This processing rate requires the VME backplane to transfer data to WILDFIRE at 50 Mbytes per second.

The probability of a false hit using a single board system is the same for Splash 2 and WILDFIRE, which is 2^{-48}, or approximately 10^{-15}. This probability can be decreased by using a multiple board system. The Splash 2 approach would increase the latency of the result due to the longer pipeline through multiple boards. A WILDFIRE multiple array board approach would not require additional pipeline delay, due to the use of a crossbar broadcast data path.

Another advantage of a multiple array board WILDFIRE implementation is that it can search through multiple lists of keywords for hits in the same document. The multiple keyword lists could be multi-lingual dictionaries, or possibly a single dictionary that is distributed across multiple array boards. For the distributed dictionary case, the probability of a false hit is reduced to 2^{-112}, or approximately 10^{-34}.

The algorithm could also be used to detect "misses" instead of "hits". This can be used to detect when words in a document are not contained in a dictionary. Many other variations of this and other database searching algorithms have proven to be good fits for the WILDFIRE architecture [4].

4 Conclusions

The WILDFIRE text searching algorithm is an excellent example of a high-performance solution to a document image understanding problem. The image processing performance measures also show that WILDFIRE is very adept at performing image processing tasks. The image processing and searching algorithms can be incorporated into a single system that can be used to create and process document image databases. WILDFIRE continues to improve upon the successes of its Splash 2 predecessor to become one of the most powerful machines that is commercially available today.

References

[1] J. M. Arnold, D. A. Buell, and E. G. Davis. Splash 2. In *ACM Symposium on Parallel Algorithms and Architectures*, 1992, 316-322.

[2] P. Athanas and L. Abbott. Real-time image processing on a custom computing platform. In *IEEE Computer*, **28**(2) pages 16-24, 1995

[3] P. Bertin, D. Roncin, and J. Vuillemin. Programmable active memories: a performance assessment. In *Symposium on Integrated Systems*, MIT Press, 1993.

[4] D. Buell, J. Arnold, W. Kleinfelder, editors. DRAFT: Splash 2: FPGAs in a Custom Computing Machine. To be published under same title, IEEE Press, 1995.

[5] C. E. Cox and W. Ekkehard Blanz. Ganglion - a fast hardware implementation of a connectionist classifier. In *Proceedings of the 1991 IEEE Custom Integrated Circuits Conference*, 6.5.1-6.5.4, 1991. IBM Research Report RJ8290.

[6] M. Gokhale, W. Holmes, A. Kopser, S. Lucas, R. Minnich, D. Sweely, and D. Lopresti. Building and using a highly parallel programmable logic array. *IEEE Computer*, **24**, pages 81-89, 1991.

[7] D. T. Hoang. Searching genetic databases on Splash 2. In D. A. Buell and K. L. Pocek, editors, *IEEE Workshop on FPGAs for Custom Computing Machines,* pages 185-192. IEEE, 1993.

[8] J. T. McHenry and A. Kopser. Dictionary search application on Splash. SRC Internal Report, July, 1991.

[9] D. V. Pryor, M. R. Thistle, and N. Shirazi. Text searching on Splash 2. In D. A. Buell and K. L. Pocek, editors, *IEEE Workshop on FPGAs for Custom Computing Machines,* pages 172-178. IEEE, 1993.

[10] G. Purcell and D. Mar. SCOUT: Information retrieval from full-text medical literature.

Knowledge Systems Laboratory Report KSL-92-35, Stanford University, 1992.

[11] G. Salton. *Automatic Text Processing*. Addison-Wesley, Reading, MA, 1989.

[12] C. Stanfill and B. Kahle. Parallel free-text search on the Connection Machine System. *Communications of the ACM*, pages 1229-1239, 1986.

[13] A. Walters. Implementing a 32-tap FIR filter on Splash 2. Masters Thesis in progress, 1995.

[14] Xilinx, Inc. *The Programmable Gate Array Data Book*, Xilinx, 2100 Logic Drive, San Jose, CA, 95124, 1994.

DAFS: A Standard for Document and Image Understanding

Tom Fruchterman
RAF Technology
16650 NE 79th St., Suite 200
Redmond WA 98052
tom@raf.com

Abstract

The Document Attribute Format Specification (DAFS) is a new format for Document and Image Understanding. It improves over commercial OCR formats by being free, open, and capable of representing all scripts and languages, because it uses the Unicode character encoding standard. For document understanding, it uses a flexible, SGML-like hierarchy and tagging system. Unlike SGML, it is binary, compact, and capable of storing image and data in one file. It is designed to straightforwardly handle document decomposition. DAFS can represent uncertainty in recognition and page decomposition with confidence values and alternative interpretations. Developers have complete support in adding their own types and data. DAFS is both a file format and a library for manipulating structures programmatically. The Illuminator program is an X-Windows/Motif editor and viewer for DAFS.

1. Introduction

While many formats exist for composing a document from electronic storage onto paper, no satisfactory standard exists for the reverse process. The Document Attribute Format Specification (DAFS) is intended to be a standard for document decomposition, for applications requiring document image understanding and the exchange of data from document images. DAFS is a file format, but it has a C-language API which we refer to as the DAFSLib.

DAFS is a "rich" format designed for encoding any information which might be extracted from a document — from content to physical structure to logical structures interrelated across different parts of a document. The key features of DAFS are hierarchical structure, the ability to encode all languages and scripts, the ability to express ambiguity and uncertainty, and uniform treatment and file storage of image and text. Combined with its easy convertibility into other formats, this makes DAFS ideal as a document interchange format.

Since we are already overwhelmed with standards that attempt to address document composition and interchange, such as SGML, RTF, PDF, why do we need a new standard for document decomposition? Quite simply, the requirements for document decomposition are very different from those for document composition.

Existing commercial OCR formats generally have limited abilities to represent document structure. They are not open formats, either in the sense of being well-documented or in the sense of allowing application-defined data. DAFS is designed to be useful from the beginning of the document understanding process (an ordinary captured image), through every stage to the final result (a document with both text and structure labeled according to the user's requirement). From this point, DAFS' similarity to SGML allows easy conversion to SGML or other fancy text formats. At every step in the process of document understanding, DAFS is capable of representing all the information traditionally used in the process, with hooks for application-defined data.

The design of DAFS was a joint effort by RAF Technology and a working group organized by the Advanced Research Project Agency (ARPA). The format has been implemented by RAF Technology. In this paper we talk about the design and implementation of DAFS, with a brief description of Illuminator and some sample scenarios of using DAFS and Illuminator.

2. Design Considerations

In publishing applications, documents are "encoded" via standards such as SGML in preparation for the actual printing process. In the document understanding and page decomposition arena, we perform "reverse encoding", seeking to reverse-engineer the meaning from an image of the printed page. The greatest difference between encoding for document creation and the reverse encoding of document images is that during the reverse encoding process, there may be varying levels of uncertainty in the interpretation of aspects within the document. In the document creation process, this ambiguity is not present, since the document is usually being encoded by the same person who created the representation of the document. A data format for document reverse encoding must have a mechanism for representing these ambiguities.

Document composition systems usually store an internal format and generate a rendering, whereas a document decomposition system starts with the rendering, and must work backwards. Thus, even if a composition standard allows mixed image and text, it doesn't associate the image with the text. An interesting hybrid, hoever, is the PDF format used by Adobe. Normally used as an interchange format, there is also an OCR package called Acrobat Capture that recognizes what it can, and leaves uncertain sections in the raw image format.

Document encoding standards have evolved over the years from raw text, to text with escape codes, to text with physical tagging, and finally to text with logical tagging. In a similar way, OCR developers and researchers are realizing the importance of doing more than just reading individual characters. When we know more about the document structure, we have two benefits:

- higher level understanding gives us context to improve low-level recognition accuracy ;
- document understanding is what we really want.

It is thus desirable for a document decoding standard to have the ability to express complicated physical and logical structure. When composing a document in a system such as SGML, the user can be held to the strict rules of a document archetype. A document decomposing standard must be much more forgiving, since the data can be incomplete, incorrect, or uncertain.

3. DAFS Implementation

3.1. Basic Concepts

The basic unit of a DAFS file is an entity, which can be any object of interest in the document. The entities can be arranged in arbitrarily complex hierarchies. See Figure 1 for a sample hierarchy.

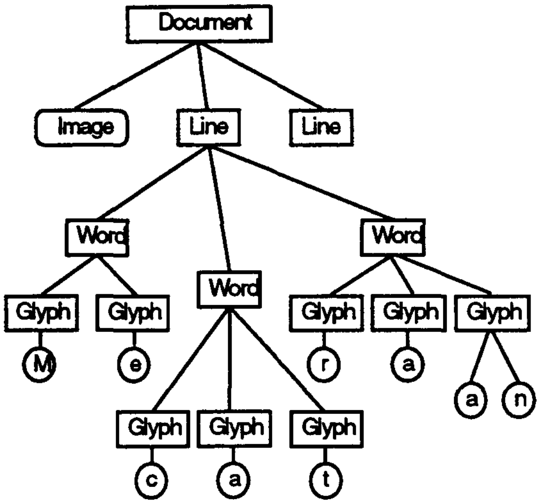


Figure 1: Example Document Structure

There are two sorts of data associated with an entity: traditional information such as pointers to image, bounding boxes, and text content; and arbitrary programmer-defined data. All the data in the former class have specialized formats and get and set functions. The data in the latter class are represented by properties, which are name, value pairs. An entity need have every sort of data. For example, a logo data base would have no text content associated with entities, just image. On the other hand, a document that was in the final process of being OCR'd, having already been scanned and corrected, might have all references to images and bounds removed, retaining only text and hierarchy for its meaning.

Every entity has a type, which is simply a string associated with that entity. In some other programming systems, the word "type" implies a regularity and definite nature that is not present in DAFS. All DAFS entities are alike; the meaning associated with the entity type is merely conventional. There are two exceptions to this principle. First, properties may be associated with types as well as individual entities. Second, the *illum* program has some hardwired assumptions about type names in its Text Mode. DAFS cannot afford to be as restrictive with types as, say, SGML, because of the essentially incremental and uncertain process of document understanding.

A typical convention might be to give the entities of a document the following types: Glyph, Space, Word, Line, Newline, Paragraph, and the Document as a whole. Each entity is a region of image delimited by an image bounds. For example, a Word entity is represented by a region bounding the image of a word on screen. Typically, the image of a line of text will be associated with a Line entity which may have Word child entities; each Word may have Glyph children; each Glyph may be labeled with its textual content (a character). The choice of names and the structure of the hierarchy in the above example is purely conventional, and the applications programmer is free to invent his own conventions.

DAFS images on disk are stored in a run-length compressed format based on the TIFF standard. In memory, they are expanded to a form more suitable for run-time manipulation. Typically one-third the size of a bitmap, they are can be efficiently scaled, rotated, or converted to a bitmap.

All DAFS functions that deal with text come in both ASCII and Unicode flavors. Unicode is the emerging worldwide character encoding standard. Each character is a fixed two bytes, and most characters are easily translated to national character standards, especially ASCII characters. DAFSLib also includes functions that are Unicode analogues to the standard C ASCII string and character functions.

3.2. Textual Content

Every entity can have the following types of textual content.

- (Unicode) character
- ASCII string
- Unicode string
- (Unicode) character possibility set
- integer value
- no data

In the next section, we will see how to express ambiguity or uncertainty using "borrows" and "ors", but there is a more efficient way for characters. A possibility set is a list of character value, confidence pairs. The entity is considered to have one of the character values in the possibility set. The associated confidences are not necessarily probabilities, but higher numbers are considered by convention to be more likely.

3.3. Hierarchy

DAFSLib has functions that allow you to create, traverse and rearrange hierarchies. It also has several functions that for searching in read order for entities that match specified criteria. Two non-traditional features are referred to as "ors" and "borrows". Or paths allow multiple interpretations of the same part of a document. Borrowed entities allow multiple representations of a document to share data where feasible.

An example of the use of borrowed entities is a document understanding program that wants to have hierarchies that represent both logical and physical structure of a document. The ordering arrangement of the higher level entities in these two sub-hierarchies would be different, but the smallest elements, say words and characters, would be the same. One way to do this would be to create completely parallel hierarchies down to the smallest elements. The problem with this approach is that the data would be duplicated, using memory and requiring any editor to be able to make corrections at the smallest levels to modify the structure in multiple places or to allow the two hierarchies to get out of sync.

The solution is for the second hierarchy to "borrow" the data in the lowest level of the hierarchy. Each entity at the lowest level has an independent place in the hierarchy but shares the same data (image pointer, bounding box, and text), as the corresponding entity in the other branch. When the data pointed to by one is edited, it is the same data that its analogue points to, and so both are updated. As a result, the hierarchy is really a Directed Acyclic Graph (DAFS prevents an entity from borrowing an ancestor).

DAFS represents uncertainty via alternative branches in the hierarchy. Every entity that has children has a logical type field that describes how the children are to be interpreted. Most entities have "and" children. An example is Word whose children are Glyphs; the entity is interpreted to be composed of the sum of the children. On the other hand, when a recognition algorithm is uncertain, an entity will be given "or" children. All the children of that entity are interpreted as different possible alternatives of the parent entity; implying that just one of these alternatives must be chosen to reconstruct the entity.

By convention, then, if a parent has "and" children, the entity types of the children will be "one less" than that of the parent. For example, the children of "Paragraph" might be "Line". If an entity has "or" children the types of the children will be "equal" to that of the parent. Nothing enforces these conventions. Borrowed entities are often used in combination with "or" paths. Figure 2 gives an example with "ors" and "borrows".

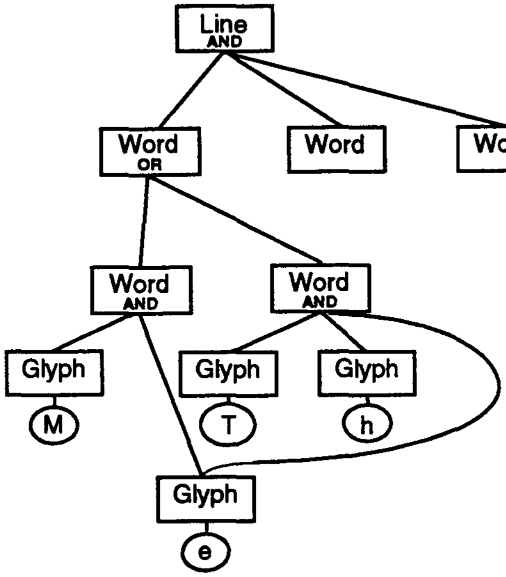


Figure 2: Structure Using Or and Borrow

4. Properties

Properties allow the application's programmer to attach arbitrary data to entities. Each property has a name, and a value. One entity can have only one property of a given name. The value can be of several types

- ASCII string
- Unicode string
- Integer
- Boolean
- User Data — permanent user data
- User Pointer — temporary user data

DAFS provides two methods for storing arbitrary user data. The first is user data. It has the constraints that it must be a pointer to contiguous memory and that a swapping function (described below) must be supplied. User data properties are saved to disk like other properties. User pointer properties can be any kind of data that can fit in a pointer. DAFS does not know anything about such data so it is treated a little differently. Other property values are copied into DAFS own data space, which implies a memory allocation for those property values that are pointers. In contrast, DAFS just remembers the original value of a user pointer property. Because it has no idea what is in a User pointer property, DAFS does not attempt to save it to disk.

In addition to being able to set and get properties, one can also iterate through the all the properties in an entity. The order of properties is not specified.

One of the common problems in file portability is *byte-swapping* . Different machines order bytes differently. For example if a short is located at address 0x1234 on a Motorola 68000 processor, then the more significant byte of the short will be located at 0x1234 and the less significant byte will be located at 0x1235, in contrast, on an Intel 8086 processor, the more significant byte would be located at 0x1235 and less significant byte at 0x1234. DAFS automatically takes into account that DAFS files may be written on different machines. It keeps track of the origin of all data and swaps it as necessary. However, the internal structure of a user data property is known only to the application programmer, and so he must supply a swapping function when such a value is retrieved.

DAFS also allows the association of properties with an entity type. These act as "default" values for entities of the specified type, but a value for the specified property assigned to a specific entity overrides the "type" property. Type properties are not saved to or read from disk in the current implementation of DAFS.

4.1. How Entities Refer To Images

It is common for all the images in a hierarchy to refer to the same image. To make this easier, whenever an entity is created, it is automatically set to point to the same image as its parent does. However, an entity hierarchy can still refer to multiple images or none at all. Each entity can refer to a portion of its image using one of three formats. The three formats are:

- Box - a simple rectangular shape
- Box List - a collection of rectangles
- Orthogon List - a collection of orthogons

An orthogon is a polygon where the angle between any 3 successive points is a right angle. The box list was requested by OCR developers, whereas the orthogon list made it easier to render complex areas graphical user interface.

4.2. DAFS File Formats

Currently DAFS files may be stored in DAFS-BINARY format. DAFS uses a tagging system reminiscent of SGML, but is not human-readable because it contains binary images. The ASCII text portion of a DAFS file (the combined entity textual content) may also be saved as ASCII text with Unicode characters encoded using the UTF/FSS multi-byte encoding scheme, or as Unicode text encoded according to the Unicode Standard. Of course, all the information about the image or the bounding boxes is lost in such a transformation. SGML output is planned for the future.

The DAFS Library can read in bi-level TIFF images or PDA (a Caere proprietary format) images, and convert them to DAFS. It is also possible to avoid converting the image itself and maintain a reference to an external image file.

4.3. Miscellaneous

In this section we talk about a select number of miscellaneous features of DAFSLib. *Callbacks* make it easier for an application programmer to write a user interface for DAFS. When DAFS structures are created, modified, or destroyed, the DAFSLib will call a user specified function with a code specifying the nature of the callback and a pointer to the affected entity.

An application programmer can sort the children of an entity, for example, by the increasing *y* coordinate of the upper left hand corner.

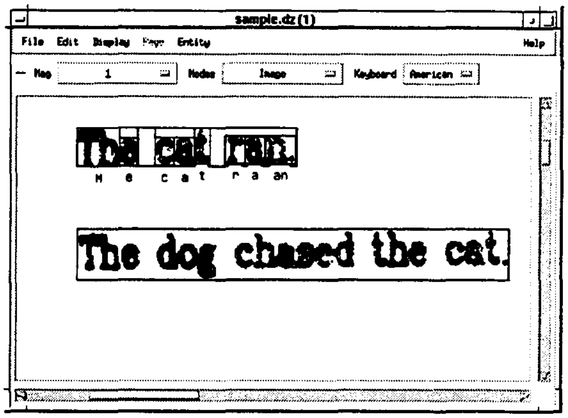
By default, DAFS uses the standard stream library. For embedded applications, the programmer can override these and supply his own low-level file io commands.

5. Illuminator

Illuminator is a DAFS file view and editor. Figure 3 shows the sample document that has been the basis for our previous examples in Illuminator. The program has 5 viewing modes, each of which captures a different facet of the documents structure:

- Image mode
- Text mode
- Out-Of-Context mode
- Flag mode
- Tree mode

No assumptions are ever made about whether a document has a certain structure or has certain data stored in it. For example, if there are no images, image mode will simply show a blank screen. If there are 15 images, the user will be able to view any of them in image mode.



sample.dz (1)

Map 1

Modes Image

Keyboard American

Figure 3: Illuminator

There are also a number of dialogs that aid in international text input and in viewing the contents of individual entities.

5.1. Image Mode

Image mode sorts out an entity hierarchy according to which image each entity points to. Each image in the document gets a separate page and only one is visible at a time. The current image is displayed with the bounding boxes and contents of the entities overlying it. Different entity types can be represented in separate user-specified colors. All entity types can be displayed, or just the types selected by the user. Entities may be created and deleted. A given entity's content can easily be edited, and its bounding box moved or resized. Those which do not have bounding boxes cannot be directly represented, but they may be indirectly affected if a related entity is modified.

5.2. Text Mode

Text Mode displays the text contents of entities for which such text exists. Special entities such as Space and Newline allow the text to be laid out in lines with spaces. There is currently no support for separate pages or columns. If the document does not contain formatting information, the text is run together on a single line. One can add or delete text in a manner similar to a basic text editor.

Illuminator also can jump to entities that have been flagged by an OCR program as questionable. An image of the suspect entity pops up and the flag can be cleared or the text can be corrected.

One can even create a de novo (from scratch) document, with no image whatsoever, or edit a plain Unicode file. Because the text mode is so crude, the only reason one would do this is to take advantage of Illuminator's rare ability to display and input Unicode.

5.3. OOC Mode

In the Out-of-Context (OOC) mode, all the Glyphs of one text value can be displayed together. The user is able to bring up a window of just the 'a's, for example. They will be displayed packed from left to right and top to bottom, without regard to the original location on the page or even what image (in the case of a document with multiple images) they came from.

OOC Mode is a completely different way of verifying a document's accuracy, and provides a powerful tool for creating error-free documents and training sets. Simply proof-reading a document does not produce an error-free document. The very abilities which make people good at reading degraded documents, and capable of filling in a pattern from a few clues, also cause them to miss many kinds of typographical errors. This is especially true for documents which were recognized by OCR, where errors tend to look like the correct result. If, for example, the word 'bold' were misrecognized as 'hold', a proof-reader might well read 'hold' as the similar-looking 'bold' just because the surrounding context would suggest it. A spell-check program would also miss this particular error. Because Out-of-Context Mode brings all entities of the same type into one window, all the contextual information is stripped out of the document. The user cannot read the document and therefore cannot read right over errors. Errors can usually be spotted with a quick glance at the page.

5.4. Flagged Mode

The look of Flagged Mode is somewhat similar to Out-of-Context Mode. Flagged Mode is for correcting entities which have been flagged by OCR as questionable or unknown. Some OCR programs may flag single glyphs; some may only flag entire words. The flagged entities are grouped by entity type rather than by content, so that the user may view a page of flagged Glyphs, a page of flagged Words, flagged Lines, etc. The OCR-derived content of each entity is displayed along with its image, and the mode has key bindings that allow quick processing.

5.5. Tree Mode

Tree Mode displays information about the entity hierarchy in compact form, listing the entities in pre-order. The Tree Mode replaces the hierarchy mode that debuted in release 0.9, and is the one feature described in this paper that is substantially different from the available version. Tree mode allows you to choose or paths, rearrange hierarchies, and collapse a hierarchy in much the same way you might collapse a directory in the Macintosh file finder. Different entity types are displayed in the same user-configured color scheme as the image mode.

5.6. Dialogs

The keyboard dialog allows one to override the standard American English keyboard and input characters from the supplied scripts. Included are most Western European languages, Greek, and Cyrillic. Arabic and Korean are in the works.
The Japanese input dialog allows one to input Japanese characters by inputting the sound of a phrase and choosing from a list of possible characters that match.

The Entity dialog displays information about a selected entity: its image, bounding box, properties, and text content.

6. Scenarios For Using DAFS and Illuminator

6.1. Development

We give the example of bootstrapping the development of a new character recognizer for a new script or language.

1. Label initial training set, and create ground truth set.
2. Create a character classifier.
3. Run classifier on a larger set of documents than the initial set.
4. Correct the results of the classifier using ooc mode of illum some analysis here may tell us that some characters in the initial training set was mislabeled, or uncover some flaw in the classifier.
5. Improve classifier.
6. Run classifier against ground truth, and compare results. RAF has created an error count program that generates both statistics and "diff" files.
7. Repeat step 5., until we are satisfied with classifier.

There are a number of alternatives for step 1.

1. Draw bounding boxes around characters in image mode of Illuminator and label them.

2. Have a segmenter block the characters and again label the characters by hand.
3. Generate a training set by converting the system font file. RAF has created converters for bdf, MetaFont, and TrueType fonts.

6.2. Production

We give the example of a production environment that uses DAFS and Illuminator. In the diagram, the gray region is the area in which the common format is DAFS. DAFS can be used efficiently for all document understanding tasks, because it allows the user to specify many different types of information, but never requires the kinds of information the user is not interested in. The only places is it is inappropriate to use DAFS is as the initial output of scanner, where TIFF is the standard, and DAFS has built-in support, and when the user wishes to have fancy formatting such as in Microsoft Word.

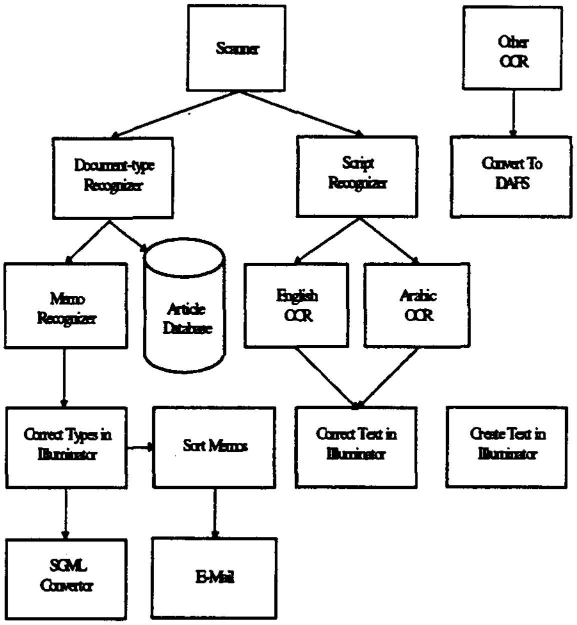


Figure 4: Production Scenario

In Figure 4, we give an example of an office that takes advantage of DAFS' document understanding, OCR, and international text capabilities. The gray boxes represent stages that use or result in non-DAFS format data. Some notes to go along with select boxes follow.:

Scanner: The DAFSLib can read the popular TIFF format.

Convert To DAFS: DAFSLib comes with a converter for PDA, and a prototype of a converter for XIS exists.

Script Recognizer: It is conceivable that both scripts might exist in one document and that the regions of like text would be so labeled.

Article Database: We posit that an operator might do some minimal organization of article images, and then users could retrieve the images. They could add their own annotations or run OCR, saving the updated article back to the database.

Correct Text in Illuminator: One would probably use the flag popup in the text mode or the flag mode. The OOC mode can help in a pinch.

Correct Types in Illuminator: An operator views the memo in image mode and verifies that the **to**, **from**, and **subject** fields are labeled correctly, and corrects them if they are not. Set up the colors that each type is to be drawn in to make this easier.

Create Text in Illuminator: Take advantage of Illuminators rare international text input capabilities to create a Unicode text document.

SGML Converter: Simple write a program that recurses through the DAFS hierarchy, printing out the entity type for the begin and end tag and the children or the text content in the middle.

7. Availability

DAFSLib is the C-language API for reading, writing and manipulating DAFS structures. It is available in a compiled form for Sun SPARC workstations, SGI Workstations, and PCs running Linux (a free Unix clone). The library has no user interface, but the illum program can be used to view and edit DAFS files. illum or Illuminator, is an X-windows/Motif application available for the same platforms as DAFSLib. The latest release of illum is version 0.9, which was completed in June 1995. The latest release of DAFSLib is version 0.9.2, and it was complete in September, 1995. The two releases are compatible and can be found at *ftp://ftp.cfar.umd.edu/pub/contrib/source/illuminator*

8. Acknowledgments

I would like to thank Mitch Buchman, our contract officer, who contributed several paragraphs to this paper. DAFS and Illum have been funded by ARPA under contract MDA 904-93-C-7219.

Turning Multimedia Data into Information

Mike Kennedy **Tom Polivka**
Excalibur Technologies Corporation
9255 Towne Centre Drive, 9th Floor
San Diego, CA 92121

Abstract

Information retrieval today stands on the threshold of becoming a strategic tool for information management. With the prolific digitization of information, the development of technical standards which cut across databases, networks and platforms, and the merging of computers and telecommunications to create an environment of virtually unlimited information distribution, there is a growing need for intelligent software that organizes, adapts, searches and retrieves information on demand.

Information retrieval, like every other aspect of the information industry, has been and will continue to be shaped by the twin forces of technological innovation and market demands. Computers entered our workplaces some forty years ago as machines with tremendous power to perform complex numerical computation and analysis at great speed. Today, they process and store not only our numbers, but our words, pictures, sounds, and signals as well. The evolution of information technology—powerful workstations on the desktop, networks, intelligent software—has brought us to the threshold of seemingly limitless data and computing power. For decades we have been asking for more, faster. Now that we have it, the real challenges of information management are about to begin.

In the final analysis, a world of electronic multimedia documents dispersed across global networks will create a virtually unlimited market for intelligent software that organizes, adapts, searches and retrieves the right information on demand. Companies that succeed in this market will be "understanding" businesses that add value by bridging the gap between data and knowledge, making information meaningful and useful.

Turning Multimedia Data Into Information

Mike Kennedy Tom Polivka
Excalibur Technologies Corporation
9255 Towne Centre Drive, 9th Floor
San Diego, CA 92121

Information retrieval today stands on the threshold of becoming a strategic tool for information management. With the prolific digitization of information, the development of technical standards which cut across databases, networks and platforms, and the merging of computers and telecommunications to create an environment of virtually unlimited information distribution, there is growing need for intelligent software that organizes, adapts, searches and retrieves information on demand.

With the proper information retrieval tools, there exists the opportunity to develop a new class of application by providing data elements such as voice, text, images and full motion video natively. This has the potential of adding incredible value to the application.

There are emerging technologies and tools in the area of parallel processors, text retrieval systems, pattern matching technology, fractal compression and natural languages that address these needs.

We can begin with a few observations made by Richard Saul Wurman in his best-selling book, *Information Anxiety.*

"More new information has been produced in the last 30 years than in the previous 5,000 . . . Today, a weekday edition of *The New York Times* contains more information than the average person was likely to come across in a lifetime in seventeenth-century England. . . . The English language now contains roughly 500,000 usable words, five times more than during the time of Shakespeare . . . The amount of available information now doubles every five years; soon it will be doubling every four."

Today we mass-produce information the way we once mass-produced automobiles, and information has become an international currency upon which fortunes will rise and fall. We will explore some of the key factors shaping the business of information, particularly as they relate to information retrieval.

Today, the glitter of information is digital. Virtually every means we have of comprehending information — words, numbers, pictures, and sounds — can now be transmitted, stored and processed as digital data. The word "multimedia," though much in vogue these days, may prove to have a remarkably short half-life. In point of fact, any and all media are, or can be, digital. Today, there is really only one primary information medium, and it is digital. To quote Nicholas Negroponte, the head of MIT's Media Laboratory, from a recent *Byte* magazine article: "Bits are bits."

But we must remember that everything that glitters isn't necessarily gold. It is curious that the word "information" has come to define our modern age but has itself become ambiguous and overused almost to the point of senselessness. Once commonly defined as instruction, teaching, and meaningful communication relating to the formation of ideas and knowledge, the word "information" has been increasingly applied as a technological term referring to anything that can be sent over wires, beamed through fiber, bounced off a satellite, or stored in a computer's memory.

This technological meaning has become so pervasive that the critical distinctions between data, information, and knowledge have been largely — and mistakenly — forgotten. The heart of the matter is this: raw data can be, but is not necessarily, information. Data are facts, information is the meaning that human beings assign to these facts. Further, meaning, or useful knowledge, is the inherent value of information. As one observer of the information age stated: "Information is not knowledge. You can mass-produce raw data and incredible quantities of facts and figures. You cannot mass-produce knowledge."

NEW ERA

We may be on the verge of a new era in information retrieval technology that will, in fact, allow us to extract knowledge from data using computers in new ways.

Information retrieval, like every other aspect of the information industry, has been and will continue to be shaped by the twin forces of technological innovation and market demands. Computers entered our workplaces some forty years ago as machines with tremendous power to perform complex numerical computation and analysis at great speed. Today, they process and store not only our numbers, but our words, pictures, sounds, and signals as well. The evolution of information technology — powerful workstations on the desktop, networks, intelligent software — has brought us to the threshold of seemingly limitless data and computing power. For decades we have been asking for more, faster. Now that we have it, the real challenges of information management are about to begin.

INFORMATION BUSINESSES

Broadly speaking, within the information industry there are four kinds of businesses.

These businesses require a non-restrictive, unstructured means to access, distribute, manage, store and retrieve information. First, and most obvious, is the creation business. Today, we can include in this category any organization — large or small — that has anything to do with the media, education, manufacturing, consulting, financial services, health care, legal services, insurance, law enforcement, government, research . . . the list goes on and on. Suffice it to say that virtually all of us, to some degree, are in the business of creating information. In large part, the information creation business represents the marketplace for the other three.

The second information business is easily recognized. It is the transmission business. Powered by brilliant technology, insatiable demand, and handsome profits, this business is virtually the exclusive domain of Fortune 500 companies: the ones who crowd the headlines with news of multi-billion dollar mergers; the general contractors who are competing to build the "information superhighway."

The third business of information, still speaking broadly, is the storage business, and this domain includes companies of every size. This is the province of computer hardware and software, and it is the area where some of the most innovative technological breakthroughs have taken place. Changes of scale in power and performance have become routine in this business. Devices that once filled entire rooms now sit on our desktops, powered by microprocessors the size of a fingertip. Yet while physically shrinking, storage technology has at the same time increased in capacity in inverse proportion to its size. Information that once filled four file cabinets now fits on a disk no bigger than our hand.

There is a fourth domain in the information industry, and it is largely unexplored. This is the new frontier of seemingly infinite information and limitless computing power. It offers untapped potential for new kinds of businesses that enable "understanding" by helping customers bridge the gap between data and knowledge, making information meaningful, applicable, and approachable. These are the businesses that will build the off-ramps of the information superhighway.

CHALLENGE OF THE 90's

The information technology of the 90s offers us the promise to access, distribute, manage, store, and retrieve information in real-time, regardless of database, network or platform. The information challenge of the 90s, driven by market demand, will be to liberate knowledge from data, empowering human skills and enabling productive action.

Traditionally, the information retrieval business has been viewed as a supporting player in the data storage business, and most retrieval companies and technologies have been content to focus on objectives defined by this limited view.

Legacy retrieval technologies, which search strings of text, depend almost exclusively on methods of organizing information by manually analyzing and classifying its content. Most early text retrieval systems, for example, only allowed searching for documents based on a few key words or phrases selected in advance by a librarian when the document was archived.

Legacy retrieval systems for structured database information have also failed to extract all of the value that can be gained from historical data. One industry observer has pointed out that "companies have spent billions of dollars over the past 10 years to create relational databases in the hopes of "extracting a broad range of information from data . . . (however) only 10% are really delivering benefits of the original investment . . . People want to know trends and patterns, you can't get that from the current implementation of SQL."

When we consider applications for multimedia information retrieval, the need for innovative technology and new methods becomes abundantly clear. Two basic challenges confront conventional retrieval technologies when we try to apply them to multimedia information:

Access

First, multimedia information can present daunting volumes of data. For example, a typical 90-minute movie, at only marginal resolution and color quality as compared to broadcast video, may require close to 60 billion bytes of storage. Even the most powerful compression would leave us with megabytes of data.

How can we expect to index, search, and retrieve on the full-contents of such information? Unless we are willing to invest in impractical levels of brute clerical force for hand-labeling every separate frame, the best that conventional SQL-based approaches can offer us is to treat the entire movie as a database BLOB. But that still leaves the problem of access to content unresolved.

The second major challenge of multimedia content-based retrieval is how to label and categorize difficult to describe data types. How do you label a sound? How do you describe a color, a shape, or a texture? Conventional models, which layer database structures or other predetermined organizational maps on top of data in order to retrieve its content, simply cannot handle the volume, or the complexity of multimedia information.

APRP

Adaptive Pattern Recognition Technology offers capabilities for efficiently, flexibly, and accurately managing many types of unstructured, multimedia data and for retrieving it based on its contents. This innovative technology has proven itself in text retrieval and document imaging applications and shows enormous potential for meeting the multimedia challenges of the future.

Adaptive Pattern Recognition Processing (APRP) is based on neural network models that recognize and index binary patterns in all types of digital data. These models consist of a set of algorithms or pattern processors which self-organize and self-adjust to the data being processed. Unlike traditional models which must be manually "trained" by programming, APRP systems automatically "learn" by processing data.

Because Pattern Recognition indexes are data-directed and self-organizing, they readily adapt to dynamic, unstructured information. Further, APRP enables automatic indexing of data, eliminating the cost, the inherent biases, and the inefficiencies of subjective, manual indexing methods.

Pattern recognition indexes also enable full content-based retrieval of digital information. APRP indexes contain the pattern ID number and a linked list of the locations in the original data where those patterns are found. Users simply present the system with a sample, or a clue, of the information they want to retrieve, and the search engine matches patterns in the clue with those in its index. In effect, the system matches the content of the sample with content in the data. Also the match does not have to be exact because of the fuzzy search which results from pattern recognition. In a text retrieval application within a document management system, fuzzy search allows users to find documents that contain the always present OCR errors and also does not prevent successful search results even with keystroke errors.

Pattern Recognition indexes are memory efficient, typically requiring only about one-third the space of keyword indexes, for example. And APRP algorithms are executable in parallel, allowing for dramatically reduced processing time using multiple processors.

APRP's integrated architecture enables content-based retrieval for any digital data, regardless of its type. As far as APRP is concerned, "bits are bits" and "patterns are patterns." The technology has proven successful in text retrieval and document imaging application products in a wide range of markets.

As an example, Pattern Recognition can learn a series of handwritten Kanji images and then when presented a Kanji character clue written by someone other than those who created the original database, an APRP based application can find the closest match.

USER-CENTRIC ENVIRONMENT

Just as the bit is the basic unit of binary information, and the database field the basic unit of structured alpha-numeric information, the electronic document will become the basic unit of unstructured multimedia information; and its dynamic character will shape the document management systems through which it flows.

- Legacy systems of the 70s and 80s created "continents of information"
- In contrast, today's IS infrastructure is a heterogeneous, highly distributed, user-centric environment.
- Organizations are rich in information only if they can bring together the diversity of platforms, data repositories, and applications under one roof.
- The growing popularity of imaging has come in large part for its obvious benefit of reducing paper.
- Imaging has played a greater role in opening the door to a common electronic metaphor by which to join otherwise separate and distinct elements of the organization — the electronic document. This shift to user-centric, document-based information is the foundation of technologies such as workflow and groupware.

NEW MARKETS

New consumer markets are emerging as well. Perhaps no more compelling metaphor has emerged in the information industry in recent memory than that of the "information superhighway." For the past year, or more, it has been difficult to pick up a newspaper or a magazine, or turn on the television, without encountering another story of another mega-billion dollar merger involving some of the largest companies in the world, vying to build the information superhighway. And then the subsequent derailment of the deal because of Judge Greene.

America loves "big ideas" and the information superhighway is one of the biggest to come along in some time. Most people aren't sure exactly what it's going to be. Ask ten different people and you're likely to get ten different answers — particularly if they happen to be in the information business.

But in the area of on-line information services, one thing that millions of people are agreeing on is the Internet, which one observer has described as " . . . a remarkable worldwide computer cooperative, a government-subsidized experiment in distributed computing, electronic community, and controlled chaos." Total subscribers on the Internet have exceeded 15 million, and are growing by as many as 150,000 new users per month. Add to this the commercial on-line services — Compuserve, America Online, etc. — and you can see just how big the potential consumer market is.

In the area of broadcast and cable television, the promise of a 500-channel universe is causing a radical rethinking of how this industry is going to do business in the future. Freed from the control of supply-side programming and scheduling, consumer television is heading into an age of microcasting, "open sea" channel-surfing, and on-demand programming. The market for helping users navigate those 500 channels is already starting to attract the attention of a number of companies, ranging from start-ups to some of the largest enterprises in the world.

INTERACTING WITH THE DATA

But, whether you connect to the data highway by copper, coaxial cable, fiber or radio, the key unanswered questions are how you will interact with the giant network — whatever it is and whatever it's called — and what you will find there. Being linked to everybody and everything in the world won't do much good if you can't use the system or locate services you need – or if there's no data on-line you care about.

Another area of immediate concern is the area of document and information management. An early promise of computing was rapid access to information, but that benefit is not necessarily delivered by traditional methods of manual searches or subjective index references upon which many "advanced" offices still rely.

Quick access to correct information enables responsiveness and competitiveness. Some organizations have implemented and expanded the use of databases and various document management software packages to meet the demand for information access. However, these approaches can't always respond to the specificity and depth required by advanced information requests. Full-featured imaging systems can provide the missing elements to immediate information access, and also complement installed database solutions.

Imaging products provide the strategic software solution that completes the suite of tools required for competitive information management.

Product requirements for choosing an imaging system include the need for:
-Versatile document handling: Organization, storage and preparation.
-Fast and easy document retrieval
-Multiple platform support: Availability on a variety of platforms, environments and compatible with multiple databases and other integral systems software. Long-term commitment to open systems and client/server architecture.
-Ease of use: Intuitive operating environment and GUI interfaces that shorten training time. Natural query processes, and a variety of file formats and data capture methods.
-Integrated imaging: Integrates scanning, OCR, display and printing, all working together seamlessly.

PATTERN-BASED RETRIEVAL

Once we've gone to the expense to load an image database, simple prudence says we should be able to find the information we need, whether we expected to need it or not. We believe that the future of imaging will be based on pattern-based retrieval of the actual digital information in the images. Pattern-based retrieval offers significantly greater flexibility than conventional methods, and we avoid re-cataloguing costs over the life of the information.

APRP permits the retrieval of digital information based on patterns in the digital information. With appropriate human interface software, a user could, for example, circle a customer name or a part number on an image of an invoice, and retrieve all of the invoices for that customer or part. If the organization later discovers a business need to locate all of the shipments to a geographical area, the system is able to respond without re-programming.

The potential applications for pattern-based retrieval technology are enormous. Digital information is expensive to store today, but not too many years ago we were concerned about storing abstracts of text documents instead of their full text for the same reasons. Ten years from now, most of our newly created information will be digital.

Imaging systems are already available with an underlying architecture that supports digital information retrieval even though the user still works with what appears to be a more conventional system. As we need to migrate to more flexible applications over time, the capabilities are already there to do so.

Information technology has already transformed our society, our economy, and our culture to such a degree that it's easy to forget that we are only at the beginning of the Information Age — and we can only speculate on the changes it will make in our lives over the long term.

The near term, however, is clear.

- As standards become widely adopted, technology borders will begin to disappear.

- As computer power and storage become even more inexpensive and widespread, and as computers and telecommunications combine to create unlimited information distribution, universal access to information will become a practical reality.

• Many industry analysts believe that in the next few years, electronic documents will become containers for many types of multimedia data, including text, still images, full motion video, voices and other sounds. While the technology is well in hand for converting these data types to digital form, application tools for managing, categorizing, filing and retrieving multimedia information are only in the earliest stages of development but absolutely required for users to retrieve the information that they need.

• In the final analysis, a world of electronic multimedia documents dispersed across global networks will create a virtually unlimited market for intelligent software that organizes, adapts, searches and retrieves the right information on demand. Companies that succeed in this market will be "understanding" businesses that add value by bridging the gap between data and knowledge, making information meaningful and useful.

OCR Voting Overview

Kenn T. Dahl
Prime Recognition
Four Buttercup Street
San Carlos, CA 94070
415-637-8382

Abstract

Voting OCR is a relatively new OCR technology which generates 65-80% fewer errors than conventional OCR products.

Voting OCR is slower than conventional OCR but imaging systems may achieve any necessary throughput by implementing multiple voting OCR units in parallel. Voting OCR is more expensive than conventional OCR on a per character recognized basis but leads to substantially lower costs for the full imaging system as it reduces the need for OCR error correction resources.

Voting OCR's lower error rate and superior error marking leads to cleaner data flowing into the application's database.

A number of misconceptions have surfaced concerning voting OCR. These are addressed in the text. Finally, expected developments in voting OCR technology are discussed.

1 Description of OCR Voting

Conventional OCR engines from the leading vendors use different algorithms to perform OCR. Hence the errors from one engine often do not match the errors from another engine.

Voting OCR is the process of recognizing an image with multiple conventional OCR engines, and then correlating, and voting the results.

The correlation of multiple engine results is surprisingly difficult as the engines may disagree on whole lines, words, or multiple character sequences. The voting process may be as simple as a majority vote but voting OCR vendors have implemented much more sophisticated voting strategies.

The voting results have a lower error rate than any one conventional OCR engine. Figure 1 illustrates the error rate of a voting OCR engine vs. a conventional OCR engine as a function of the number of voting engines. As more engines are included voting error reduction increases, however, there are diminishing returns after 3 engines. (Appendix A very briefly describes key aspects of the voting engine, images, and test data used in all figures in this report. Note that all test data and most voting OCR products are currently machine print only.)

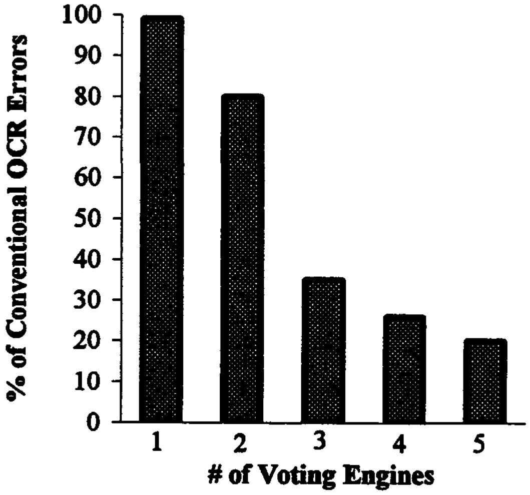


Figure 1: Voting OCR Error Reduction.

However, voting does not guarantee better accuracy on each document. Figure 2 indicates the statistical distribution of voting OCR (3 engines) vs. one conventional OCR engine. The average error reduction indicated by this test was 65% fewer errors for voting OCR, however, there were a small number of documents where voting OCR was worse than that specific conventional OCR engine. If, on the other hand, voting OCR was compared to an average of the leading conventional OCR engines, then it would be extremely unlikely for voting OCR to cause more errors than the "average" conventional engine.

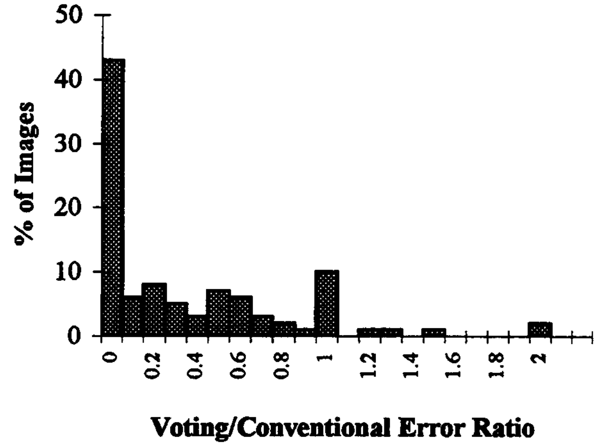


Figure 2: Voting OCR Error Reduction Distribution.

A properly designed OCR engine takes advantage of an OCR engine's strengths and minimizes its weaknesses, hence voting OCR results exhibit much lower variation in performance than any one conventional engine. This behavior is illustrated in Figure 3.

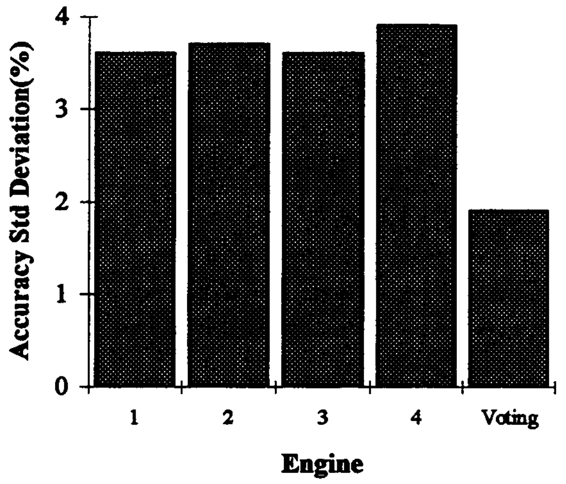


Figure 3: OCR Error Variability.

In addition to lower error rates voting OCR is able to identify suspicious characters more accurately than conventional OCR. Figure 4 illustrates the performance of the leading conventional OCR engine vs. a voting engine. For any chosen percentage of characters marked voting OCR marks a higher percentage of its errors. Voting OCR's performance is all the more impressive considering that it is performing better error marking on a much smaller base of errors.

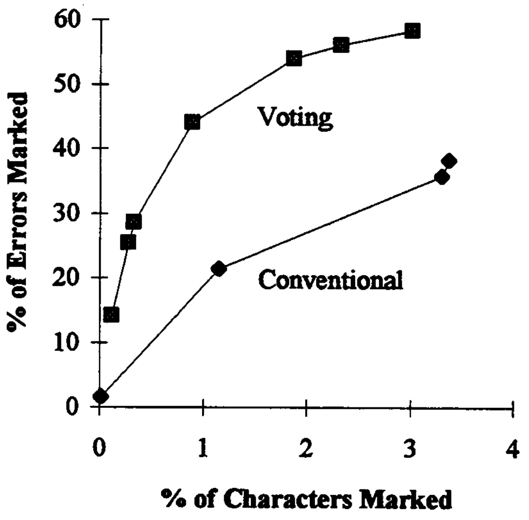


Figure 4: Error Marking Efficiency.

2 Apparent Disadvantages

Voting is by definition slower than conventional OCR, since it must run multiple conventional engines for equivalent output. However, most reasonably sophisticated imaging systems already support multiple OCR units, in parallel, to allow high throughput. Voting OCR merely requires more units. Therefore there is no technical barrier to achieving high throughput with voting OCR.

Voting OCR is significantly more expensive than conventional OCR when measured on a $ per recognized character basis. A shortsighted analysis of voting's price might conclude that voting is much more expensive than conventional OCR. However, as will be shown in Section 2.1, voting's higher price is counterbalanced by other cost savings in the full imaging system. Voting OCR is actually less expensive than conventional OCR in most applications when costs are measured at the system level.

3 Benefits

The two main benefits of voting OCR are cost savings and "cleaner" data.

3.1 Cost Savings

Most imaging systems employ manual labor operating on workstations to verify and correct OCR errors. The cost of manual OCR correction is much higher than the cost of OCR processing as shown in Figure 5.

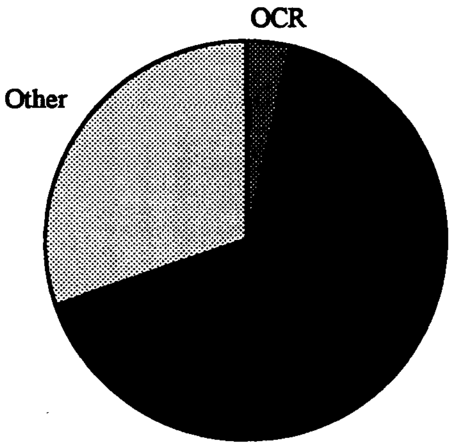


Figure 5: Imaging System Costs.
(99.8% Accuracy, Full Text)

Voting OCR reduces the need for error correction workstations, the personnel sitting at these workstations, as well as related assets such as real estate and managerial resources.

An aggressively priced voting OCR engine therefore requires no extra capital cost. The higher costs of OCR equipment are counterbalanced by the reduced costs of error correction workstations. Voting OCR also saves 65% of the ongoing manual labor error correction cost as illustrated in Figure 6.

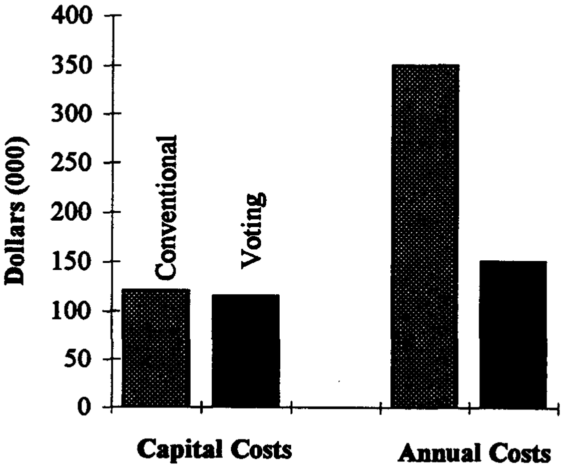


Figure 6: Imaging System OCR Costs.

3.2 "Cleaner Data"

Manual error correction only verifies and corrects errors that have been marked as suspicious by the OCR engine. The leading conventional OCR engine only marks 40-50% of its errors, hence, 50-60% of the errors flow through into the application's database.

Voting OCR generates fewer errors and does a better job of marking its errors as suspicious, hence 70-80% fewer errors flow through into the application's database.

For many applications, financial being the most obvious, the cost of incorrect data is extremely high.

4 Current Issues

Voting OCR has become a "hot" topic within the OCR community hence a number of issues have been raised by competitive vendors, potential users, etc..

4.1 Word Vs. Character Accuracy

One widely circulated concern is that voting OCR may improve character accuracy but it might not improve word accuracy. In other words, voting is able to remove errors in words with multiple errors but is not able to remove the last error from a word.

This is an important concern, if true, because error verification costs are more correlated to word errors than character errors.

Actually the opposite is true, voting OCR is able to more easily correlate and vote one single error in a word than multiple errors in the word. Voting OCR has slightly higher word accuracy than character accuracy.

4.2 Rapid Improvement In Conventional OCR Will Close The Gap

Some observers estimate that conventional OCR is improving at very rapid rates. They point to the ISRI/UNLV statement that leading engines improved at a 28% rate in 1995.[1] They also use the WordScan 4.0 release as an example, which, it is claimed, increased in accuracy by 40%.

They therefore conclude that if they wait two years the 65% error reduction advantage of voting OCR will be matched by conventional OCR.

There are two obvious flaws with this argument. Most importantly, voting OCR is not static. A good voting OCR engine always integrates the best and latest engines from the leading OCR vendors. As the accuracy of these engines goes up voting theory and practice indicates that voting actually generates higher error reductions than in the past.

Finally, the actual annual increase in accuracy for the leading OCR engines is significantly smaller than noted above. It took WordScan more than two years to upgrade its engine, therefore the implied annual improvement rate is significantly less than 20%. ISRI/UNLV's remarks were based on accuracy improvements on their magazine sample between 1994 and 1995. However, 1994 was the first year for the magazine sample test. It is well known that the inclusion of the magazine sample in 1994 was a surprise to vendors and many performed poorly, because, among other reasons, they had not optimized their engines for magazine type text. A significant part of the improvement in 1995 was due to the vendors expecting and optimizing their engines for a magazine sample.

4.3 Low Quality Images Render Voting Irrelevant

ISRI/UNLV's testing has shown that voting OCR accuracy improvement declines in low quality images. ISRI/UNLV also notes that most errors occur in low quality images.[2]

Some observers thus conclude that voting's average error reduction ratio is actually much lower than that claimed by vendors and therefore voting is not worth its higher cost.

In practice most imaging systems weed out poor quality images for key entry instead of OCR. Images that OCR at an accuracy of less than 95% should be manually entered according to an industry rule of thumb.

Very poor quality images are not relevant for OCR tests because they are not currently being recognized using conventional OCR and they greatly skew accuracy results. Consider the following extreme example:

100 images with 1000 characters per image

-98 images are good quality

-2 images are unrecognizable

Conventional OCR generates 20 errors on each good quality image, and 1000 errors per image on the poor quality images. Voting OCR generates 6 errors on each good quality image and 1000 errors on the poor quality images.

In a typical imaging system the poor quality images would be selected for key entry before getting to the OCR process. If the poor images did get to the OCR process their poor recognition results would flag them for key entry.

Table 1: Impact of Bad Quality Images on OCR Statistics.

	Conventional OCR Errors	Voting OCR Errors
Good Quality	1960	588
Reject Quality	2000	2000
Total	3960	2588

If voting OCR's error reduction rate is measured on all images it is 35%, however, if it is measured on those images that are actually being recognized in practice today, it is 70%.

Voting theory suggests that voting accuracy does decline with image quality, however, the actual rate of decline is dependent on the distribution of image defects. Figure 7 illustrates the decline in error reduction performance of a voting engine as a function of image quality. The decline is not nearly as dramatic as that noted by ISRI/UNLV. The differences may be due to the distribution of errors in the respective images and/or a difference in voting engine implementation.

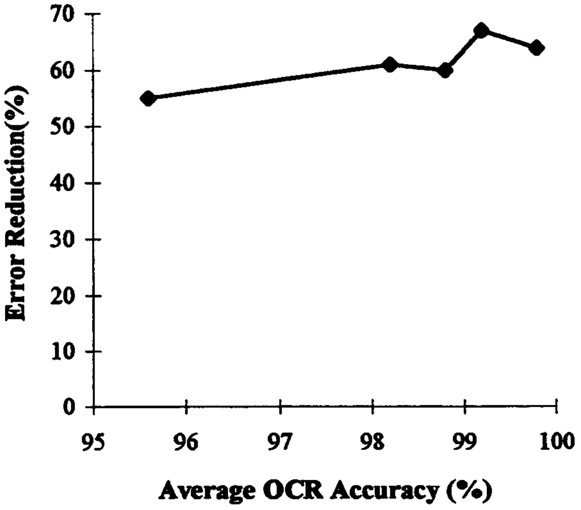


Figure 7: Voting OCR Error Reduction Declines with Image Quality.

4.4 Many New "Voting" Engines Are Not True "Voting" Engines

As "voting" has become a "sexy" marketing term a number of companies have introduced "voting" engines. Almost all of these new engines vote only a small portion of the recognition process, typically the character recognition process. They do not vote the character segmentation process, the page segmentation process, the lexical check process, etc.

The author is only aware of three companies which market a full "voting" machine print OCR engine: Prime Recognition, TRW, and SAIC.

The best way to tell the difference between a true voting engine and a "marketing" voting engine is to test their accuracy. A "marketing" voting engine will achieve only a marginal improvement in accuracy over conventional OCR.

4.5 Is Voting OCR Important In Fuzzy Search Applications?

Fuzzy search is the ability to search data that has errors in it, and yet still find the relevant data. Fuzzy search vendors therefore claim that expensive manual OCR error correction is not required, and some users generalize that the accuracy improvement of voting OCR is not important in fuzzy search applications.

Fuzzy search does still depend on the accuracy of OCR. A fuzzy search cannot find "Patriot" if it has been OCR'd as "Deina" or "...ot". Even if an extremely "loose" fuzzy search does find "...ot" as a match with "Patriot" it will put this match way down on the "hit" list.

A very small database, with high quality images, and a low value application may not benefit from voting OCR. However, a database with more than 1000 images, and/or average quality images, and/or a high value application, would benefit greatly from voting OCR.

Voting OCR is even more important in an environment in which OCR errors do not get corrected. More so than before the accuracy of the OCR engine will determine the success of searching the database. Voting OCR's value in a fuzzy search environment boils down to a risk judgment. How important is it that data be accurately and fully retrieved by a search?

5 Applications Not Suited to Voting

There are applications in which voting is not a good choice:

* The application already has high OCR accuracy due to:
  - -A single font environment which allowed custom OCR development
  - -Extremely high redundancy in data allowing custom post processing development
  - -Low accuracy requirement of application and very high quality images

* The application is single font and one conventional OCR engine is very strong on this font, e.g., WordScan on 9 pin dot matrix.

6 Future Directions

Voting engines have traditionally been "omnifont", that is, they could work "off the shelf" in almost any environment. Voting engine architectures will be extended to allow customization of the voting engine for particular applications including single font applications:

-Selection of which engines to run and the "weight" of each engine in voting

-Training on a single font

-Inclusion of user specified/developed engines

Image processing algorithms will be used to preprocess images, in particular to connect broken characters such as 9 pin dot matrix.

Voting engine output will be extended past conventional OCR engines, to include, for example, identification of 2nd and 3rd choice characters.

Voting algorithms will increase in sophistication and accuracy.

Machine print OCR vendors will extend their technology to hand print.

References

[1] S.V. Rice, F.R. Jenkins, and T. A. Nartker, The Fourth Annual Test of OCR Accuracy, *UNLV Information Science Research Institute 1995 Annual Report* (1995) 11-51.

[2] S.V. Rice, J. Kanai, and T. A. Nartker, The Third Annual Test of OCR Accuracy, *UNLV Information Science Research Institute 1994 Annual Report* (1994) 11-40.

Appendix A

All test data noted in this report are based on PrimeOCR, the base configuration. PrimeOCR is a machine print voting OCR engine from Prime Recognition. The base configuration includes three conventional OCR engines. Four and five engine versions of PrimeOCR are available, these versions would yield better results.

The test data was based on 168 images which included 542,738 characters. One hundred and twenty six (126) of these images were from the ISRI/UNLV database as distributed on the Univ. of Washington CD-ROM, with the remaining images selected from the Prime Recognition database.

The ISRI/UNLV database images were predominately technical reports and published technical documents. Prime Recognition's images included fax, magazine, business memos, newspaper, and other types of images.

The test database was constructed to include a large number of images that were available in the public domain (ISRI/UNLV images) plus a cross section of other document types. It is not intended as an exhaustive and statistically balanced database, its purpose is merely to illustrate voting OCR performance.

Extracting Tables from Printed Documents

*Hassan Alam, *, BCL Computers, 155A Moffett Park Dr., Suite 104, Sunnyvale, CA 94089*
*C. Hwa Chang, ** Zhongwen Shi, ** Scott Tupaj**, Tufts University, Medford, MA*

**hassan@shell.portal.com, **{hchang, zshi, stupaj}@ee.tufts.edu*

Abstract

This paper presents BCL's work in locating and extracting tables from page images. The method developed by BCL and Tufts University locates and extracts table cells, titles, and footers from financial and technical publications. It also determines headers for columns and arithmetic relationships in table data. Initial experiments produced promising results. The work is done under ARPA Contract #DAAH01-94-C-R156.

1. Introduction

With the wide co-existence of tabular information on paper and electronic documents, today companies and organizations need to capture printed tabular data and enter it into a structured electronic analysis tool such as a spreadsheet, so that the tabular data can be analyzed, manipulated, and re-used in other applications requiring this data. While optical character recognition (OCR) captures a document's text, and now, increasingly its format, the tabular data is not captured in a format suitable for easy computer-based analysis. The table needs to be entered manually by visually analyzing the document, recreating the table's structure, and keying the data into the document.

BCL and Tufts University are developing a technique for automatically locating tables in printed documents, capturing their format, and extracting and placing them in spreadsheets for further analysis. Initial results have shown high precision and recall rates. This paper presents an overview of our method and our results.

The paper is divided into five major parts. Section 2 describes our research goals. Section 3 gives a background on the types of tables we examined, Section 4 describes our method, Section 5 describes the results, and Section 6 describes areas of future work.

2. Goals

A number of research efforts have examined methods for analyzing tables in printed documents. These efforts have focused on different parts of the whole problem. Some have focused on extracting tables in specific domains [1,5], some have assumed that the table is already isolated and extracted [1,3,5], and others have examined the semantics of tables [2]. BCL has taken a two-step approach, first locating and extracting generalized tables from page images, and second understanding the table semantics.

This paper presents the first part of our research for locating and extracting generalized tables from page images. By selecting generalized tables, we developed algorithms for table analysis in any domain without any previous knowledge or constraints.

3. Overview of Tables

While humans easily grasp the concept of tables, our examination of tables in printed documents showed a large variation in the way tables are presented. To set the context for table analysis, we first present a brief survey of tables in printed documents. We first look at the information presented in tables. In this we differentiate between tables and forms and look at information in two different domains, technical and financial. We then look at the format of tables in printed matter. In this format we examine the location of tables in documents and the different methods of formatting tables.

3.1 Tables vs. Forms

In defining tables we included all representations of data where two indices were required to uniquely identify any element in the table. This allowed representation of the data in rows and columns. However, it does not include forms where many data elements need only one unique identifier for each element, obviating the need for tabular representation. Data where there was one index for identifying an element was viewed as a vector or a row of a table.

3.2 Table Domains

To develop a broad, robust table analysis system we selected tables from two distinct domains, scientific journal articles for technical tables and financial reports for financial tables.

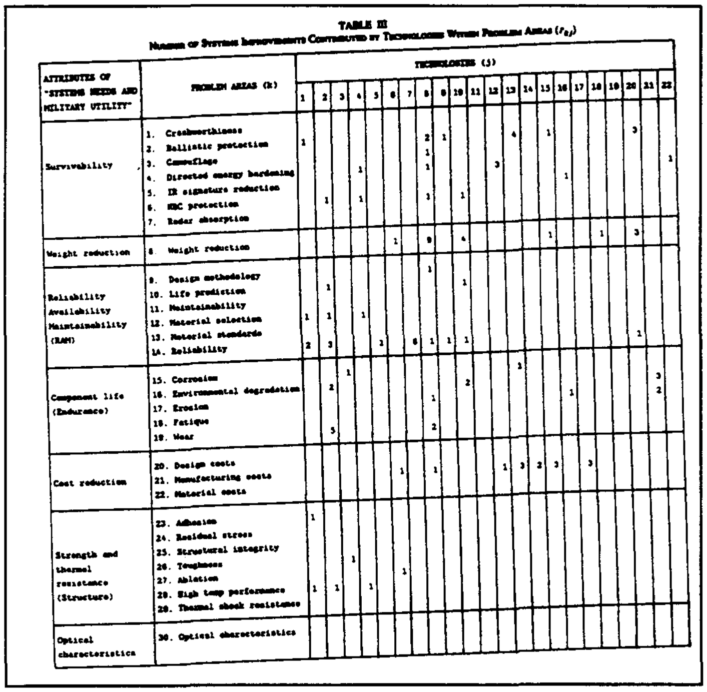


TABLE III
NUMBER OF SYSTEMS IMPROVEMENTS CONTRIBUTED BY TECHNOLOGIES WITHIN PROBLEM AREAS (r_{kj})

ATTRIBUTES OF "SYSTEMS NEEDS AND MILITARY UTILITY"	PROBLEM AREAS (k)	TECHNOLOGIES (j)																					
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Survivability	1. Crashworthiness 2. Ballistic protection 3. Camouflage 4. Directed energy hardening 5. IR signature reduction 6. NBC protection 7. Radar absorption	1	1		1 1				2 1 1 1	1	1		3	4		1	1				3		1
Weight reduction	8. Weight reduction						1		9		4					1			1		3		
Reliability Availability Maintainability (RAM)	9. Design methodology 10. Life prediction 11. Maintainability 12. Material selection 13. Material standards 14. Reliability	1 2	1 1 3		1	1		6	1 1	1	1 1										1		
Component life (Endurance)	15. Corrosion 16. Environmental degradation 17. Erosion 18. Fatique 19. Wear		2 5	1					1 2		2			1			1					3 2	
Cost reduction	20. Design costs 21. Manufacturing costs 22. Material costs						1		1				1	3	2	3		3					
Strength and thermal resistance (Structure)	23. Adhesion 24. Residual stress 25. Structural integrity 26. Toughness 27. Ablation 28. High temp performance 29. Thermal shock resistance	1 1	1	1	1		1																
Optical characteristics	30. Optical characteristics																						

Figure 1 Table with multiple entries per cell

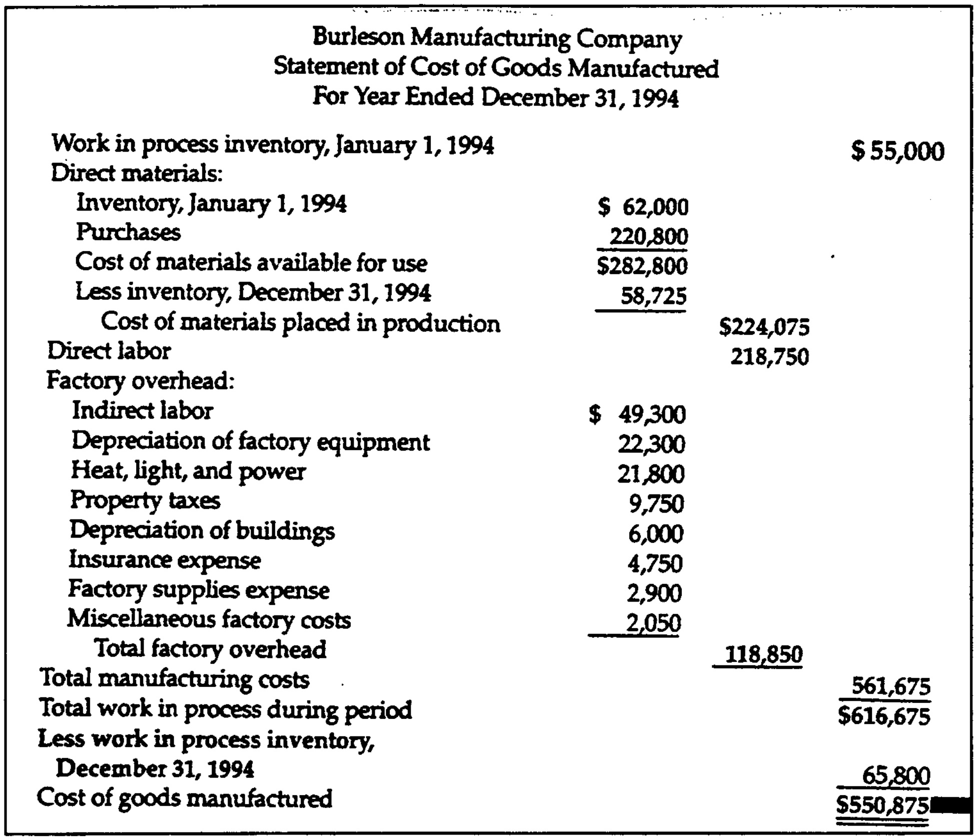
3.2.1 Technical Tables

We define technical tables as those presenting data in scientific and technical journals. The data types include text and numeric data. In addition to standard numeric data, there is extensive use of specific measuring units such as distance, energy, time, or units specific to an experiment.

Except for Figure 2, all figures show examples of technical tables. Often there is semantic information embedded in the table that allows multiple entries per cell, as well as tables embedded inside tables. Figure 1, shows an example of a table with multiple entries per cell. Syntactic cell analysis is required to understand this table's semantics.

3.2.2 Financial Tables

Companies and organizations use financial tables to convey financial results for companies, mutual funds, and other investment and financial related institutions. Tables include balance sheets, income statements, and annual performance results. The data types include dates, currency, percentage, and text. Arithmetic calculation in tables often includes sums and ratios (percentages). Figure 2 shows an example of a financial table.



Burleson Manufacturing Company
Statement of Cost of Goods Manufactured
For Year Ended December 31, 1994

Work in process inventory, January 1, 1994			$ 55,000
Direct materials:			
Inventory, January 1, 1994	$ 62,000		
Purchases	220,800		
Cost of materials available for use	$282,800		
Less inventory, December 31, 1994	58,725		
Cost of materials placed in production		$224,075	
Direct labor		218,750	
Factory overhead:			
Indirect labor	$ 49,300		
Depreciation of factory equipment	22,300		
Heat, light, and power	21,800		
Property taxes	9,750		
Depreciation of buildings	6,000		
Insurance expense	4,750		
Factory supplies expense	2,900		
Miscellaneous factory costs	2,050		
Total factory overhead		118,850	
Total manufacturing costs			561,675
Total work in process during period			$616,675
Less work in process inventory, December 31, 1994			65,800
Cost of goods manufactured			$550,875

Figure 2 Financial Table

3.3 Table Location

The first task in table analysis is locating a table on a page with other text and graphics. Locating a table in a page involves differentiating the tables from other elements such as body text, headings titles, bibliography, lists, author listing, abstracts, line drawings, and bit map graphics. Figure 3 shows an example of a table embedded in a document containing some of the other elements described.

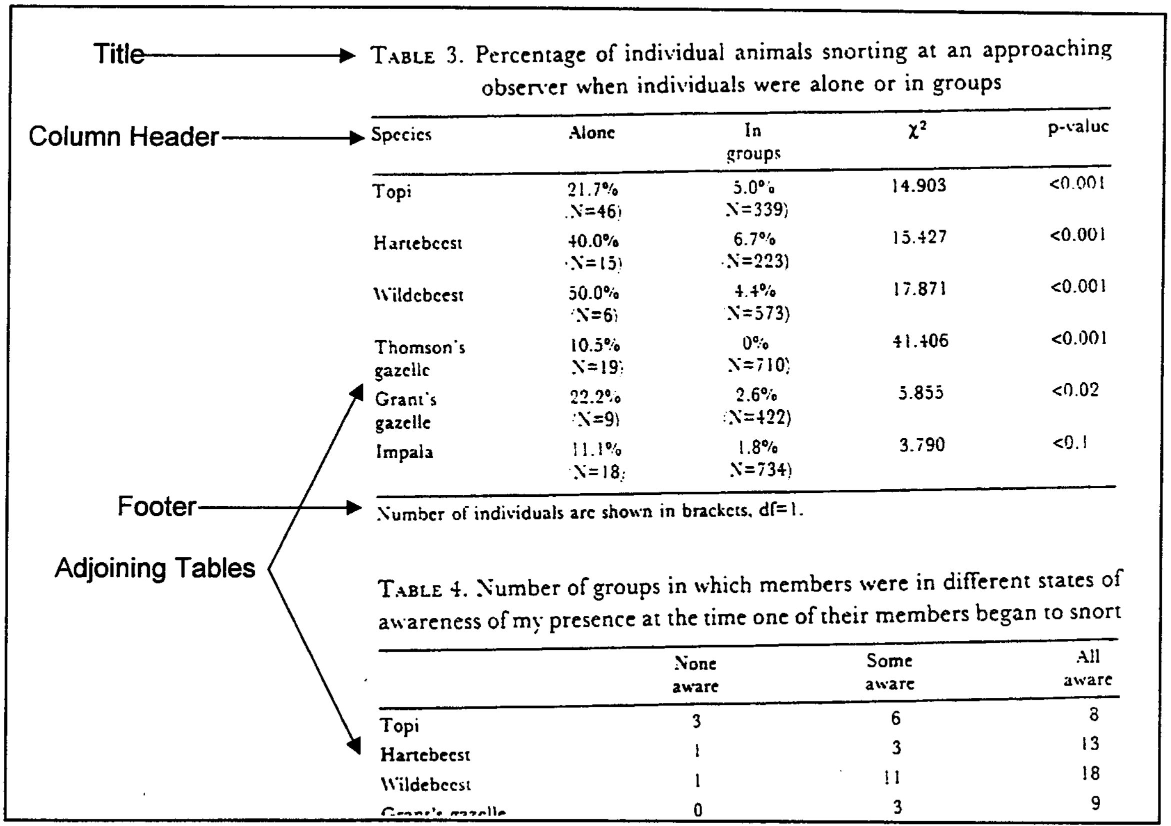

Title
Column Header
Footer
Adjoining Tables

TABLE 3. Percentage of individual animals snorting at an approaching observer when individuals were alone or in groups

Species	Alone	In groups	χ^2	p-value
Topi	21.7% (N=46)	5.0% (N=339)	14.903	<0.001
Hartebeest	40.0% (N=15)	6.7% (N=223)	15.427	<0.001
Wildebeest	50.0% (N=6)	4.4% (N=573)	17.871	<0.001
Thomson's gazelle	10.5% (N=19)	0% (N=710)	41.406	<0.001
Grant's gazelle	22.2% (N=9)	2.6% (N=422)	5.855	<0.02
Impala	11.1% (N=18)	1.8% (N=734)	3.790	<0.1

Number of individuals are shown in brackets, df=1.

TABLE 4. Number of groups in which members were in different states of awareness of my presence at the time one of their members began to snort

	None aware	Some aware	All aware
Topi	3	6	8
Hartebeest	1	3	13
Wildebeest	1	11	18
Grant's gazelle	0	3	9

Figure 3 Table headers and footers

Variations on table formatting include tables embedded on two-column page (Figures 4 and 5), multiple tables on a page (Figures 3,4,5, and 6), and pages with tables only (Figure 6).

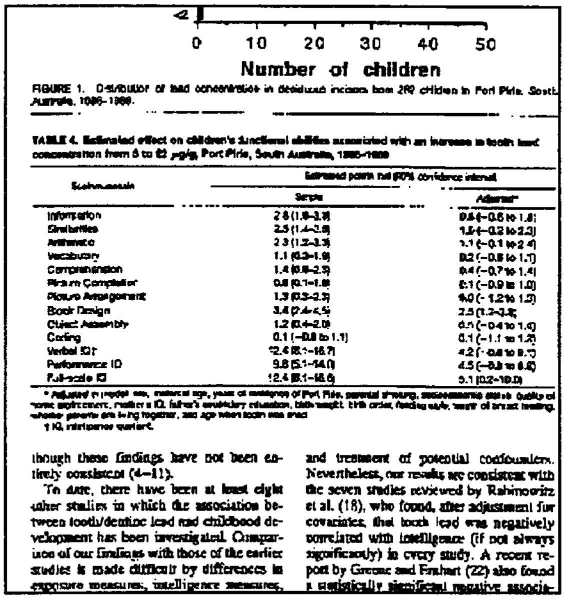


FIGURE 1. Distribution of lead concentration in deciduous incisors from [illegible] children in Port Pirie, South Australia, 1986–1989.

TABLE 4. Estimated effect on children's functional abilities associated with an increase in tooth lead concentration from 5 to 22 μg/g, Port Pirie, South Australia, 1986–1989

Scale/subscale	Estimated points lost (95% confidence interval)	
	Simple	Adjusted*
Information	2.8 (1.8–3.8)	0.6 (−0.6 to 1.8)
Similarities	2.5 (1.4–3.6)	1.0 (−0.2 to 2.3)
Arithmetic	2.3 (1.2–3.3)	1.1 (−0.1 to 2.4)
Vocabulary	1.1 (0.3–1.9)	0.2 (−0.8 to 1.1)
Comprehension	1.4 (0.5–2.3)	0.4 (−0.7 to 1.4)
Picture Completion	0.8 (0.1–1.6)	0.1 (−0.9 to 1.0)
Picture Arrangement	1.3 (0.3–2.3)	0.0 (−1.2 to 1.2)
Block Design	3.4 (2.4–4.5)	2.5 (1.3–3.8)
Object Assembly	1.2 (0.4–2.0)	0.5 (−0.4 to 1.4)
Coding	0.1 (−0.8 to 1.1)	0.1 (−1.1 to 1.2)
Verbal IQ†	12.4 (8.1–16.7)	4.2 (−0.8 to 9.1)
Performance IQ	9.8 (5.1–14.0)	4.5 (−0.8 to 9.8)
Full-scale IQ	12.4 (8.1–16.6)	5.1 (0.2–10.0)

* Adjusted in model: sex, maternal age, years of residence in Port Pirie, parental smoking, socioeconomic status, quality of home environment, mother's IQ, father's secondary education, birthweight, birth order, feeding style, length of breast feeding, whether parents are living together, and age when tooth was shed.

† IQ, intelligence quotient.

though these findings have not been entirely consistent (4–11).

To date, there have been at least eight other studies in which the association between tooth/dentine lead and childhood development has been investigated. Comparison of our findings with those of the earlier studies is made difficult by differences in exposure measures, intelligence measures, and treatment of potential confounders. Nevertheless, our results are consistent with the seven studies reviewed by Rabinowitz et al. (18), who found, after adjustment for covariates, that tooth lead was negatively correlated with intelligence (if not always significantly) in every study. A recent report by Greene and Ernhart (22) also found a statistically significant negative associa-

Figure 4 Spanning Table in a Two-Column Page

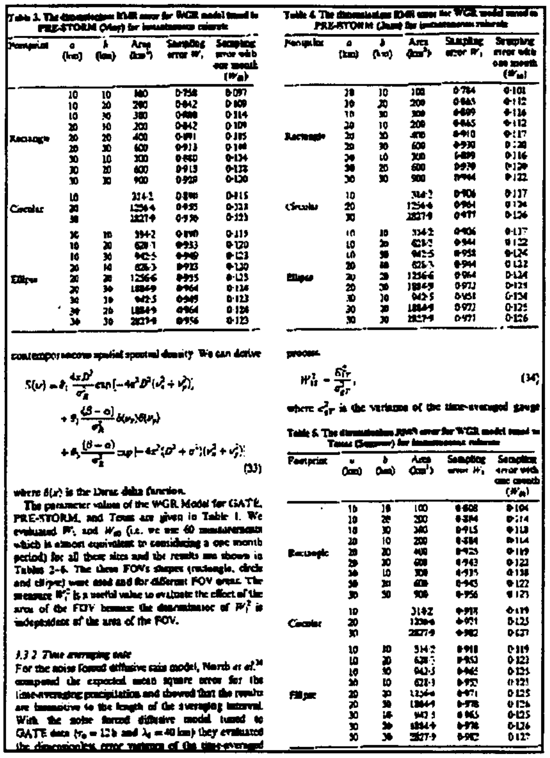


Figure 5 Multiple Tables in a Two-Column Page

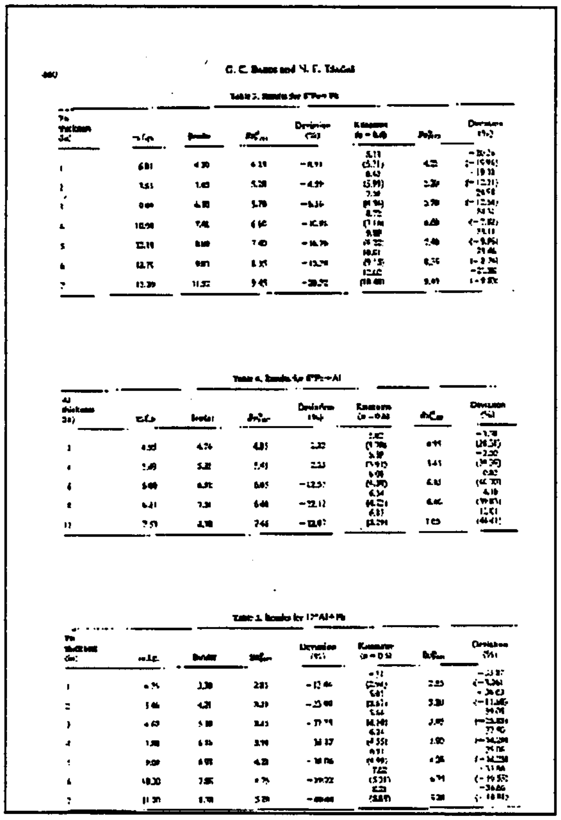


Figure 6 Tables Alone in Pages

Our goal was to develop table extraction methods that correctly find vertical and horizontal bounding boxes that isolate the table and its associated elements (titles, footers) from the surrounding document. In addition to identifying the main body of the table from other elements, the table extraction system also has to differentiate table titles and footers from other text elements and separate tables from adjoining vertical and horizontal tables. Figure 3 shows examples of table elements, figure 5 shows adjoining tables along with text and equations.

3.4 Types of Tables

In developing our table analysis system, our goal was to develop a system that would analyze a very large set of tables without any previous constraints on, or assumptions about the table format. Any table identified by a human reader as a table should be located and extracted. Variations we considered included tables that were completely bounded by lines, tables that had horizontal or vertical lines separating parts of the table, tables with no line boundaries, and tables that were partially empty.

3.4.1 Completely Bounded Tables (Closed Tables)

Completely bounded tables as shown in Figure 7 have each cell completely bounded by a line or combination of lines, allowing easy semantic interpretation of the table. However, without the lines, the tables may often have ambiguous structure, as cell separation is not always easily evident (Figure 1).

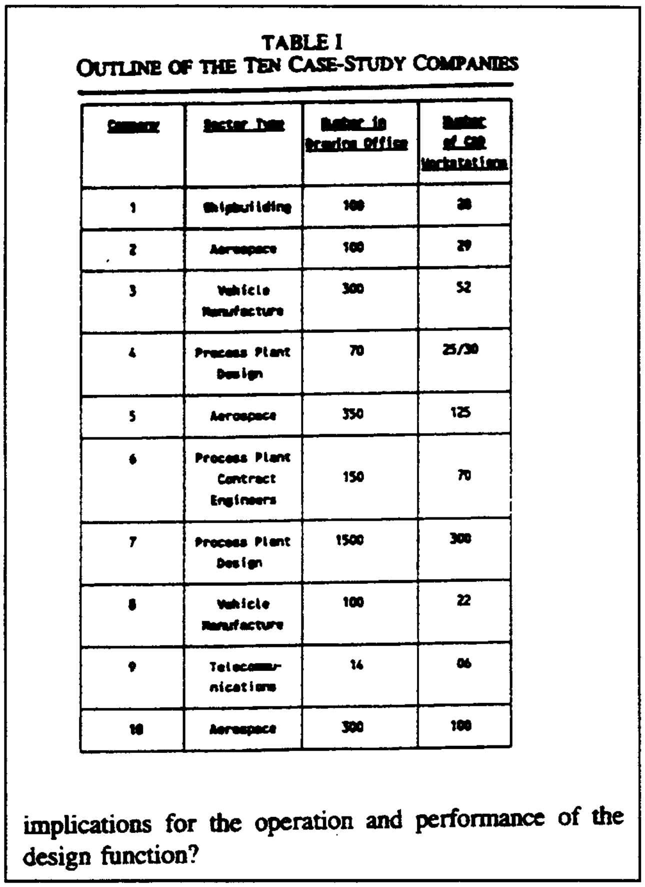


TABLE I
OUTLINE OF THE TEN CASE-STUDY COMPANIES

Company	Sector Type	Number in Drawing Office	Number of CAD Workstations
1	Shipbuilding	100	28
2	Aerospace	100	29
3	Vehicle Manufacture	300	52
4	Process Plant Design	70	25/30
5	Aerospace	350	125
6	Process Plant Contract Engineers	150	70
7	Process Plant Design	1500	300
8	Vehicle Manufacture	100	22
9	Telecommu- nications	14	06
10	Aerospace	300	100

implications for the operation and performance of the design function?

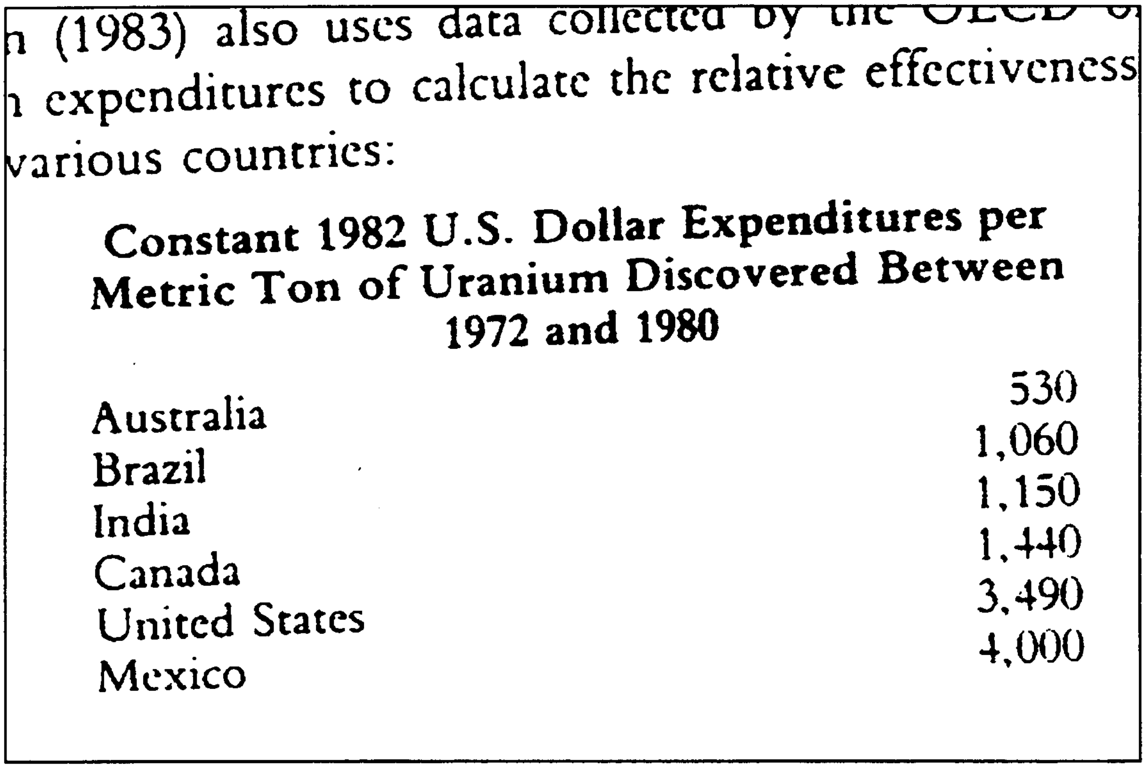
Figure 7 Completely Bounded Table

3.4.2 Partially Bounded Tables

Partially bounded tables have some lines separating parts of the table. These lines may bound the entire table but not individual cells, or have lines that separate the table components semantically. Figure 3 shows an example of a header separated from the table body by a set of lines. The header contains descriptions of the columns and may be of a different data type than the columns.

3.4.3 Unbounded Tables

Unbounded tables have no lines, and separation between rows and columns is done by a combination of white space and data type. The white space visually separates the columns, while data type semantically separates rows (e.g., header and body). An example is shown in Figure 8.



n (1983) also uses data collected by the OECD o
n expenditures to calculate the relative effectiveness
various countries:

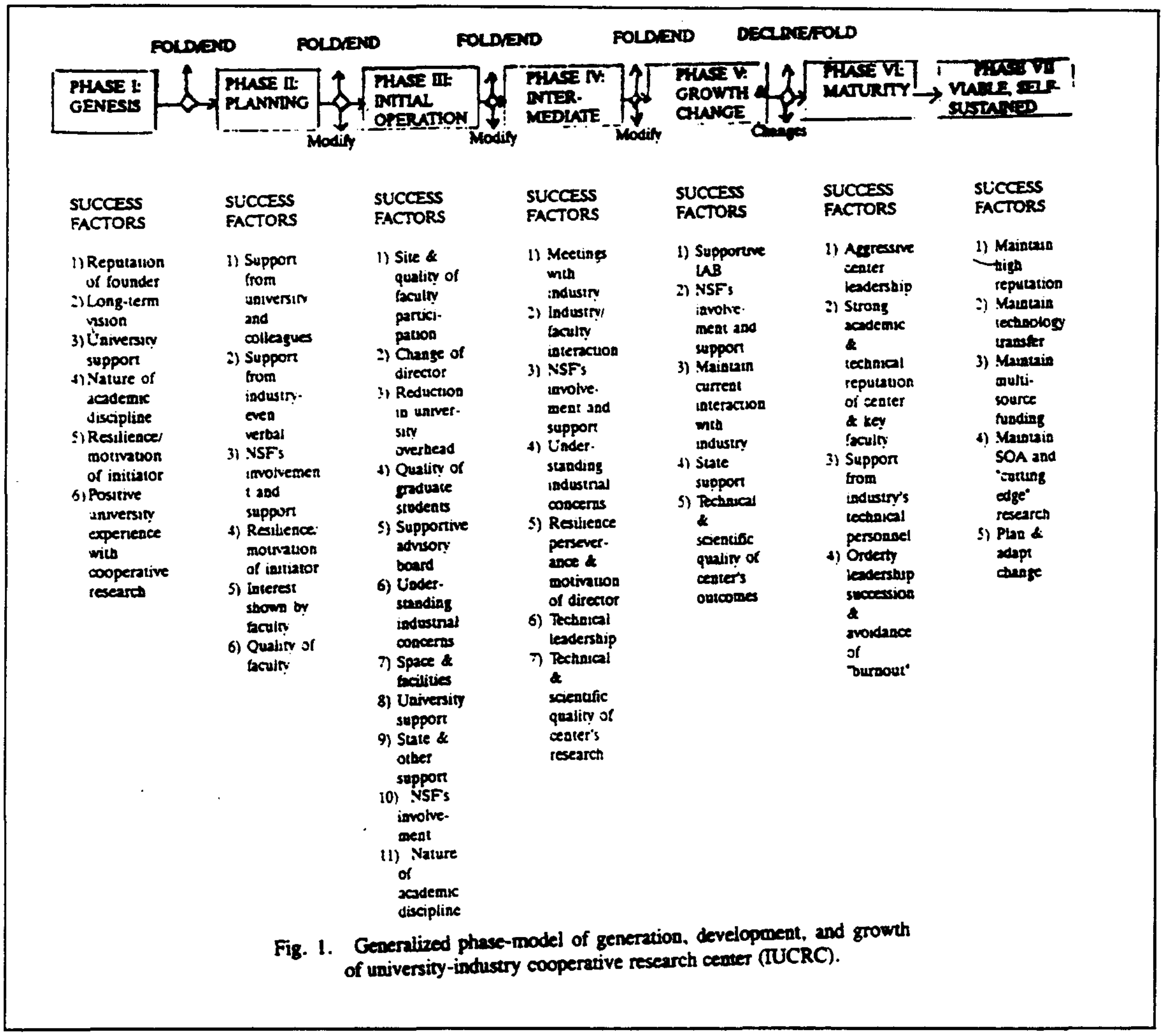
Constant 1982 U.S. Dollar Expenditures per Metric Ton of Uranium Discovered Between 1972 and 1980

Australia	530
Brazil	1,060
India	1,150
Canada	1,440
United States	3,490
Mexico	4,000

Figure 8 Unbounded Table

3.4.4 Partially Filled Tables

Partially filled tables provide an additional challenge in extracting tables because the table extraction system has to semantically differentiate column boundaries as well as table boundaries (Figure 9). In examining such tables we need to accurately determine where the table ends and what represents empty cells.



PHASE I: GENESIS	PHASE II: PLANNING	PHASE III: INITIAL OPERATION	PHASE IV: INTER-MEDIATE	PHASE V: GROWTH & CHANGE	PHASE VI: MATURITY	PHASE VII VIABLE, SELF-SUSTAINED
SUCCESS FACTORS	SUCCESS FACTORS	SUCCESS FACTORS	SUCCESS FACTORS	SUCCESS FACTORS	SUCCESS FACTORS	SUCCESS FACTORS
1) Reputation of founder 2) Long-term vision 3) University support 4) Nature of academic discipline 5) Resilience/ motivation of initiator 6) Positive university experience with cooperative research	1) Support from university and colleagues 2) Support from industry- even verbal 3) NSF's involvement and support 4) Resilience: motivation of initiator 5) Interest shown by faculty 6) Quality of faculty	1) Site & quality of faculty participation 2) Change of director 3) Reduction in university overhead 4) Quality of graduate students 5) Supportive advisory board 6) Understanding industrial concerns 7) Space & facilities 8) University support 9) State & other support 10) NSF's involvement 11) Nature of academic discipline	1) Meetings with industry 2) Industry/ faculty interaction 3) NSF's involvement and support 4) Understanding industrial concerns 5) Resilience perseverance & motivation of director 6) Technical leadership 7) Technical & scientific quality of center's research	1) Supportive IAB 2) NSF's involvement and support 3) Maintain current interaction with industry 4) State support 5) Technical & scientific quality of center's outcomes	1) Aggressive center leadership 2) Strong academic & technical reputation of center & key faculty 3) Support from industry's technical personnel 4) Orderly leadership succession & avoidance of "burnout"	1) Maintain high reputation 2) Maintain technology transfer 3) Maintain multi-source funding 4) Maintain SOA and "cutting edge" research 5) Plan & adapt change

Fig. 1. Generalized phase-model of generation, development, and growth of university-industry cooperative research center (IUCRC).

Figure 9 Partially Filled Table

4. Method

To analyze tables in printed documents, we divided the system up into seven steps: image preprocessing, closed table structure identification, document segmentation, zone classification, table geometry analysis, lexical analysis, and table understanding. These steps all consult a rule base as shown in Figure 10.

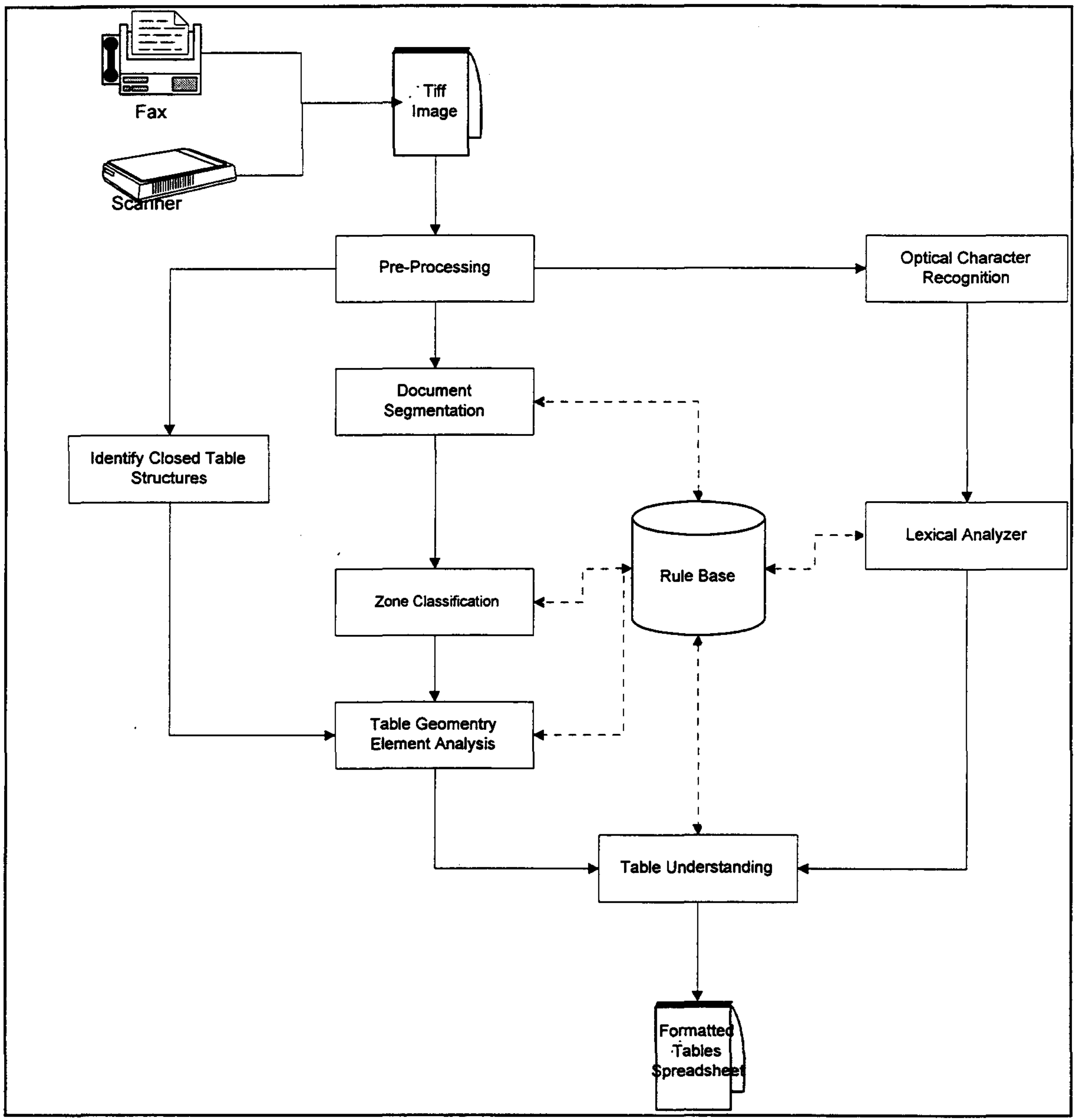
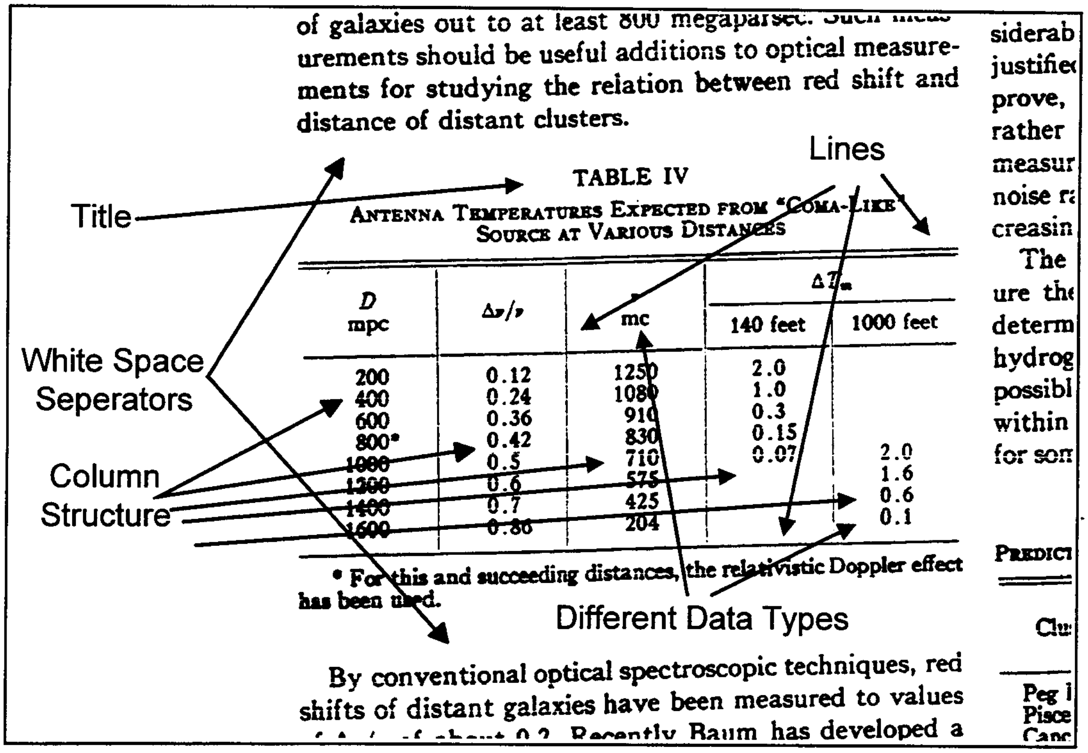

Fax
Tiff Image
Scanner
Pre-Processing
Optical Character Recognition
Document Segmentation
Identify Closed Table Structures
Rule Base
Lexical Analyzer
Zone Classification
Table Geomentry Element Analysis
Table Understanding
Formatted Tables Spreadsheet

Figure 10 Table Analysis Process

4.1 Rule Base

The rule base contains all the visual and lexical rules for locating, extracting, decomposing and understanding tables. These rules are based on human cognition techniques that detect column structure in tables and differentiate elements in a table based on content. The human cognition techniques are based on clues embedded in documents for table understanding. Figure 11 shows some of these clues. Other modules refer to this rule base to make decisions about the components they are examining.


Lines
Title
White Space Seperators
Column Structure
Different Data Types

of galaxies out to at least 800 megaparsec. Such measurements should be useful additions to optical measurements for studying the relation between red shift and distance of distant clusters.

TABLE IV

ANTENNA TEMPERATURES EXPECTED FROM "COMA-LIKE" SOURCE AT VARIOUS DISTANCES

D mpc	Δν/ν	mc	ΔTa 140 feet	ΔTa 1000 feet
200	0.12	1250	2.0	
400	0.24	1080	1.0	
600	0.36	910	0.3	
800*	0.42	830	0.15	
1000	0.5	710	0.07	2.0
1200	0.6	575		1.6
1400	0.7	425		0.6
1600	0.80	204		0.1

* For this and succeeding distances, the relativistic Doppler effect has been used.

By conventional optical spectroscopic techniques, red shifts of distant galaxies have been measured to values of about 0.2. Recently Baum has developed a

siderab
justifie
prove,
rather
measur
noise r
creasin
The
ure the
determ
hydrog
possibl
within
for son

PREDICT

Clu

Peg I
Pisce
Canc

Figure 11 Clues for Table Analysis

4.2 Pre-Processing

Typically a page containing a table is received with errors caused by imperfect scanning or faxing. Errors typically contain rotation, skewing, spurious lines, image degradation, noise and in the case of faxes, 180-degree rotation. Using standard image analysis techniques, we correct the image before we analyze a page for tables. Line detection and removal are applied to the page image. These lines are used for processing. They are used for isolating closed tables (line defined) within the page.

4.3 Page Segmentation

After image clean-up, the page is segmented using rules from the Rule Base to identify the structure and major components of the document. Segmentation is done using horizontal and vertical projection profiles and the extracted lines.

4.4 Classify Body-Text Zones

The zones from the segmentation process are classified as different types of document elements such as body text, graphics, headers, etc. The classification of zones is done using statistical methods based on properties of different elements.

4.5 Table Geometry and Element Analysis

Once the text zones are isolated, we perform connected component analysis to determine region where tables may lie. The components are examined for column structure that show likelihood of containing tables.

4.6 Lexical Analyzer

In parallel to the table geometry analysis, the lexical analyzer examines the OCR output to look for key words and phrases that indicate the existence of tables.

4.7 Table Understanding

The output of the table geometry analysis and the lexical analyzer are taken by the table understanding module to analyze the regions suspected of containing tables. This analysis extracts table elements such as rows, columns, titles, and footers. Once rows and columns are determined, they are broken down into cells and column headers are determined by lexical and geometric analysis. This table is then exported to a spreadsheet.

5. Results

Our work in table analysis can be measured in the domains of technical and financial tables. Each of these accuracies is measured by a precision and a recall. *Precision* measures the correct identification of elements, and *recall* measures the location of all the elements. The equations for measuring precision and recall are:

Recall (I) = number of type I objects identified accurately /total number of type I objects
Precision (I) = number of type I objects identified accurately/number of objects identified as type I

In calculating our results we examined 100 scientific journal pages with 132 tables and 4385 cells. We also examined 20 financial report pages with 40 tables and 1908 cells. The results are given in Figure 12.

	Technical Tables	Financial Tables
Table Precision	100.00	95.00
Table Recall	99.24	100.00
Cell Precision	97.10	92.05
Cell Recall	91.63	89.26

Figure 12 Results (%)

6. Future Work

A number of areas still need to be worked on. In technical tables we need to handle multiple entries per cell (Figure 1) and tables embedded in tables. We also need to improve detection and analysis of financial tables. These will be done through more emphasis on semantic analysis of tables by content examination, as well as better topological models describing the tables. Once the semantic analysis is done, the table will be converted to its surface form for uniform two-dimensional representation [2]. The semantic analysis is also dependent on the quality of optical character recognition. A number of efforts are underway in this area, and we expect to see improvements in the future.

7. Bibliography

1. Balasubramanian, S., Chandran S., Arian J., Kasturi R., and Chhabra, A.. Information Extraction from Tabular Drawings Proc. *SPIE.* Document Recognition Feb. 1-10, San Jose pp. 152-163.

2. Douglas, S., Hurst, M., Quinn, D. Using Natural Language Processing for Identifying and Interpreting Tables in Plain Text. *Fourth Annual Symposium on Document Analysis and Information Retrieval* pp. 535-546

3. Green, E. and Khrisnamoorty M. Model-Based Analysis of Printed Tables. *Proc. of the Third International Conference on Document Analysis and Recognition* Aug. 14-16, Montreal, Canada pp. 214-217

4. Hori, O. and Doermann, D.S. Robust-Table Form Structure Analysis Based on Box-Driven Reasoning. *Proc. of the Third International Conference on Document Analysis and Recognition* Aug. 14-16, Montreal, Canada pp. 218-221

5. Rahgozar, M.A., Fan Z., and Rainero, E. Tabular Document Recognition Proc. *SPIE.* Document Recognition Feb. 1-10, San Jose pp. 87-96.

6. Wang, S. Yagasaki, T. *Block Selection:* A Method for Segmenting Page Image of Various Edition Styles. *Proc. of the Third International Conference on Document Analysis and Recognition* Aug. 14-16, Montreal, Canada pp. 128-133

7. Watanabe,T. and Fukumura T. A Framework for Validating Recognized Results in Understanding Table-Form Document Images. Proc. of *the Third International Conference on Document Analysis and Recognition* Aug. 14-16, Montreal, Canada pp. 536-539.

Forms and Logos

Document Processing Research at Michigan State University

Anil K. Jain, Bin Yu, Yu Zhong, Øivind D. Trier and Nalini K. Ratha
Dept. of Computer Science
Michigan State University
East Lansing, MI48824
jain@cps.msu.edu

Abstract

This paper presents ongoing research work on document image processing in the Pattern Recognition and Image Processing Laboratory at Michigan State University. The research topics include document skew detection, language separation, page segmentation, localization of text in images, form dropout, map processing, and hardware implementation.

1 Introduction

Due to rapid development of low-cost computing and scanner technologies in recent years, document image analysis has received an increasing interest in both academic and industrial organizations. The goal of document image analysis is to enable a computer to automatically read and interpret paper-based data (e.g., text, graphs, maps). This involves a number of modules such as preprocessing, segmentation, optical character recognition, page structure analysis, symbol recognition and postprocessing. A number of commercial systems are available which have been successfully used in many application fields.

During the past four years, we have been working on a number of projects in the area of document image analysis. The first project deals with document skew detection which employs a hierarchical Hough transform to quickly detect the skew angle of various types of documents with arbitrary orientation. The second project is on language separation which attempts to separate texts of different language in an image by using their texture attributes. The third is page segmentation. Texture cue has also been used to hierarchically classify text, line-drawing, halftone, and background regions. The fourth project is on localization of text in complex color images. This has enabled us to extract the titles on book covers and compact disc covers. We have developed a form dropout algorithm which can extract filled-in characters even when they are written or typed so as to touch form frames. The sixth research project deals with map image processing. We have developed a map data capture method based on gray scale topographic analysis. To speed up the processing of document images, we have explored the use of custom computing machines based on FPGA technology.

2 Skew Detection

Many of the important document analysis algorithms, including optical character recognition (OCR) and page layout analysis, are sensitive to the orientation (or skew) of the input document image.

A generic document of interest can be a machine printed or handwritten text on paper or envelope, a page of technical article composed of text lines of different fonts, bar codes, drawings, figures and tables, a form and a photocopied document with black margins and noise.

Most previous skew detection methods work well for documents containing printed text only and fail in situations where the text lines are not in majority, e.g., in a generic document described above. One of the main disadvantages of these algorithms is their high computational cost which quickly increases as a function of the angular detection range.

The skew detection algorithm which we have developed [1] consists of two steps. The first step quickly extracts the centroids of connected components based on a graph data structure, called block adjacency graph (BAG) [2] and selects a few components based on the size of their bounding boxes. In the second step we detect the skew by a hierarchical Hough transform (HHT).

2.1 Proposed Algorithm

The minimum information needed to determine the document skew is one or more symbol lines, each consisting of several separately aligned symbols. If a group of symbols is closely aligned along a line then their centroids are also roughly aligned along a line with the same direction. The centroid, therefore, is regarded as a key feature for skew detection.

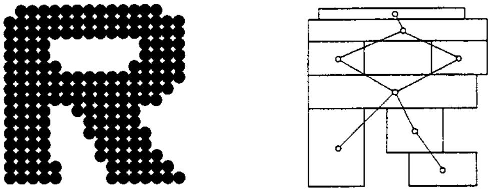


Figure 1: A connected component and its BAG.

Yu et al. [2] described the block adjacency graph (BAG) which can be created concurrently when the document is being scanned row by row. Figure 1 shows a connected component and its BAG which consists of eight connected blocks. Note that each block is an approximate bounding box of a partial area in the component and each area involves either one or a series of runs connected and closely justified in margin, given a prespecified tolerance. It is easy to obtain the centroid coordinates of the connected component by searching the graph and averaging the centroids of the blocks connected in the BAG. Furthermore, some connected components are filtered out with respect to the size of their bounding boxes, which are introduced due to noise, black margins, lines in tables or forms, figures, etc. Therefore, only useful data are fed to the Hough transform (HT).

For straight line detection, the HT is $\rho = x\cos\theta + y\sin\theta$, where a point (x, y) is converted to its polar coordinates (ρ, θ). Its computational cost is $\mathcal{O}(N\Psi)$, where N is the number of points in the physical space and Ψ is the number of different values of θ used in constructing the accumulator array. Note that $\Psi = \Delta\theta/\delta\theta$, where $\Delta\theta = (\theta_2 - \theta_1)$ is the desired angular range, $0^\circ \leq \theta_1, \theta_2 \leq 180^\circ$ and $\theta_1 < \theta_2$. The parameter $\delta\theta$ is the angular resolution which controls the accuracy of the skew detection (orientation of a line). The computational cost can be decreased by reducing $\Delta\theta$ or increasing the interval $\delta\theta$. We implement the HT twice. Initially, we select a relatively large value of $\delta\theta$, say, $\delta\theta_1$ to roughly determine the skew angle $\hat{\theta}$ in the desired angular range $\Delta\theta$. Then, we implement the HT again with the desired angular resolution $\delta\theta_0$ within the reduced range $(\hat{\theta} - \delta\theta_1, \hat{\theta} + \delta\theta_1), \delta\theta_0 < \delta\theta_1$. We call this method the hierarchical Hough transform. The computational complexity of HHT is $\mathcal{O}(N\Psi_1) + \mathcal{O}(N\Psi_2) = \mathcal{O}(N\Psi_H)$, where $\Psi_H = 2\delta\theta_1/\delta\theta_0 + \Delta\theta/\delta\theta_1$. Minimizing Ψ_H, we have $\delta\theta_1 = \sqrt{0.5\Delta\theta\delta\theta_0}$ and $\Psi_H = \sqrt{8\Delta\theta/\delta\theta_0}$. Traditionally, $\Psi_T = \Delta\theta/\delta\theta_0$, therefore, $\Psi_H/\Psi_T = \sqrt{8\delta\theta_0/\Delta\theta}$. If $\Delta\theta = 180^\circ$ and $\delta\theta_0 = 0.1^\circ$, we have $\delta\theta_1 = 3^\circ$ and $\Psi_H/\Psi_T = 7\%$. At the same time, the storage requirement which is proportional to Ψ is also reduced.

Table 1: Performance of the skew detection algorithm.

Img No.	Size (pixels)	Res. (dpi)	Time (s)	Estim. skew
1	268×225	50	0.1	25.1°
2	319×300	50	0.1	−23.8°
3	326×286	100	0.1	−36.9°
4	297×327	50	0.1	51.2°
5	415×596	50	0.4	−5.1°
6	238×268	50	0.1	−13.5°
7	422×598	50	0.5	4.5°
8	414×587	50	0.4	3.1°
9	412×576	50	0.2	−4.8°

The lines in the image with similar angles (θ) and different ρ should have the same contribution in skew detection. A two-dimensional window can be introduced for this purpose where all the accumulator cells inside the window with values larger than a threshold will be summed. The skew angle is the weighted average within that window where the peak is found; the weight values are based on the entries in the accumulator array. With this process, the detected peak will be more robust in situations where the document image has several different candidate directions due to a non-rigid skew or text/symbol lines with different orientations.

2.2 Experimental Results

Our skew detection algorithm has been applied to a large number of document images to test its capability in different applications. These images came from envelopes, journals, magazines, postal labels, yellow pages and coupons. Some of the input images and their corresponding deskewed images are shown in Fig.2. Table 1 gives the performance of the algorithm implemented on a SPARC20 CPU. In our experiments, we have used $\delta\theta_0 = 0.1^\circ$, $\Delta\theta = 180^\circ$ and, therefore, $\delta\theta_1 = 3^\circ$.

2.3 Summary

A robust and fast skew detection algorithm based on hierarchical Hough transform is proposed. It is capable of detecting the skew angle for various document images, including technical articles, postal labels, handwritten text, forms, drawings and bar codes. The algorithm is robust even when black-margins introduced during photocopying are present in the image and when the document is scanned at a low resolution of 50 dpi. The algorithm consists of two steps. In the first step, we quickly extract the centroids of connected components using a graph data structure. Then, a hierarchical Hough transform of selected centroids is computed. The skew angle corresponds to the location of the highest peak in a two-dimensional window in the Hough space. The performance of the algorithm is shown on a number of document images collected from various applications. The algorithm is not very sensitive to algorithmic parameters. For an A4 size document image scanned at 50 dpi (typically 413×575 pixels), our algorithm is able to detect the skew angle within the whole angular range $(0°, 180°)$ with an accuracy of 0.1° in 0.4 seconds of CPU time on a Sun Sparc 20 workstation.

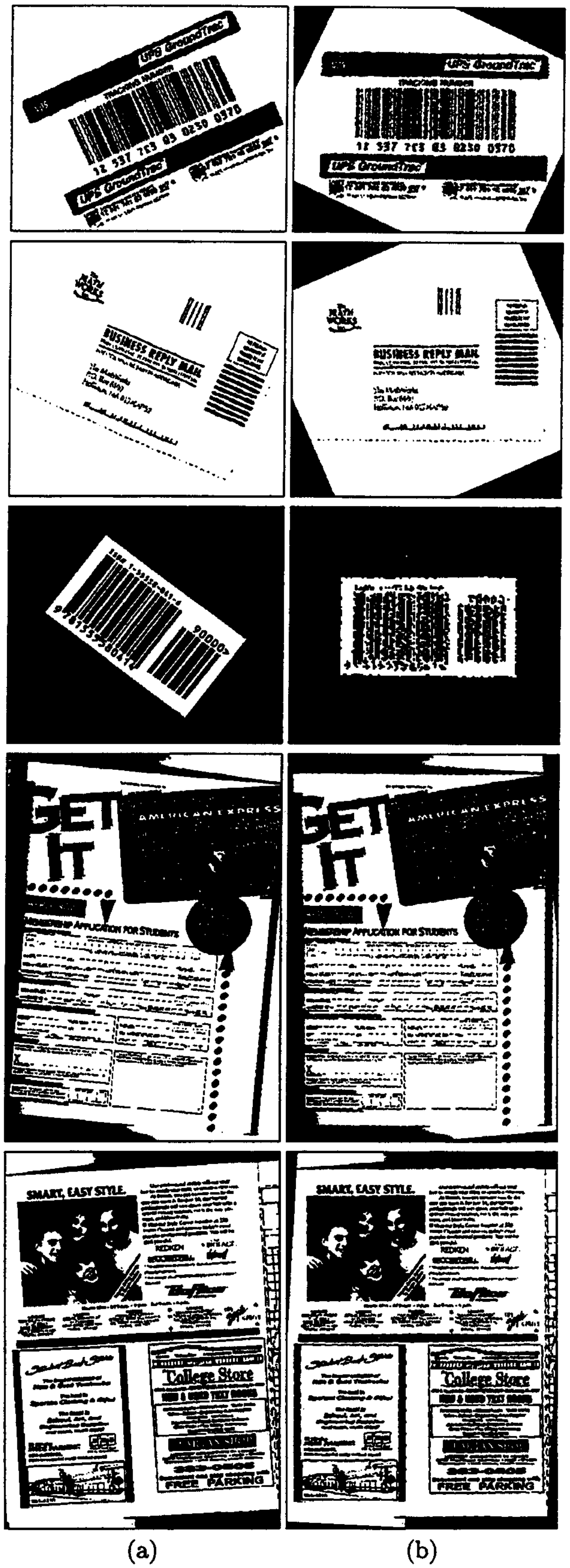


Figure 2: Skew detection experimental results. Input images in column (a) have been deskewed and are shown in column (b).

3 Language Separation

Language separation, where texts of different languages need to be discriminated and identified, is a problem of increasing interest in document image understanding. In this section we propose a texture-based segmentation method which is able to separate Chinese text from English text. This method, based on the observation that the texts of the two languages represent different textures, employs a neural network to train a set of "optimal" texture discrimination masks which minimizes the classification error for the given texture classes in the language separation problem. Texture features are obtained by convolving the trained masks with the input image. These features are then used in the language classification. A nice property of our algorithm is that both the feature extraction and classification stages have been combined into a single multilayer feedforward neural network to provide an efficient segmentation.

3.1 Learning Texture Discrimination Masks

An image region is textured if it contains some repeating gray level patterns [3]. Text regions are strongly textured because they typically follow a specific arrangement rule: each region consists of text lines of the same orientation with approximately the same spacings between them and each text line consists of characters of approximately the same size. We hypothesize that there exists some textural differences between texts of different languages because of the possible differences in the construction of fundamental elements (letters, characters, etc.) of languages or the placement rules of the elements. When the textural differences are large enough to be captured by texture analysis techniques, texts of different languages may be discriminated by an appropriate texture segmentation or classification method.

A widely used texture segmentation method is the multichannel Gabor filtering technique [4]. In this method, different Gabor filters are tuned to capture desired local spatial frequency and orientation characteristics of a textured region. Gabor filtering technique while popular in texture analysis because of its optimal localization in both the spatial and spatial frequency domains, does not guarantee that it is optimal for a given texture segmentation

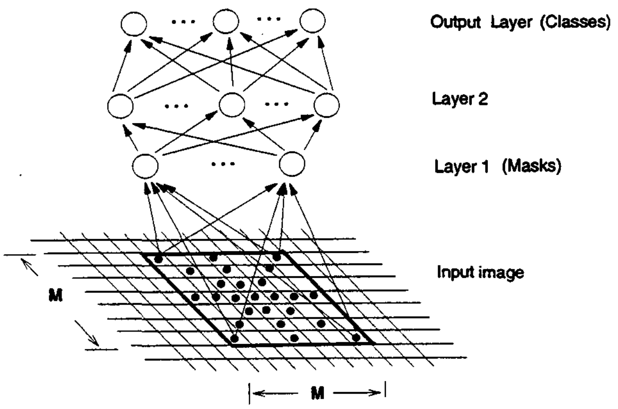


Figure 3: Three-layer neural network for texture classification

or classification task. Also, certain heuristics and domain-specific information have to be used to find a set of filters which gives good performance in a specific application. Jain and Karu [5]. have proposed a small number of simple "special purpose" filters, optimized for a given texture discrimination problem using a multilayer feedforward neural network. This set of filters is designed to minimize the classification error for a specific texture segmentation problem.

We have applied the above approach of learning texture discrimination masks to the language separation problem. A set of masks which best discriminates between the texts of different languages can be obtained by training a neural network on sample data from these classes. During the classification stage, each pixel in the input image, based on its neighborhood information, is classified by the network into one of the prespecified classes. A schematic diagram of the neural network classifier is shown in Figure 3. The input layer of the network is presented with pixels in a spatial configuration inside an $M \times M$ window of the input image as shown in Figure 3. It is assumed that the image texture is homogeneous in this window. Choosing this specific configuration instead of the entire $M \times M$ window reduces the number of connection weights (parameters) and results in an improved generalization performance and classification speed. Each node in the first hidden layer corresponds to a mask, where the weights of the links from the input nodes to the hidden layer nodes are the coefficients of the mask. Each node in the output layer corresponds to each of the output classes. Analogous to the Gabor filtering, the first layer of the network performs feature extraction, while the part of the network above the first hidden layer acts as a classifier.

The neural network operates as a standard multilayer perceptron and it is trained with the backpropagation algorithm. During the training session, each node in the hidden layers is also examined; nodes of low saliency are pruned. Details of the learning and node pruning algorithm are given in [6]. The node pruning process results in a compact and efficient network. It finds a smaller set of masks which has better or equivalent performance.

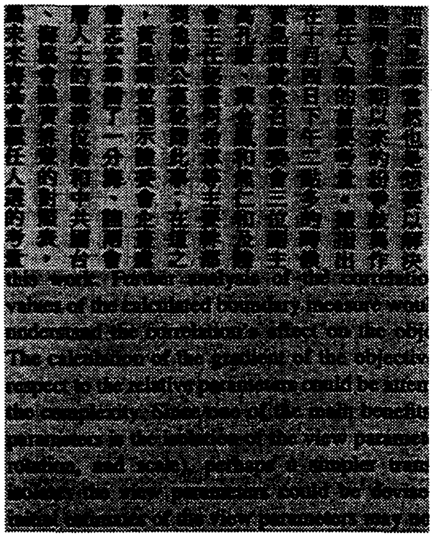


Figure 4: Training image of size 332 × 264 (100 *dpi*) for English/Chinese text segmentation.

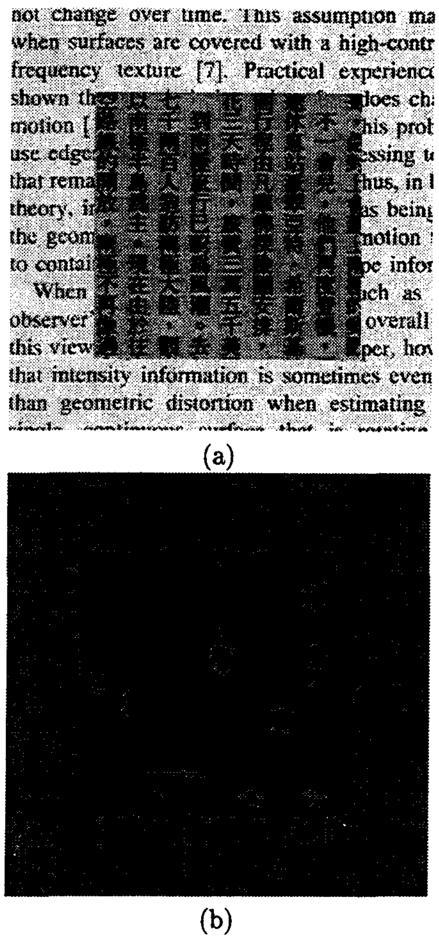
The training session of the neural network automatically tunes the weights of the masks so that the classification error is minimized. Domain specific information is incorporated in the network during the training. We have trained the set of masks for the two-class English/Chinese text separation. This set of masks is then convolved with the input image to give the feature vectors for text classification.

3.2 Experimental Results

We have applied the texture discrimination algorithm to perform the English/Chinese text separation. The test images were scanned at 100 dpi using a Sharp JX-300 flat-bed scanner. All the test images are grey scale byte images.

We used the neural network architecture in Fig. 3 to train a small number of masks which discriminate between the two languages. The training image is shown in Fig. 4. We used 1,000,000 patterns (with replacement) in the training session. After training and node pruning, fifteen 7 × 7 masks were retained according to the node saliency and the classification error.

This set of fifteen learned masks is then applied to segment English text from Chinese text. Fig. 5(a) is a test image which has a block of Chinese text nested inside the English text. The segmentation result is given in Fig. 5(b). Note that the output of the neural network classifier has been processed by a 5 × 5 median filter to smooth the 2-class segmen-



(a)

(b)

Figure 5: English/Chinese text segmentation. (a) A test image of size 253 × 253 (100 *dpi*). (b) segmentation result using fifteen learned masks.

tation result shown in Fig. 5(b). Small connected components have also been removed to enhance the segmentation results.

4 Page Segmentation

Page segmentation is a document processing technique which is used to automatically determine the format of a page. A scanned page image is first divided into blocks which are then classified as text, halftone, or line drawing. A correct output of a page segmentation system is necessary for many higher level analyses of the input document image including ICR, logical segmentation and recovery of the scanned documents.

We have used a hierarchical approach to classify text, line-drawing, halftone, and background regions. First, texture cues are used to discriminate between the three classes: (i) text and line-drawing, (ii) halftone, and (iii) background, using the same texture-based segmentation technique as described in section 3.1. The text and line-drawing regions are further discriminated based on connectivity analysis.

The proposed page segmentation method has the following desirable properties: (i) we are able to successfully segment the input image into four distinct types of regions (text, half-tone, graphics, and background), and (ii) the algorithm performs well at a relatively low scanning resolution of 100 *dpi*.

4.1 Page Segmentation Algorithm

Text regions possess a unique texture because they typically follow a specific arrangement rule: each region consists of text lines of the same orientation with approximately the same spacings between them and each text line consists of characters of approximately the same size. This specific texture property makes text regions distinguishable from the non-text regions. At the same time, the non-text regions, such as halftone and background, can be considered to represent different textures. The above observations suggest that the primary components of page layout – text, halftone, line drawings, and background – can be discriminated by an appropriate texture segmentation or classification method.

We have used a hierarchical approach to classify text, line-drawing, halftone, and background regions. In this approach, text and line-drawing regions are first put in the same category, because of their textural similarity in a 7 × 7 neighborhood. Using the same idea as in language separation, we used the neural network architecture described in Fig. 3 to train a set of masks which best discriminates between halftone, background, and text and line-drawing regions on sample data from these three classes. Text and line-drawings are further discriminated based on connectivity features. One reliable discriminator for text versus line-drawings is given by Wahl [7] who used the border-to-border distance within a connected component of the binarized document image. We have observed that line-drawing regions usually include irregular lines and curves which form connected components with larger sizes than those of individual characters. For each region classified as text or line-drawing, we threshold it to get a binary image and then perform a connected component analysis. If a region includes connected components whose sizes (either in the horizontal direction or in the vertical direction) are larger than a threshold, then that region is classified as a line-drawing region, otherwise a text region. The threshold size is empirically determined based on the size of the characters in the text. This simple heuristic (as well as all the processing done before) is invariant to rotation. Therefore, we can approximately locate areas of text, graphics, and halftone in the input image even if they are tilted or form irregular regions.

A postprocessing procedure is used to refine the segmentation results and calculate the bounding boxes for each identified regions. This process consists of two main steps. The first step removes small noisy elements and merges neighboring regions. The second step places bounding boxes around the labeled image regions. To speed up the processing without deteriorating the classification performance, we have applied all the postprocessing operations on a subsampled (50 *dpi*) segmented image.

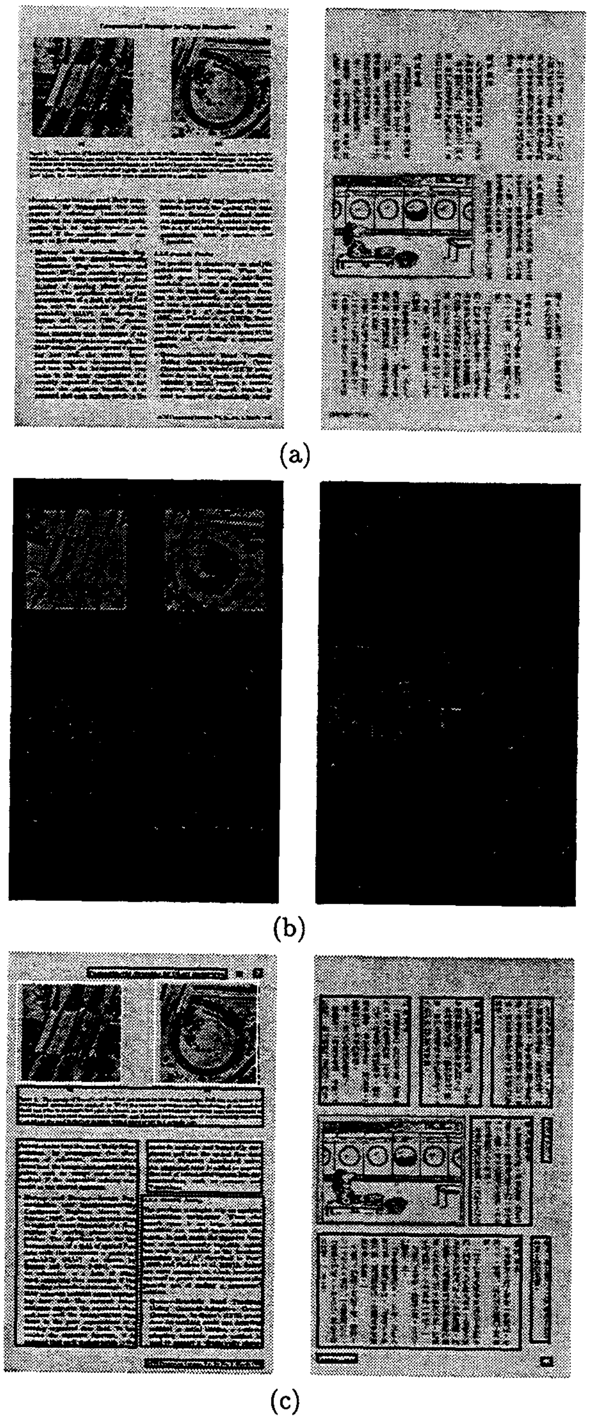
4.2 Experimental Results

We have applied our page segmentation algorithm on both English documents and Chinese documents. The document images used in the experiments were either scanned using a Sharp JX-300 flat-bed scanner or taken from the University of Washington (UW) English document image database at a resolution of 100 *dpi*. We have obtained consistently good segmentation results for a collection of document images from a variety of journals and magazines, including the *IEEE Trans. PAMI*, *American Scientist*, *Reviews of Geophysics*, *ACM Computing Surveys*, and *Reader's Digest* (Chinese).

A collage of portions of scanned pages (100 *dpi*) from the *IEEE Trans. PAMI* was used to train the texture discrimination masks in the neural network. Although the training image contains only English text, the test images consist of both the English and Chinese documents. These test images are also different from the training image. Fig. 6 presents the page segmentation results on two document images of different languages. The input images are shown in part (a) of the figure. The 3-class segmentation based on the texture discrimination masks is presented in part (b). Part (c) displays the resulting page layout with bounding boxes for each labeled region, where white, gray, and black bounding boxes are used to denote halftone, line-drawing, and text regions, respectively. These results show that the final page segmentations are correct and consistent for the test images considered here, even though the test images are different from the training image in scanning conditions, source journal, fonts, and language.

4.3 Summary

We have proposed a texture-based page segmentation method which is able to accommodate a variety of fonts, formats, and types of languages of document images from different journals. Although, we currently assume that the text skew is known a priori, it is not an inherent limitation of our approach since the segmentation is based on the texture information which can be considered as rotation invariant. The computational requirement of our algorithm is moderate. It takes approximately 70 seconds to segment an input image $(1,080 \times 780)$.



(a)

(b)

(c)

Figure 6: Page Segmentation results of an English (*ACM: Computing Surveys*, 939 × 606) and a Chinese (*Reader's Digest*, 760 × 496) document image (sixteen 7 × 7 masks). (a) input images (100 *dpi*); (b) three-class classification results; (c) final page layout with bounding boxes overlaid on the input images: black for text, grey for line-drawing, and white for halftone.

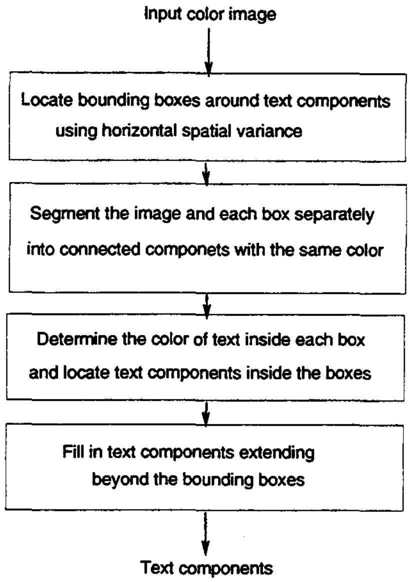

Input color image
Locate bounding boxes around text components using horizontal spatial variance
Segment the image and each box separately into connected componets with the same color
Determine the color of text inside each box and locate text components inside the boxes
Fill in text components extending beyond the bounding boxes
Text components

Figure 7: Block diagram of the algorithm for extracting text.

Note that the feed-forward neural network can either be implemented in hardware, or ported on commercially available accelerators to achieve "real-time" performance.

5 Locating Text in Images

In this section, we address the problem of automatically finding text in a complex color image. By a complex image we mean that the characters cannot be segmented by simple thresholding, and the color, size, font and orientation of the text are unknown. Applications of locating and recognizing text include address localization on envelopes, scanning and understanding a printed document, identifying products (such as books, CDs, canned food, etc.) by reading the text on their covers or labels. Another application area of a text understanding system is in image database retrieval, where the capabilities of usual textual database systems are extended to image databases [8].

We present a method for automatically locating text in complex color images. A high-level block-diagram of the algorithm for locating text is drawn in Figure 7. The first step of the algorithm finds the approximate locations of text using the large (horizontal) spatial variance of the image intensity on text lines. The output of the first step is a set of bounding boxes around predicted text lines. The second step performs color segmentation of the image. Individual characters are assumed to have uniform color, so that color segmentation followed by connected component analysis partitions the image into components, from which character components can be extracted using various heuristics. The proposed method has been used to locate text in compact disc (CD) and book cover images, as well as in the images of traffic scenes captured by a video camera.

5.1 Locating Text by Spatial Variance

The first step of the algorithm depicted in Figure 7 is to find approximate locations of text lines in a gray-scale image [9]. As a text line usually consists of aligned characters, if we compute the spatial variance along each horizontal line over the whole image, we see that regions with high variance correspond to text lines. We locate horizontal text lines in an image as regular rectangular regions with large horizontal spatial variance. We find significant horizontal edges and then pair the edges with opposite directions into lower and upper boundaries of a text line. From a pair of lines, it is a simple matter to construct the smallest rectangle which contains the two 'paired' lines as its upper and lower edges. All such rectangles form the output of this step of the algorithm – bounding boxes around candidate text regions.

5.2 Locating Text Components Using Color Segmentation

The second step of the algorithm segments an input image into components with uniform color. If we require that the color of an individual component be slowly varying, with sharp edges at its boundaries, then the basic algorithm for finding connected components can be applied. In that case, two neighboring pixels are in the same component if the difference between their colors is less than a threshold. This simple algorithm, however, does not work with most natural images where the characters have blurred boundaries, and are, therefore, connected with the background. A better method is to divide the image into components of constant color. The image segmentation can now be done by taking one color (a triple of the R-G-B values), and finding all the connected components with this color.

Due to the large number of possible colors, we first quantize the color space into a few prototype colors. The prototypes are found as local maxima in a smoothed color histogram of the input image, all other colors are then assigned to the nearest prototype. In most images used in our experiments, this results in about 5 – 500 peaks in the histogram, and the components that seem different to the human eye are quantized into different classes. Image segmentation can then be done with the basic connected component algorithm.

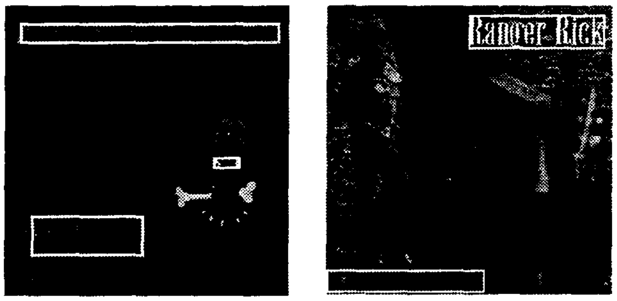


Figure 8: Locating text in CD cover images.

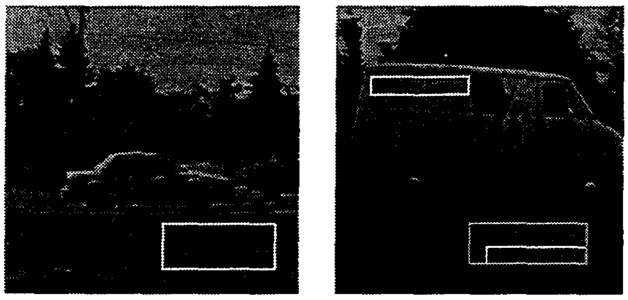


Figure 9: Locating text in video frames of traffic scenes.

5.3 Experimental Results

The proposed method was tested on CD cover images, book cover images and video frames. Image sizes varied from 256 × 256 for video frames to 700 × 500 for book covers. Figure 8 shows two input CD cover images together with the computed bounding boxes around text regions. It can be seen that the text which has a uniform color can be located accurately. A few spurious boxes produce some non-text components, which can be easily removed by an OCR system. Figure 9 shows images captured with a video camera. Again, bounding boxes are put around text located in these images. The algorithm for locating text is relatively fast. The spatial variance method, in addition to performing edge detection, requires no more than two scans over the image to compute variances, and the same number of scans to perform the connected component analysis of the edge pixels. Color segmentation requires one scan over the whole image to construct the color histogram, and two other scans to find connected components. For a typical 256 × 256 image, the entire processing takes approximately 10 seconds on a Sun Sparc station 20.

5.4 Summary

A method for extracting text in a color image is described. The approach is based on certain heuristics, and the algorithm cannot, therefore, be expected to find text in all possible situations. The most serious shortcomings of our algorithm are in failing to locate text which is not well separated from the background (this means that the color of the text and background is similar, or that the contrast of the text is low), and characters which are not aligned, or which have differently colored components. Nevertheless, the method gives good results on a variety of test images. The algorithm is relatively robust to variations in font, color, and size of the text.

6 Form Dropout

A typical form consists of the following three components: (i) form frames; (ii) preprinted data; and (iii) user filled-in data. The major task of form processing is to extract filled-in characters in data areas. Therefore, machine analysis of form documents should involve an intelligent form drop-out procedure which removes preprinted data with user filled-in data preserved. Since most character strokes are either attached to or written across the form frames, the problem of form drop-out involves two issues: (i) separation between characters and form frames and (ii) reconstruction of broken strokes introduced during separation. Because the pixels located in the areas where a character stroke crosses a form frame represent both a part of the stroke and a part of the frame, the stroke will be broken after removing the form frame. Therefore, we need to reconstruct the broken stroke. In this section we describe an algorithm that separates touching or overlapping characters from form frames. The basic idea is to identify the horizontal frames even when the form model is unknown, and in the presence of a modest amount of skew introduced during scanning.

6.1 Form Drop-out Algorithm

Figure 10 shows a block diagram of our algorithm. In this subsection, we will explain some key modules of the algorithm.

With the use of BAG [2] (Figs. 11(a) and (b)), we are able to find long horizontal frames by finding nodes whose length is longer than a threshold (Fig. 11(c)). The skew angles of the horizontal lines are statistically averaged. During the removal of horizontal form frames, some of the strokes of those touching or crossing characters will be broken. Therefore, we should perform character reconstruction to restore the broken strokes. By searching a neighboring area in BAG, we match the pair of broken strokes and interpolate the blocks between them (see Fig. 11(d)). Some of results of character reconstruction are shown in Fig. 12.

Vertical frames are processed as broken strokes. Then the long vertical frames are easily identified and removed. Some short vertical frames are difficult to distinguish from the long vertical strokes of characters such as numeral "1". We use two sources

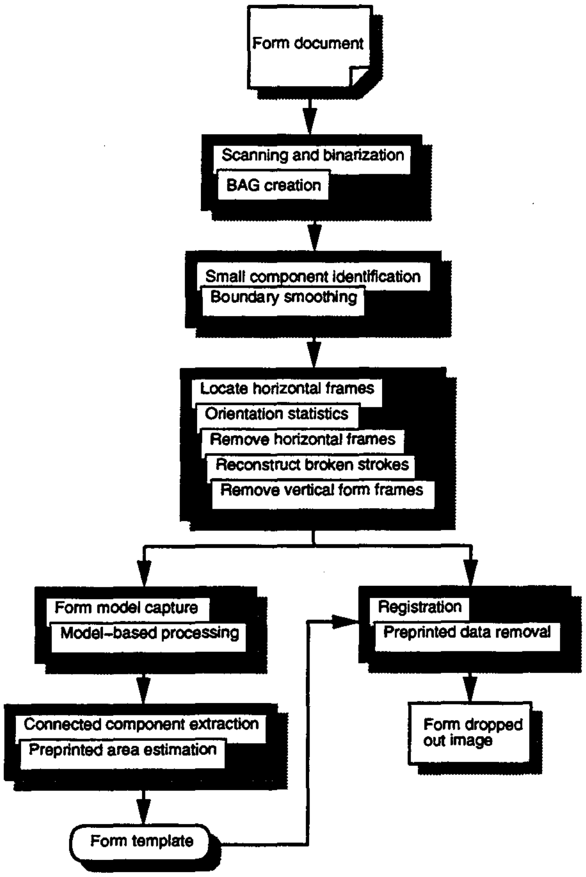


Figure 10: Block diagram of the form drop-out algorithm.

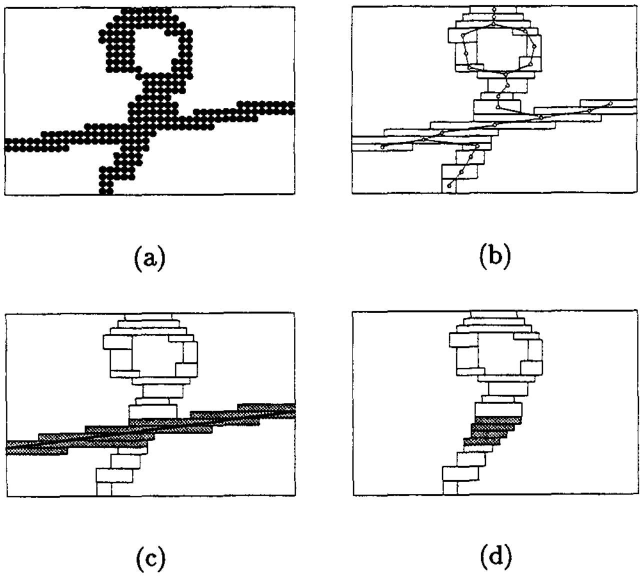


Figure 11: Form drop-out: (a) raster image; (b) BAG representation; (c) locating horizontal frame; (d) character reconstruction.

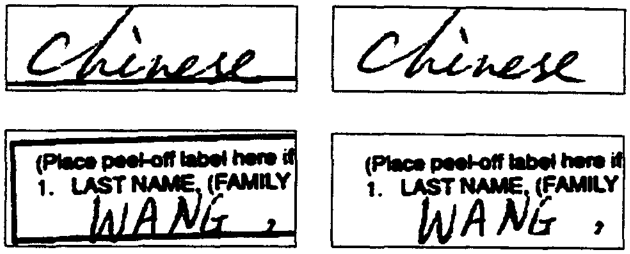


Figure 12: Some examples of character reconstruction.

of information in dealing with this problem. One is the linearity of the thin blocks representing a vertical segment. The other is the angle against a vertical reference line. The techniques mentioned above can be used to process a blank form image in order to automatically capture the form structure. After the removal of form frames, only the preprinted objects remain. We group them by clustering their bounding boxes according to their distance between one another. The skew angle and these preprinted data areas are automatically stored as a form template.

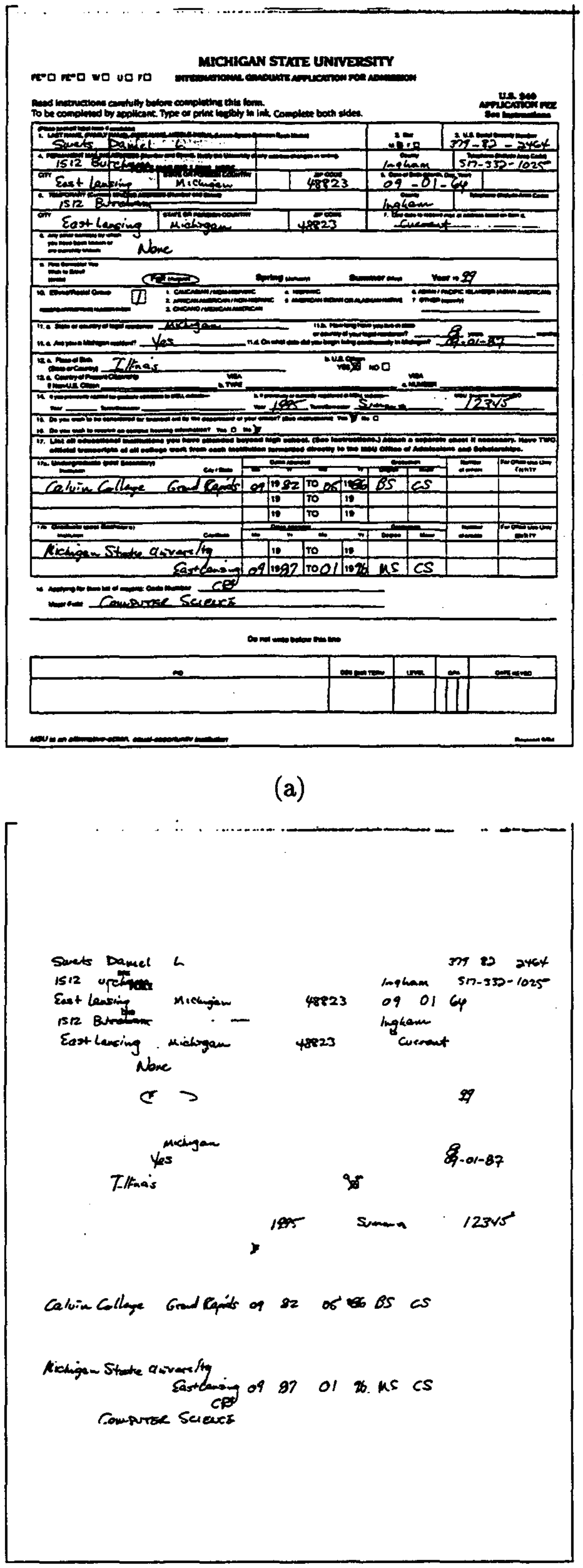
To process an input form, we first remove form frames and estimate the skew angle and then perform form registration with respect to the form template. Preprinted objects in the input form image whose bounding boxes have an overlap area with the preprinted data areas as defined by the form template higher than 50% are removed. We regard the remaining components as filled-in data which can be fed to an ICR module or data compression module.

6.2 Experimental Results

Ten graduate application forms for enrollment at the Michigan State University and eleven business reply mail (BRM) cards filled in by different people were scanned with a small amount of skew at a resolution of 300 dpi. The typical image sizes for an application form and a BRM card are 2550 × 3246 and 1609 × 1044, respectively. The average CPU time on a SPARC 20 is 11.36 seconds for an application form and 2.4 seconds for a BRM card. One of the application form images is shown in Fig. 13(a) and the form drop-out result for this image is shown in Fig. 13(b). One of the BRM card images is shown in Fig. 14(a) and the extracted filled-in data from this image is shown in Fig. 14(b).

6.3 Summary

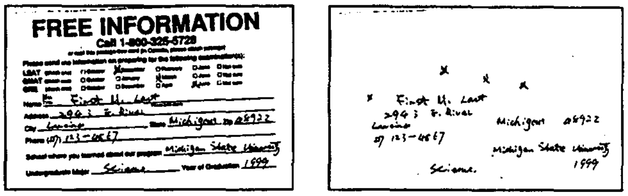
We have described a generic method for form drop-out when the filled-in handwritten characters or symbols are either touching or crossing the form frames and the form model is unknown. First, we propose a method to separate these characters from form frames. Then, we automatically capture the form structure from a blank form and utilize it to


MICHIGAN STATE UNIVERSITY
INTERNATIONAL GRADUATE APPLICATION FOR ADMISSION

(a)

(b)

Figure 13: Form dropout: (a) An input filled-in application form; (b) form drop-out result.



(a) (b)

Figure 14: Results on BRM card: (a) an input filled-in business reply mail card; (b) the extracted filled-in data.

extract the filled-in data. Since character strokes are either attached to or written so as to cross the form frames, the form drop-out problem of interest to us involves two issues: (i) separation between characters and form frames and (ii) reconstruction of broken strokes introduced during separation.

Our system has the following properties: (i) it can process a variety of forms and business reply mail cards; (ii) it is able to accommodate moderate skew in the input images and no special skew detection and deskew processes are needed; (iii) the system can drop-out both horizontal and vertical form frames without using any particular form knowledge even when there are characters touching and crossing the frame boundaries; (iv) it can correctly reconstruct most strokes that are broken by the removal of form frames; (v) by processing a blank form, the system can automatically pick out the form structure which is then stored as a form template; (vi) based on the captured form template, the system can extract filled-in data for a class of input form images.

7 Data Capture from Maps Based on Gray Scale Topographic Analysis

There is a large number of documents, including hand-printed maps, where useful information is lost if binarization is performed on the scanned image before higher-level processing. For such documents, methods which utilize the gray scale values must be used in order to extract as much of the available information as possible. Topographic analysis has been used in the literature to recognize characters directly in the gray scale images. We have extended the topographic analysis method so that characters and lines can be extracted from gray scale map images where methods using only the information in the binary image fail.

In topographic analysis, the gray scale image is treated as a topographic terrain surface. In our version of the method, each pixel is assigned one of twelve topographic labels: *peak, pit, ridge, ravine,*

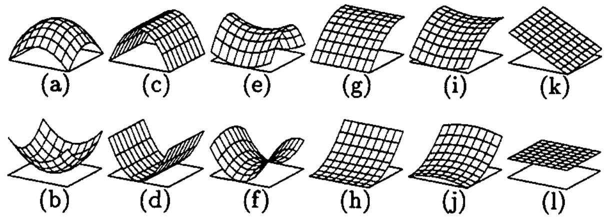


Figure 15: Topographic labels: **(a)** peak; **(b)** pit; **(c)** ridge; **(d)** ravine; **(e)** ridge saddle; **(f)** ravine saddle; **(g)** convex hill; **(h)** concave hill; **(i)** convex saddle hill; **(j)** concave saddle hill; **(k)** slope hill; and **(l)** flat.

ridge saddle, ravine saddle, convex hill, concave hill, convex saddle hill, concave saddle hill, slope hill, or *flat* (see Fig. 15). The labeling is used to segment the image or classify objects in the image.

7.1 Topographic Analysis for Maps

The proposed topographic analysis method for maps is an extension of Wang and Pavlidis' method [10]. It was designed to solve the specific recognition problems encountered when trying to extract map information automatically. The most prominent new component in our approach is the use of a binary image in the topographic analysis. The binary image, derived from the topographically labeled image, is used in combination with topographic labels, geometrical vectors, and directional information to extract the characters and lines in the input image.

Preliminary results showed that smoothing of the gray scale map image was necessary before applying topographic analysis. The implicit smoothing used in Wang and Pavlidis' method [10] was not adequate to suppress detection of spurious topographic features for map images. We experimented with various smoothing filter sizes to obtain a satisfactory noise suppression without loosing too much detail. The best result was obtained using a 17 × 17 Gaussian filter with a standard deviation $\sigma = 3.5$.

We obtained a binary image by dividing the set of labels into print objects and background. Pixels labeled *peak, ridge, ridge saddle, convex hill,* and *convex saddle hill* in the topographic label image all indicate negative dominant curvature, and were thus labeled *print* in the binarized image. Conversely, pixels labeled *pit, ravine, ravine hill, concave hill, concave saddle hill, slope hill,* and *flat* indicate positive or zero dominant curvature, and were labeled *background.*

In the topographic label image, many minor or almost flat hills and ridge lines were detected in background areas due to noise in the input image. These were seen as false print objects in the binary image, and were removed by the postprocessing step used in Yanowitz and Bruckstein's binarization method [11]. The print objects removed from the binary image were also removed from the topographic label image.

With the aid of the directional information available from the topographic analysis, the topographic labels were used to segment characters and lines as described below. Note that this method is rotation invariant.

Many characters in the binary image of a map touch or overlap each other. They need to be segmented into individual characters in order to be recognized.

A popular approach is to locate candidate split lines in the connected symbol, and for each split line, the two halves are classified using a statistical classifier. The split line producing the most probable pair of characters is taken as the correct split. These segmentation methods focus on separating characters appearing on the same text line. Here, we propose a new method which uses the topographic labels, the gradient, and the maximum curvature directions. This enables us to segment characters that are touching and belong to neighboring text lines. Also, this method is independent of the rotation of the characters.

We observed that both *ridge saddles* and *peaks* indicate candidate split points. Saddle points may indicate locations, where a character is about to be broken, or where two characters touch. Peaks may occur where two characters overlap, but also at stroke intersections within a character. Ridge saddles and peaks may also occur due to fluctuations of the ridge line height, so we developed rules to eliminate these, as described below.

To locate candidate split lines, the idea is to descend along the negative gradient direction from ridge saddles and peaks. For ridge saddles, the positive and negative direction of maximum curvature, $-\mathbf{e}_1$ and $\mathbf{e}_1$, were used as the starting directions. For peaks, the midline vectors between the lines meeting at the peak in the skeleton image were used as start directions. These directions were used as long as the descent was still inside the *ridge saddle* or *peak* labeled area. Once outside the ridge saddle or peak area, the negative gradient direction was used. The descent was required to be within convex saddle hill and convex hill pixels when descending from a peak, and convex saddle hill pixels only when descending from a ridge saddle. The descents stopped at the first background pixels on each side, and the closest contour pixels were used as the endpoints for the straight candidate split lines. Split lines which did not have a *ravine, ravine saddle,* or *pit* label at either end were discarded. For split lines having one end at an interior hole, two split lines ending in the same hole had to be used to split the connected symbol candidates. So, for each hole, all combinations of pairs of split lines were tested.

The skeleton image was traversed to identify intersection-free line segments that were too long to be part of a *pair* of touching characters. The long lines were removed from both the skeleton image and the binarized image. In the binarized image, we cut the connected component in such a way that characters touching or crossing the long lines had their shape preserved.

When the long lines had been removed from the binary image, the characters connected to these lines were segmented using the methods described above, provided no other lines were connected to the characters.

7.2 Experimental Results

Sections of a hydrographic map were used for testing. The map contained depth values, contour lines of constant depth, symbols, and grid lines (Fig. 16(a)). The map was scanned at 1,000 dpi with 256 gray scales with an Agfa Arcus scanner.

Following binarization and removal of noise using Yanowitz and Bruckstein's postprocessing step (PS) (Fig. 16(b)), the topographic binary image was compared to the binary image produced by applying Niblack's method (augmented with PS) to the unsmoothed gray scale image. Visually, Niblack's method gave slightly thinner print objects, and thus, slightly fewer pairs of touching digits (Fig. 16(d)). On the other hand, the new binarization method gave smoother object boundaries (Fig. 16(b)).

The segmentation steps were used to remove long lines and separate touching digits (Fig. 16(c)). Most digits touching other digits or connected to long lines were successfully segmented. However, short line segments still remained, and some were touching digits.

7.3 Summary

We have extracted unthinned binary images from the topographic analysis of gray-level map images. The topographic information was used to guide segmentation of characters and lines. The performance evaluation clearly indicated that the proposed method is superior to using the best available binarization method. However, it is unclear whether the binary image obtained before segmentation is any better than the results of using the faster Niblack's method with postprocessing. Wang and Pavlidis' method [12] produced a noisy topographic label image.

The current line extraction module was only capable of recognizing lines which were too long to be part of a pair of connected characters. It needs to be improved to recognize shorter line segments as well.

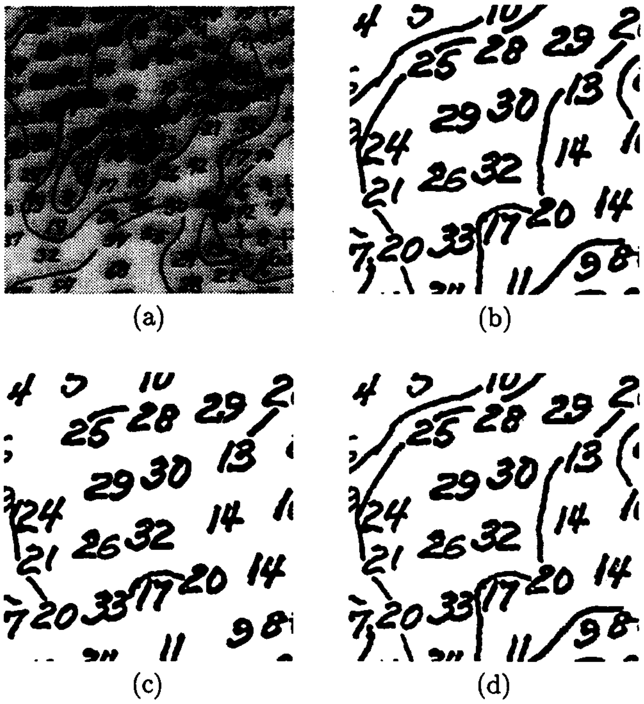

(a) (b) (c) (d)

Figure 16: Binarization and segmentation results on a 512 × 512 subimage: (a) A 1024 × 1024 portion of a gray scale image of a hydrographic map; (b) Binarized topographic labels after removal of false print objects; (c) Binary image after removal of long lines and segmentation of characters; (d) Binarized image using Niblack's method with PS.

The segmentation method for connected characters must be extended to handle self-touching characters. Self-touching characters have an interior hole that needs to be opened, and the topographic labels may provide good candidate split locations.

8 Hardware Implementation

The architecture of a general-purpose compute element is designed so that it provides a reasonable performance over a wide variety of applications. Many compute-intensive application areas, including image processing, face the challenge of optimizing the performance of algorithms on general-purpose architectures. Application Specific Integrated Circuits (ASIC) have been designed for situations where parallel processing techniques are not able to provide the desired performance cost-effectively. However, the ASIC-based approach suffers from the following limitations: (i) once an ASIC is fabricated, it is difficult to make changes in design; (ii) the design cycle is fairly long; (iii) ASIC solution is very costly if a large number of units is not produced to amortize the fixed cost of the masks needed for fabrication. These limitations of ASICs can be overcome by adopting the Field-Programmable Gate Array (FPGA)-based custom computing approach. Designers can implement their algorithms on FPGAs easily without using any special fabrication facility. The main architectural advantage of this approach is that the designer is not constrained by a general-purpose instruction set. The special instructions suitable for a given application can be implemented to enhance the overall performance of the system.

Many vision tasks have been implemented using FPGA-based custom computing. A number of these implementations have utilized Splash 2 – an FPGA-based processor attached to SUN SPARCstations. Splash 2 has been one of the successful FPGA-based custom computing machines. Many useful algorithms have been implemented using Splash 2 with impressive performance gains. Cox and Ekkeherd [13] have presented a FPGA-based implementation for neural networks. Since we have access to a Splash 2, we have chosen it as our platform for mapping the page segmentation algorithm.

8.1 Page Segmentation Algorithm

The page segmentation algorithm by Jain and Zhong [14] has three stages of computation, namely, (i) feature extraction, (ii) classification and (iii) post-processing. The feature extraction stage is based on a set of twenty 7 × 7 masks obtained by the learning paradigm proposed in [5]. The second stage is a multilayer feedforward neural network with 20 input nodes, 20 hidden nodes and three output nodes. The weights and other parameters of the neural network have been learnt for document images using the approach described in [5]. This page segmentation algorithm takes about 250 seconds of CPU time on a SPARCstation 20 for a 1,024 × 1,024 image. We now analyze the computational requirements of our page segmentation algorithm. There are twenty 7 × 7 feature extraction masks. Each feature vector (with 20 elements) needs 20 × 49 multiplications and 48 × 20 additions. Therefore, for an input image size of 1,024 × 1,024 pixels, the number of arithmetic operations needed in the filtering stage is of the order of 10 billion multiplications and 10 billion additions. The neural network classifier requires 400 integer multiplications in the first stage and 60 floating point multiplications in the second stage. The first stage also involves 20 × 19 adds and the second stage needs 3 × 19 additions. Thus, there are 460 million floating point multiplications in the classification stage. It is now evident that the computational requirements of our segmentation algorithm are enormous; a special processor is necessary to achieve a real-time performance.

8.2 Splash 2 Architecture

Splash 2 system consists of an array of Xilinx 4010 FPGAs. Figure 17(a) shows a system-level view of the Splash 2 architecture. The Splash 2 system is connected to the host (typically a Sun workstation) through an interface board that extends the address and data buses. The host can read/write to memories and memory-mapped control registers of Splash 2 via these buses. Each Splash 2 processing board has 16 Xilinx 4010s as PEs ($X_1 - X_{16}$) in addition to a seventeenth Xilinx 4010 (X_0) which controls the data flow into the processor board. Each PE has 512 KB of memory. There is a 36-bit linear data path (SIMD Bus) running through all the PEs. The PEs can also communicate through a crossbar. A broadcast path also exists by suitably programming X_0 and the crossbar. The processor organization for a PE is shown in Figure 17(b). The Splash 2 system supports several models of computation, including PEs executing the single instruction on multiple data (SIMD mode) and PEs executing multiple instructions on multiple data (MIMD mode). The most common mode of operation is systolic in which the SIMD bus is used for data transfer. Individual memory available with each PE makes it convenient to store temporary results and tables. The steps in programming a Splash 2 are (i) simulation; (ii) synthesis; and (iii) host interface development. In simulation stage, the logic designed using VHDL is verified.

The I/O functions are primarily carried out in the host other than initializing Splash 2 and loading the control bit-stream data that sets up the FPGAs for the specified instructions. To change the instructions on the FPGAs, a new bit-stream file needs to be loaded by the host. The results at the end of the operations are read by the host. These functions are provided through a set of C-callable routines for host-interface program development.

8.3 Mapping Page Algorithm on Splash 2

There are two main phases in the segmentation algorithm not considering the post-processing stage. The filtering stage is carried out first. The Splash 2 system is then reconfigured to implement the neural network classifier.

The mask values in our application have been derived using the learning paradigm described in [5]. The twenty 7 × 7 feature extraction masks are real-valued. Our filtering algorithm is implemented as a 2-D convolution. In our earlier work [15], masks with integer values only were considered. We have now extended the 2-D convolution to real-valued masks. One of the limitations with the current FPGAs is the minimal support for floating point operations.

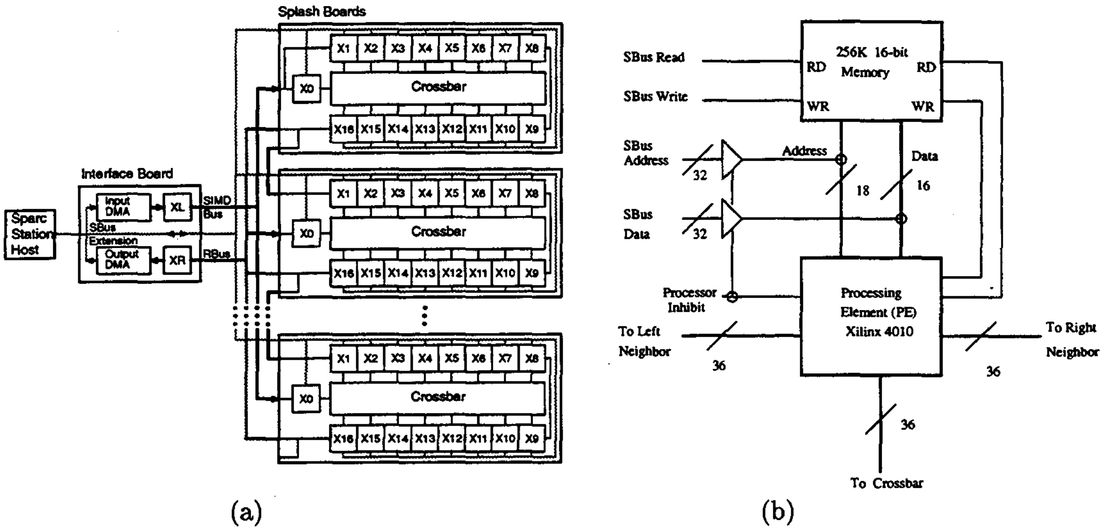


Figure 17: Splash 2 attached processor: (a) Architecture; (b) One processing element.

Our floating point multiplier involves following three main stages: (i) normalization: the mask coefficients are normalized in the range [-1, 1] by dividing the mask values by a constant chosen to be a power of two and greater than or equal to the largest absolute mask value. (ii) floating point multiplication: multiplication of an integer by a fraction is considered as successive sums of partial results. (iii) renormalization: the result is scaled back by the normalization constant used. With this approach, a general-purpose floating point multiplier is replaced by a series of adders by taking advantage of the input range being [0, 255]. The 2-D convolution algorithm with integer masks is described in detail in [15]. We provide only a brief summary here. A systolic algorithm has been selected because of its architectural suitability on Splash 2. The algorithm uses a linear array of processing elements. Thus, every PE has a left neighbor and a right neighbor. The first PE receives input from the host and the last PE outputs the result to the host. In order not to confuse with the PEs of Splash 2 (physical PEs), let us call these as virtual PEs. Each virtual PE receives a partial sum and the pixel value from its left neighbor and multiplies the pixel and mask value assigned to it. After adding the partial sum (received from the left neighbor), the result is sent to the right neighbor. After the latency involved in the linear array of PEs, we get an output result at every clock cycle. The number of virtual PEs per physical PE is decided based on the mask values.

The network has twenty input nodes, each node corresponding to one of the texture filter outputs (real-valued). A MLP consists of several perceptrons interconnected in a feedforward manner. In implementing a neural network classifier on Splash 2, a perceptron implementation has been used as a building block. Hence, we focus on the design of a perceptron. In our case, the perceptron is assumed to have 20 inputs which uses a non-linear function (typically a sigmoid function) to produce a real-valued output. A perceptron consists of two stages namely, (i) an inner product computation, and (ii) a non-linear function applied to the output of the previous stage. The inner product of the feature vector with the weight vector can be easily computed using a 1-D convolution algorithm. In order to handle the floating point multiplication, we use a lookup table. The input to the non-linear function is the 1-D convolution of the feature vector with the weight vector. In order to simplify the implementation of the non-linear function, we have used a lookup table stored in the PE memory available on Splash 2. The output layer in the network consists of three perceptrons. This stage is different from the earlier stage in the sense that the input data is available for all the neurons at the same clock cycle. We use a set of multipliers and tree adders to implement the inner product for this stage. However, the non-linearity implementation is still based on lookup tables.

8.4 Experimental Results

The segmentation algorithm has been mapped onto a Splash 2 with 2 processor boards (i.e., at most 32 PEs are available for mapping). The functions of the PEs are modeled using VHDL. The result of simulation is verified using the timing diagrams obtained from the simulator. Using the synthesis phase, the control bit stream for the FPGAs has been generated. Based on the delay model for Xilinx 4010, the maximum speed for the logic accommodated on the FPGA chips for the filtering stage is estimated to be 10.8 MHz.

We need 20 filters and we have 2 processor boards.

Each filter occupies one Splash processor board. Hence, we have to make a total of 10 passes over the input image. For this purpose, we use the reconfigurability of the FPGAs to change the instructions dynamically. Note that for each filter, the coefficients change. Hence, we need a different set of adders which changes the instruction for the PEs. In terms of the number of operations per second, the clock speed reflects the rate at which the input pixels will be handled, i.e., 10.8 million pixels/second. Hence, a 1,024 × 1,024 image can be processed in approximately 0.1 seconds. Thus, 10 passes through an image would take approximately one second. This computation time can be compared with the computation time of 250 seconds needed on a SPARC-station 20 (33 MFlops). In other words, 20 billion operations are carried out in one second using the Splash 2 system.

9 Summary

Document image processing is a challenging area of research. We have addressed a number of significant and difficult research issues and problems. Our future work will concentrate on designing complete systems for form processing, page analysis and content-based retrieval.

References

[1] B. Yu and A. K. Jain, "A robust and fast skew detection algorithm for generic documents", submitted to *Pattern Recognition.*

[2] B. Yu, X. Lin and Y. Wu and B. Yuan, "Isothetic polygon representation for contours", *CVGIP: Image Understanding*, vol. 56, pp. 264–268, 1992.

[3] M. Tuceryan and A. K. Jain, "Texture Analysis", *Handbook of Pattern Recognition and Computer Vision*, C. H. Chen, L. F. Pau and P. S. P. Wang, World Scientific Publishing, pp. 235–276, 1994.

[4] A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using Gabor filter", *Pattern Recognition*, vol. 24, pp. 1167–1186, 1991.

[5] A. K. Jain and K. Karu, "Learning texture discrimination masks", to appear in *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1995.

[6] J. Mao, K. Mohiuddin and A. K. Jain, "Minimal network design and feature selection through node pruning", in *Proc. 12th Int'l Conf. on Pattern Recognition*, Jerusalem, pp. 622–624, October 1994.

[7] F. M. Wahl, "A new distance mapping and its use for shape measurement on binary image data", IBM Research Report, San Jose, CA, vol. RJ3438, 1982.

[8] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic and W. Equitz, "Efficient and effective querying by image content", *Journal of Intelligent Information Systems*, vol. 3, pp. 231–262, 1994.

[9] Y. Zhong, K. Karu and A. K. Jain, "Locating text in complex color images", in *Proc. 3rd Int'l Conf. on Document Analysis and Recognition*, Montreal, pp. 146–149, August 1995.

[10] Peter Meer and Isaac Weiss, "Smoothed differentiation filters for images", *Journal of Visual Communication and Image Representation*, vol. 3, pp. 58–72, 1992.

[11] S. D. Yanowitz and A. M. Bruckstein, "A new method for image segmentation", *Computer Vision, Graphics and Image Processing*, vol. 46, pp. 82–95, 1989.

[12] L. Wang and T. Pavlidis, "Direct gray-scale extraction of features for character recognition", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, pp. 1053–1067, 1993.

[13] C. E. Cox and E. Blanz, "GANGLION–A fast field-programmable gate array implementation of a connectionist classifier", *IEEE Journal of Solid-State Circuits*, vol. 27, pp. 288–299, 1992.

[14] A. K. Jain and Y. Zhong, "Page layout segmentation based on texture analysis", to appear in *Proc. 2nd International Conf. on Image Processing*, Washington D. C., 1995.

[15] N. K. Ratha, A. K. Jain and D. T. Rover, "Convolution on Splash 2", to appear in *Proc. of FPGAs for Custom Computing Machines*, 1995.

Forms Decomposition/Cursive Word Recognition

Tony Cristofano
Technology Solutions Inc.
1 Colleen Road
Fredricksburg, VA 22406

Robust Handwritten Signature Recognition

Kevin Matsumura **James Peterson**
TRW Avionics and Surveillance Group
Sunnyvale, CA

Abstract

The automatic recognition of handwritten signatures within a document image can be a powerful tool for characterizing the contents of a document. Under the Inscribe contract, TRW/ASG has developed a robust signature recognition technique for low-resolution, bit-mapped document images (100 x 100 dpi), which is largely unaffected by overlapping text or horizontal lines, differences in pen thickness, or signature variability. The Inscribe signature recognition technique employs a feature-space decision algorithm to compare unknown signatures to a database of known signatures of interest. The signature feature set consists of 41 separate characteristics, including moment measures, closed loop statistics and signature linearity characteristics, which are tolerant of the "noise sources" mentioned above.

1 Problem Description

The ability to recognize handwritten signatures can provide a powerful means of automatically identifying important document images. Signature recognition may be used alone, as a signature verification tool, or may be incorporated into a comprehensive document image processing system, as shown in Figure 1. As this figure illustrates, a document image processing system can perform image enhancement and segmentation prior to signature recognition. Image enhancement attempts to correct for skew or speckle-noise errors, providing a better quality document for content analysis. Segmentation divides the document image into a number of distinct regions containing one of the following types of information: ideographic, machine-generated text, handwriting, or other, and passes each image region to the appropriate follow-on content recognition tool. Segmented regions containing potential signatures (typically, handwritten cursive text of limited size) are sent to the signature recognition module, which identifies known signatures of interest. This information can then be passed to a document understanding module which may characterize and/or prioritize each document based on the results of the individual content analysis tools.

A significant challenge in automatic signature recognition is signature variability [1,2]. Multiple samples of an individual's signature are never completely identical. Signature samples may deviate in a variety of ways, including signature size, pen thickness, embellishments, and the presence or absence of a middle initial. Additional variables can be introduced, if the document is photocopied or electronically transmitted, due to variations in the scanning angle and transmission errors.

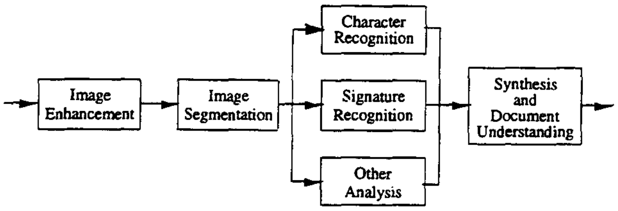


Figure 1: Conceptual System Diagram Containing a Signature Recognition Module.

A second major challenge is that signatures in documents often overlap horizontal lines or machine-generated text. Signatures present on forms and checks almost invariably extend across horizontal lines, and signatures often overrun machine-generated text at the conclusion of letters, as illustrated in Figure 2. Classical image recognition techniques usually fail when applied to these occluded cases.

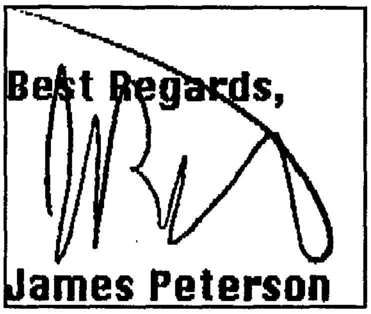


Figure 2: Typical Instance of a Signature Overlapping Text.

The goal of the initial Inscribe contract effort was to develop an automatic signature recognition module which takes a segmented signature image as input and determines whether it matches any signatures in a database of known signatures of interest. The Inscribe approach utilizes a variety of features calculated from the input signature to identify the best match(es) from a known signature set and to quantify the relative closeness of each match.

2 Approach

The initial signature recognition algorithm development effort consisted of three main phases: problem analysis, candidate feature set development and evaluation, and final system design and performance evaluation. As a result of the initial technical assessment, a feature-space decision algorithm [3,4,5,6,7] approach was selected for the recognition system. Using this method, a number of image characteristics, or features, are extracted from the input signature and stored in a feature vector, which is compared against a database of known feature vectors to determine whether the signature is of interest. The image features used are carefully selected to be tolerant of signature variability. Feature space determination offers a flexible solution to this recognition problem because the feature vector can easily be refined, as better features are developed, to further improve the system performance.

A high level diagram of the Inscribe signature recognition process is shown in Figure 3. The inputs to the Inscribe system consist of the segmented signature image and the pre-defined feature database containing the signatures of interest. The system outputs consist of the names of those signatures in the database that best match the input signature, and the corresponding confidence factor associated with each match. There are three main steps in the recognition process: pre-processing, feature extraction, and signature matching. The following sections describe each of the processing steps in more detail.

Note that Figure 3 indicates that feature extraction is performed on three separate images. Initial tests of the feature extraction module showed that variations due to pen thickness and overlapping lines or text could cause significant deviations in the feature calculations for different samples of an individual's signature. Therefore, it was necessary to implement a method of extracting invariant characteristics from the input signature prior to feature extraction. Invariant characteristics are considered to be qualities that remain approximately constant across multiple samples of the same individual's signature. The pre-processing stage of the Inscribe algorithm generates three intermediate images, each of which contains specific invariant image characteristics from the input signature. The three generated images are a line segment representation, a loop extracted representation, and a text/horizontal line-removed representation. These three representations are described in more detail in Section 2.1.2.

2.1 Step 1: Pre-Processing

As previously stated, the pre-processing stage addresses the primary difficulty involved with signature recognition: signature variability. Pre-processing is sub-divided into general and feature-based pre-processing.

2.1.1 General Pre-Processing

The general pre-processing stage filters and manipulates the input signature image into a more standardized form for the feature extraction algorithms by performing speckle noise removal, cropping, and line thinning. Within a document image understanding system such as that pictured in Figure 1, the speckle noise removal and cropping functions will typically be performed by the image enhancement and segmentation modules, respectively. However, since the Inscribe signature recognition process was initially designed to operate as a stand-alone module (for ease of evaluation), these functions were included in the Inscribe processing chain.

Removal of speckle noise (small, isolated groups of black pixels) is important because these pixels may adversely affect the recognition process. Speckle noise is removed using a modified median filter which targets isolated groups of black pixels. Pixels are considered to be in the same group if they are connected to one another in any of the 8 nearest neighbor spots [8].

Cropping eliminates the excess whitespace bordering the image, generating a more standardized input to the

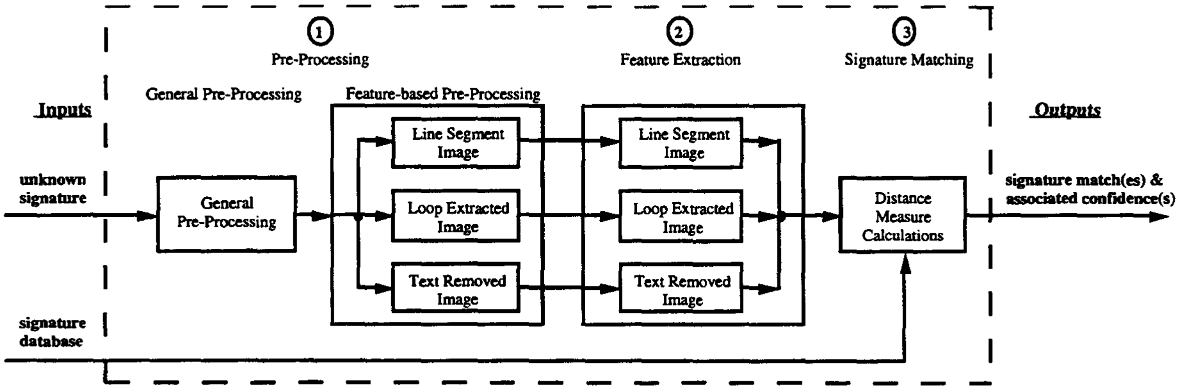


Figure 3: Signature Recognition Algorithm Process Flow.

feature extraction functions. The cropping function is performed by scanning the image inward from each segment border to establish the unique rectangle defined by the outermost black pixels in the segment.

Thinning eliminates the effects of varying pen thicknesses on the signature image. A morphological thinning technique [3,8,9] reduces all pen lines to a one-pixel thickness without losing connectivity. Figure 4 shows the results of the thinning operation.

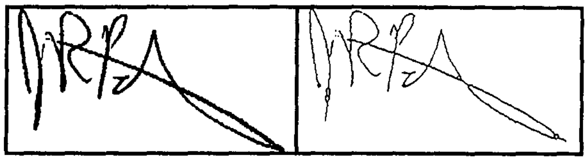


Figure 4: Input Signature (left) and Morphologically Thinned Output (right).

2.1.2 Feature-Based Pre-Processing

The second stage of pre-processing derives three intermediate images from the input signature image: a line segment image, a loop extracted image, and a text/horizontal line-removed image. Each of these pre-processed signature images retains different invariant characteristics of the input signature while canceling the effect of occluding text or lines. It is then valid to apply classical and custom techniques to these intermediate images in order to generate the feature set.

The first of the three intermediate images, the line segment signature image, only retains information from line segments in the input signature which are longer than the maximum expected text size. These line segments are identified by a set of line templates. Line segments that are predominantly horizontal are excluded from the line segment signature image, as they may represent an overlapping line that is not part of the signature. Figure 5 depicts the line segment representation.

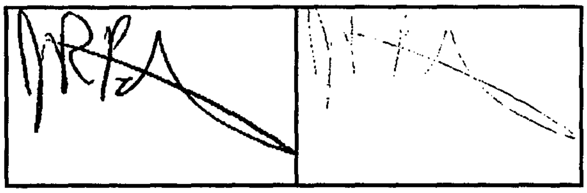


Figure 5: Input Signature Image (left) and Line Segment Intermediate Image (right).

The second intermediate image retains only closed (and nearly closed) loops [9,10] from the input signature which are larger than a specified size. A morphological closing operation [8] is performed on the signature image to close any potential loops such as the one shown in Figure 6.

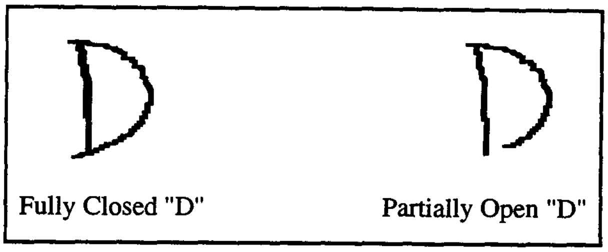


Figure 6: Example of Signature Letter Loop Gaps.

Once the morphological closing operation is complete, all enclosed areas of whitespace are identified using connectivity analysis [10]. The output image representation consists of only the large loops in the input image. Figure 7 shows a loop extracted representation.

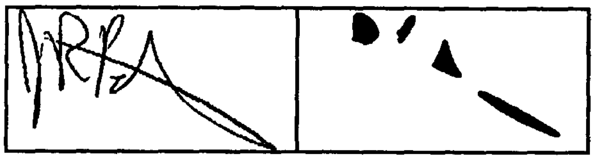


Figure 7: Input Signature Image (left) and Loop Extracted Intermediate Image (right).

The third intermediate signature image is a text/horizontal line-removed representation. This image product is generated by identifying and erasing text and horizontal lines in the input signature image. The horizontal line removal algorithm analyzes the composition of the individual rows of pixels in the signature image to decide when to erase a line. The text removal algorithm considers the expected letter mass, the horizontal distribution of "suspect" letters, and their relative positions in the image as factors in erasing text. Figure 8 depicts the results of the horizontal line removal algorithm and Figure 9 displays the results of the text removal algorithm.

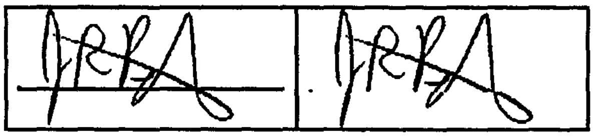


Figure 8: Signature Image Before and After Horizontal Line Removal.

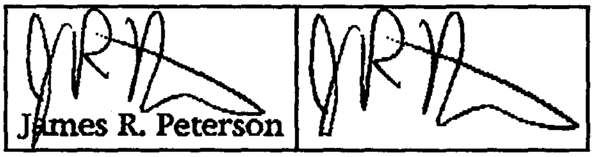


Figure 9: Input Signature Image and Text Removed Intermediate Image.

2.2 Step 2: Signature Extraction

The feature extraction process involves extracting invariant image characteristics from the three intermediate signature images generated by the pre-processing module. In the second phase of the Inscribe development effort, 180 image features were implemented for possible use in the signature recognition algorithm. The performance of each of these candidate features was evaluated on a test set of signatures using statistical calculations to determine the strongest features for inclusion in the final system design. The composition of the test signature set is described in Section 2.2.1. The method used to obtain a final feature set for signature recognition is outlined in Section 2.2.2.

2.2.1 Feature Test Signature Set

The test signature set consisted of 210 signatures--ten signatures each from 21 participants. The set was designed to include clean samples of an individual's signature as well as samples which exhibit the possible variables that a signature recognition system is likely to encounter. The composition of the final signature set is shown in Table 1. Each signature sample was scanned and converted into a bit-mapped image format.

Table 1: Test Signature Set Composition.

Signature Set	% of Signature Set	Characteristics
I	29	**Well behaved:** Signatures written by fine or medium point ball-point or felt-tip pens.
II	21	**Moderately difficult:** Signatures with overlapping horizontal lines, or written by heavy felt-tip pens.
III	50	**Heavily corrupted:** Signatures with overlapping machine-generated text.

2.2.2 Feature Evaluation

The objective in developing a feature-based recognition algorithm was to obtain a set of features which reliably identify a unique signature. The selection of a final feature set was a two-stage process. In the first stage, the original 180 recognition features were evaluated using a "feature quality measure" to determine which features were most useful in distinguishing one signature from another. This "feature quality measure" was computed for each candidate feature, by taking the ratio of the average feature variance between the different signature classes divided by the average variance within each signature class. Features with a ratio of 1.5 or less exhibited poor class separation characteristics and were discarded.

The second stage in obtaining a final feature set was to identify any highly correlated features within the remaining feature set. Since two or more features which are highly correlated contribute essentially the same information, only one of the features needs be calculated in the classification process. The final feature set was obtained by calculating a covariance matrix and reducing groups of highly correlated features to a single feature by combining or eliminating the redundant measurements.

The final signature feature set consists of classical image processing calculations (such as moment measures and aspect ratios) along with customized calculations specific to the invariant representations (including loop area, loop dimensions, and dominant line slopes). Of the 180 signature features that were initially evaluated, 41 separate feature measures were retained in the final signature feature vector. Fifteen of these features are derived from the linear segment signature image, fourteen from the loop extracted image, and eleven from the text/horizontal line-removed image. In terms of classification performance, the strongest features are evenly divided between loop-based and line-segment based features. Third order central moments [3,11] of the loop- and line-segment signature products hold particularly significant signature information. Features which measure line segment slope were also found to be distinctive for different individuals -- especially for separating the signatures of left- and right-handed individuals.

2.3 Step 3: Signature Matching

The final step in the signature recognition process is match determination [6]. This stage identifies which signatures in the known signature database best match the input signature, and reports a confidence level for each match.

As previously stated, the inputs to the signature recognition module are an unknown signature image and a database of known signatures to which the unknown signature is compared. The signature database contains signatures of interest along with their associated feature vectors. A signature naming convention enables the system to report the name of the individual to whom the matching signature belongs and permits the operator to enter multiple copies of an individual's signature into the database. When multiple copies of an individual's signature exist in the database, a single composite signature feature vector is generated based on each of the signature entries.

Once the input signature feature vector is obtained, it is compared against the feature vectors of each signature group in the database to determine if there is a likely match. The match decision is based on a modified minimum-distance classification [3,5,6,7].

The elements of the input and database feature vectors are weighted so the strongest performing features have the most leverage in influencing the final signature match determination. The database signature feature vector with the minimum distance to the input feature vector is chosen as the most likely signature match and the feature distance is used to compute a confidence rating for the signature classification result.

3 Results

The initial performance of the signature recognition module was evaluated using round-robin approach on the test signature set described in Section 2.2.1. In this approach, a known signature database was created from nine of the ten signature samples available for each individual, and the tenth sample of each individual was input as an unknown signature. Overall performance of the classification system was thus accomplished by rotating through all 210 signatures in the test signature set in this manner. This approach did not optimistically bias the module's performance by training on the actual signature, and was the best option when considering the limited data available.

3.1 Recognition Accuracy

The results obtained in the initial performance evaluation of the Inscribe signature recognition algorithm are presented for each of the three test signature sample set categories discussed in Table 1 of Section 2.2.1.

When the test signatures were limited to the "well behaved" cases in Set I, (i.e., no text or horizontal lines overlapping the signatures and no overly thick pens used) the recognition algorithm was 100 percent successful at identifying the correct signature. The well behaved signature population consisted of 60 signatures representing all of the 21 signature classes.

When the test signature population was expanded to include the 45 "moderately difficult" signatures in Set II (i.e., signatures written with thick-point felt tip pens or signatures with overlapping horizontal lines), the recognition success rate slipped to 93.3% (98 out of 105).

When the signature set used for evaluation was further expanded to include the 105 signatures in Set III, signatures with overlapping machine-generated text, the recognition success rate measured 87.5 percent (184 out of 210). Table 2 shows the cross-tabulated classification confusion matrix associated with the complete signature set of 210 signatures. The values along the diagonal represent the number of signatures (out of 10) that were correctly classified for each of the 21 individual volunteers. Values along a given row indicate how mis-classified signatures were distributed among the other signature classes.

In assessing the recognition results for specific input signatures, it was found that the highest error rates were obtained in cases of individuals who varied the way they wrote the first letter of their signature. For example, in the signature samples of Individual B, the first letter in her signature is sometimes written in cursive and other times printed. Similarly, Individual M tends to alternatively sign the "D" at the start of his signature as either a closed loop or a widely open non loop. Other factors which contributed to recognition errors include:

- Excessive corruption of the signature due to heavy overlap of machine-generated text
- Small, tightly-written signatures, portions of which (in a few instances) were confused by the text-removal algorithm with machine-generated text
- Artifacts from thick pen lines which occasionally distorted the thinned signature

Based on the results obtained, the Inscribe system performed admirably on the initial test signature set, especially when considering the large percentage of text-overlapped and line-overlapped signatures. As would be expected, the accuracy is highest when multiple clean copies of an individual's signature are included in the database training set. The multiple samples help to ensure that the reference feature values in the database are fully representative of that individual.

3.2 Processing Time

The initial implementation of the Inscribe signature recognition module was not specifically optimized for speed performance. In its current form, the software requires an average of 3.7 seconds on a Sun Microsystems SPARCstation 10 to process and analyze an input signature. Most of the processing time is expended in the feature-based pre-processing portion of the algorithm. In particular, about half of the total processing time per input signature (1.75 seconds per signature) is spent in the morphological thinning process. It should be noted that this average time per signature may represent an overestimate due to the makeup of the test signature set. More than twenty percent of the signatures in the test set had been written with a very thick felt tip pen in order to test the algorithm's performance with thick pen lines. The line thinning algorithm is an iterative algorithm and takes significantly longer to process these thick signatures. The next most time-intensive process is the line template matching (0.75 seconds per signature) which is performed during linear segment extraction.

Future Goals

Although the Inscribe signature recognition algorithm has been shown to perform well in initial tests, additional testing is required to more fully characterize its overall performance. One area that needs to be

Table 2: Classification Confusion Matrix for Initial Signature Test Set (210 Signatures Total).

		Output Signature																				
		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
Input Signature	A	10																				
	B		4	1							1									3	1	
	C			9		1																
	D				9													1				
	E			1		9																
	F						8		1										1			
	G							10														
	H								10													
	I									9											1	
	J										10											
	K											10										
	L					1	1						6	1					1			
	M								2					6					1	1		
	N										1	1			8							
	O															10						
	P																9				1	
	Q		1															9				
	R																		10			
	S							1												9		
	T																1				9	
	U																					10

investigated is the algorithm performance on an expanded signature set which includes signatures that are not part of the known signature database in addition to signatures that are. The following recommendations for further development are based on the results of the initial evaluation.

4.1 Improved Recognition Accuracy

Based on the results of our initial tests, further improvements can be made in how the signature recognition module distinguishes handwritten signatures from machine-generated text, removes overlapping text, and thins thick pen lines. Incorporation of several new types of feature measures may greatly enhance the recognition performance on such signatures. In particular, modified Hough transforms [12], have been shown to have value for quantizing linear features such as handwritten signatures. Standard Hough transforms [9] also have significant potential for improving the ability of the linear segment extraction process to distinguish handwritten lines from machine-generated text. Additional improvements in algorithm performance may be obtained by refining the signature feature matching algorithm to incorporate information from the feature covariance matrix or to use a neural-net-based recognition technique [1] instead of the current minimum distance measure.

4.2 Processing Speed

Although speed was not a consideration in the development of the initial signature recognition prototype, it is desirable for the final version of the module to execute as fast as possible. Since the most significant portion of the processing time is currently spent in the morphological thinning and line segment template matching algorithms, optimizing these two functions will dramatically improve the speed of the signature recognition module.

References

[1] Schwartz, Edward L., *The Design of a Parallel Processor System and Application to a Neural Network Based Character Recognition Machine,* Master's Thesis, University of California, Davis, 1990.

[2] Senior, A.W., *Off-Line Handwriting Recognition: A Review and Experiments,* Cambridge University Engineering Department, December 1992.

[3] Jain, Anil K., *Fundamentals of Digital Image Processing,* Prentice-Hall, 1986.

[4] Castleman, Kenneth R., *Digital Image Processing,* Prentice-Hall, 1979.

[5] Duda, Richard O. and Peter E. Hart, *Pattern Classification and Scene Analysis,* Wiley-Interscience

Publications, 1973.

[6] Tou, J.T. and R.C. Gonzalez, *Pattern Recognition Principles*, Addison-Wesley Publications, 1974.

[7] Andrews, Harry C., *Introduction to Mathematical Techniques in Pattern Recognition*, Wiley-Interscience Publications, 1972.

[8] Dougherty, Edward R., *An Introduction to Morphological Image Processing*, SPIE Optical Engineering Press, 1992.

[9] Pratt, William K., *Digital Image Processing,* Wiley-Interscience, second edition 1991.

[10] Ullmanm J.R., "Picture Analysis in Character Recognition", *Topics in Applied Physics Vol. 11: Digital Picture Analysis,* ed. A. Rosen Springer-Verlag, 1976.

[11] Gonzalez, Rafael C. and Paul Wintz, *Digital Image Processing,* Addison-Wesley Publications, 1979.

[12] Cheng, Fang-Hsuan, Wen-Hsing Hsu and Mei-Ying Chen, "Recognition of Handwritten Chinese Characters by Modified Hough Transform Techniques", *IEEE Trans on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 4, April 1989, pp. 429-437.

A Method for Automatic Logo Recognition

Brett Crockett
TRW Avionics and Surveillance Group
Sunnyvale, CA

Abstract

The automatic recognition of logós contained within a document can provide valuable insight into the nature and content of a document. In this paper we present an approach that automatically searches a document, identifying and isolating a set of "candidate logos" and then searches a pre-defined logo database in order to determine whether any of the candidate logos are contained within the database. Techniques have been implemented that allow logo isolation and database matching to be performed quickly and accurately given a database of several hundred logos extracted from a wide-range of low resolution, one-bit black and white scanned document images.

1 Problem Description

Automatic recognition of graphical logos within a document holds potential for identifying high priority document images (based upon content) an order of magnitude faster than traditional character recognition methods. For several years, TRW ASG has focused resources on developing and applying size invariant, rotation tolerant graphical icon recognition algorithms to low resolution document images (100 to 200 dpi). These techniques have been successful with a wide variety of logos and varying quality document images, and have been incorporated in both standalone logo recognition systems as well as comprehensive document understanding systems.

Graphical logo recognition is a document content analysis tool, like character recognition or signature identification, that can be used to automatically identify important documents or prioritize a set of documents containing logos. If logo recognition is incorporated into an overall image processing system, several key processing steps may be performed prior to logo recognition, as shown in Figure 1. As an optional first step, an acquired document image may be enhanced to correct for skew or speckle-noise errors. Following enhancement, the document image may be segmented into a number of distinct regions containing one of the following types of information: ideographic, machine-generated text, handwriting, or other. The segmentation module passes each image region to the appropriate follow-on content recognition tool. Segmented regions containing potential logos are sent to the logo recognition process which identifies known logos of interest. This information can then be passed to a document understanding module which may characterize and/or prioritize each document based on the results of the individual content analysis tools.

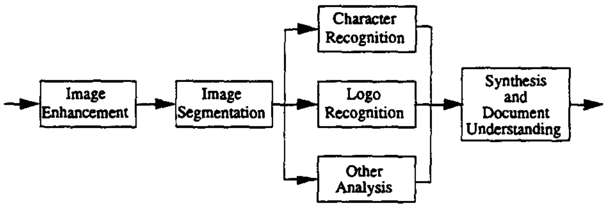


Figure 1: Conceptual system diagram containing a logo recognition module.

While graphical recognition has the potential to be faster than character recognition, logo identification must ideally possess the same robust characteristics required for reliable OCR processing. Graphical recognition algorithms should be tolerant of variations due to:

- image size
- scanner effects (degradations)
- skew (both linear and non-linear)
- image orientation
- noise (bit errors)
- speckle noise
- variations of the same icon/logo
- nearby or overlapping segments

2 Technical Approach

Our general approach to logo recognition is summarized in Figure 2. The three main steps in the logo recognition process are logo extraction (image segmentation), feature calculation, and feature comparison, or matching.

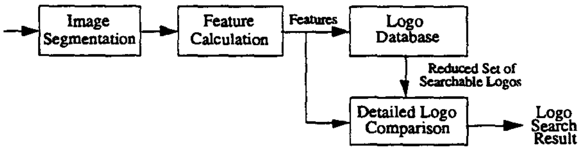


Figure 2: Summary of TRW logo recognition algorithm.

The stages of the logo recognition process are as follows. First, binary graphics thought to contain logos are extracted from digitized input documents in a consistent manner. A set of simple features are then computed from each extracted image for use in reducing the searchable database of possible logo matches. By carefully selecting a set of easily computed yet unique image features, a significant number of logos can be quickly eliminated from the set of possible matches. A more complex set of features are then computed for use in comparing the remaining set of possible matches in the database of known logos. If the known and unknown logos have very similar features based on a series of detailed image comparison algorithms, they are reported as a match.

2.1 Logo Extraction

The logo extraction algorithm searches a scanned document image, determines which regions of the document contain candidate logos and extracts the logos for further processing. The extraction algorithm incorporates horizontal and vertical pixel density measures to identify the position of candidate logos. Density measures were selected for use because during our algorithm development they were shown to accurately separate significant graphical elements from document text and other non-logo elements of documents. Figure 3 shows an example of the density measures for a typical document.

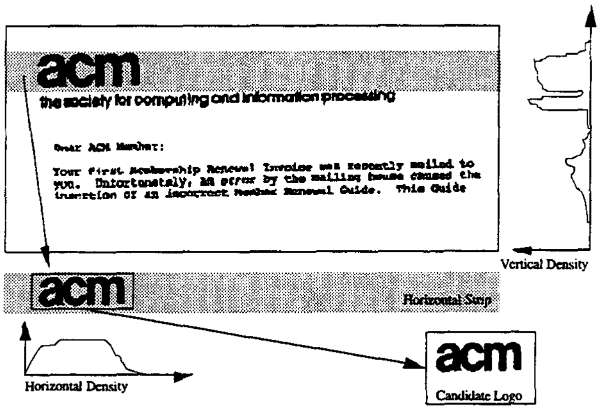


Figure 3: An illustration of the density-based logo extraction method.

The first step of logo extraction is the computation of the document's vertical density. The vertical density is computed by summing the number of black pixels in each scan row. Using programmable density thresholds, horizontal strips of the document are isolated using the vertical density measurement. The horizontal density measurement is then computed by summing the black pixels in each column of each isolated horizontal strip. Using a separate set of programmable density thresholds, candidate logos are identified in each horizontal strip, isolating the position of each logo in the document image.

A set of "intelligent" rules are also used in both of the density measurements to ensure that logos are consistently and correctly extracted. For example, one set of rules is used along with the horizontal density measurement to insure that logos consisting of separated graphical elements are grouped together as one logo.

The second step of logo extraction involves conditioning the logo for use by the feature extraction algorithm. Logo conditioning involves defining the boundaries of the logo (where the logo begins and ends in both the vertical and horizontal directions) so that logos extracted from different documents are as similar as possible. This consistency is crucial to the recognition algorithm, because without it, the pattern matching algorithm described below would not be reliable.

2.2 Feature Calculation

The following four features are extracted from each extracted candidate logo

- aspect ratio
- power
- busyness
- postage stamp.

The first three feature types, aspect ratio, power and busyness, are coarse features used to produce a reduced search set from the overall database. The fourth feature, the postage stamp, is a processed and compressed version of the candidate logo image. The postage stamp feature is used for more detailed logo comparisons on the reduced database search set.

The definition of each of the features used in logo recognition is provided in the following sections. In the following discussions, $I(m,n)$ denotes a two-dimensional array representing a binary image having M rows and N columns. In the cases of aspect ratio and postage stamp calculations, $I(m,n)$ is the extracted logo. In the cases of power and busyness, $I(m,n)$ is a region of the extracted logo.

2.2.1 Aspect Ratio

Aspect ratio, A, was chosen as a feature for database reduction because of its simplicity. Aspect ratio is the ratio of the height of the extracted logo divided by the width:

$$A = \frac{M}{N} \qquad (1)$$

2.2.2 Power

Power, P, is a measure of how many logo pixels are black with respect to the total number of bits (with a black pixel being equivalent to binary 1 and a white pixel being equivalent to binary 0). Power is calculated as the summation of the pixel values divided by the total number of pixels:

$$P = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I(m,n) \qquad (2)$$

2.2.3 Busyness

Busyness, B, is a measure of the amount of pixel to pixel changes with respect to the maximum number of changes possible. It is calculated as the summation of black to white (or 1 to 0) and white to black (or 0 to 1) transitions in the horizontal, vertical, and diagonal directions divided by the maximum number of possible pixel changes:

$$B = \frac{1}{3(M-1)(N-1)} \sum_{m=0}^{M-2} \sum_{n=0}^{N-2} I(m,n) \wedge I(m+1,n) + I(m,n) \wedge I(m,n+1) + I(m,n) \wedge I(m+1,n+1) \qquad (3)$$

where the ^ symbol is used to represent the logical exclusive-or function.

2.2.4 Logo Compression (Postage Stamp)

One of the constraints imposed on the recognition algorithm was size invariance. That is, the algorithm should match two copies of the same logo regardless of the size of either copy. Another constraint is that the algorithm be tolerant to small rotations of the logo due to misalignment of the scanned document. Logo compression addresses both of these constraints. First of all, by compressing all logos to a single standard size, the recognition algorithm is by definition size invariant. Furthermore, a standard-sized, compressed postage stamp logo, provides a convenient way of directly comparing any two logos of different shapes and sizes. The postage stamp also provides rotational tolerance because of the averaging inherent in the compression algorithm. Averaging has been shown experimentally to reduce the effects of rotational distortions as well as scanning noise and decompression errors.

The general idea of the compression algorithm is the same as that of standard sample rate conversion algorithms, only sub-sampling is performed in two dimensions. A logo is compressed by lowpass filtering the image and interpolating the new sample values at the new sample spacing. The result is a gray scale image rather than a binary image.

For efficiency, the compression algorithm performs the lowpass filtering and interpolation simultaneously. First, the compression algorithm constructs a continuous image from the discrete pixels by using a straight line approximation between pixels. Integrating the continuous image over the new pixel spacing yields the new pixel values. Figure 4 illustrates the algorithm on a single row from a logo image. In the figure, each dot represents a pixel. Those with vertical lines represent black pixels; those without represent white pixels. The area of the shaded regions is the new pixel value, and the length of the shaded regions is the new pixel spacing. The area is found simply by using the trapezoid rule for integration. After compressing each row of the logo, the algorithm compresses each column in the same way.

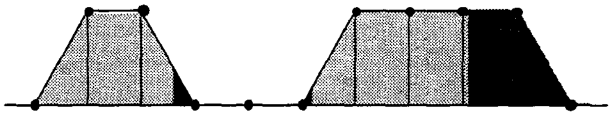


Figure 4: An illustration of the logo postage stamp compression algorithm.

The following discussion provides details on the implementation of the postage stamp algorithm. Mathematically, we can represent a logo as a two dimensional array of pixels having M rows and N columns as

$$L(m,n) \qquad m = 0, 1, \ldots, M-1; n = 0, 1, \ldots, N-1. \qquad (4)$$

The compression algorithm takes the M x N logo and compresses it into a postage stamp, $P(i,j)$, containing I rows and J columns. To do so, the compression algorithm first compresses the logo horizontally to make an intermediate value, $Q(m,j)$. It then compresses $Q(m,j)$ vertically to obtain the postage stamp. Initially, the compression algorithm pads the logo with zeros (or white space) around the perimeter of the logo to provide a fixed reference and continuity at the edges of the logo.

$$L(m,n) = 0 \qquad m = -1, M \text{ or } n = -1, N. \qquad (5)$$

Horizontal compression is performed on the logo by converting the rows of the logo into a continuous function, $l(m,h)$, and integrating over each row. The result is the intermediate value $Q(m,j)$ defined by

$$Q(m,j) = \frac{1}{N} \int_{\frac{j(N+1)}{J} - 0.5}^{\frac{(j+1)(N+1)}{J} - 0.5} l(m,n)\, dh \qquad \begin{matrix} j = 0, 1, \ldots, J-1 \\ m = -1, 0, \ldots, M \end{matrix} \qquad (6)$$

where

$$l(m,h) = L(m,\underline{h}) + (h - \underline{h})(L(m,\underline{h}+1) - L(m,\underline{h})) \qquad (7)$$

and

$$\underline{h} = \text{largest integer} \leq h. \qquad (8)$$

Vertical compression is likewise performed by converting the columns of $Q(m,j)$ into a continuous function, $q(v,j)$ and integrating over each column. The postage stamp, $P(i,j)$, results.

$$P(i,j) = \frac{1}{M} \int_{\frac{i(M+1)}{I} - 0.5}^{\frac{(i+1)(M+1)}{I} - 0.5} q(v,j)\, dv \qquad \begin{array}{l} i = 0, 1, ..., I\text{-}1 \\ j = 0, 1, ..., J\text{-}1 \end{array} \qquad (9)$$

where

$$q(v,j) = Q(\underline{v},j) + (v - \underline{v})(Q(\underline{v}+1,j) - Q(\underline{v},j)) \qquad (10)$$

and

$$\underline{v} = \text{largest integer} \leq v. \qquad (11)$$

The resulting compressed postage stamp logo , $P(i,j)$, provides a way to directly compare two logos independent of the size of either logo.

2.3 Feature Comparison

In order to compare the computed features of unknown logos and those contained within the predefined logo database, a set of distance measures have been defined.

For the aspect ratio, busyness, and power, the distance has been defined as the absolute difference between the features. The aspect ratio, busyness, and power distances are, therefore,

$$d_{AR} = | AR_k - AR_u |, \qquad (12)$$

$$d_B = | B_k - B_u |, \qquad (13)$$

$$d_P = | P_k - P_u |. \qquad (14)$$

The subscripts, k and u, denote the features of the known and unknown logos. The distance between postage stamps is the sum of the squared differences between pixels in the postage stamps.

$$d_{PS} = \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} (P_k(i,j) - P_u(i,j))^2 \qquad (15)$$

Since a postage stamp can be viewed as an $I \times J$ element feature vector, this distance is also the squared Euclidean distance between feature vectors, P_k and P_u, in the feature space.

In order to compare unknown input logos and those contained within the logo database the features for both sets are calculated. A known logo, stored in the logo database tentatively matches an unknown input logo if a distance measure falls within a predefined feature threshold. A summary of the thresholds used in the recognition algorithm and their meaning is presented in Table 1.

Table 1: Recognition Algorithm Thresholds

Threshold	Symbol	Description
Aspect Ratio	*at*	tentative match if $d_{AR} \leq at$
Busyness	*bt, bn*	tentative match if for (*n-bn*) of *n* sectors, $d_B \leq bt$
Power	*pt, pn*	tentative match if for (*n-pn*) of n sectors, $d_P \leq pt$
Postage Stamp	*dt*	tentative match if $d_{PS} \leq dt$

For each of the features, the threshold is simply a number. During algorithm development it was discovered that by evenly dividing the logo into n sectors and computing the busyness and power separately for each, logo recognition accuracy is greatly improved. There are, therefore, n busyness distances and n power distances to compare. Rather than define a threshold for each sector, a single threshold was defined to which the distances from each sector are compared. In order to address the use of a single threshold, a voting scheme was implemented. If the distance threshold is met for a specified number of sectors, the logos tentatively match.

A known logo in the database is considered to match an unknown logo if all of the distance measures meet their respective thresholds.

3 Results

The logo recognition algorithm performance is outlined below in Sections 3.1 and 3.2.

3.1 Pd and Pfa

After determining the optimal set of thresholds, the probability of detection, P_d, and the probability of a false alarm, P_{fa}, were measured using a set of "real-world" test logos. In the test set, 187 logos were stored in a database as known logos. In addition, 252 logos were obtained to be compared against the database of 187 logos. The 252 logos consisted of modified versions of the logos contained in the database (rescanned at varying size and quality) as well as logos not stored in the database. Of the 252 comparison logos, 128 were known to match logos in the database. The recognition algorithm correctly identified 126 of the known matches for a P_d of 98%. In addition, the algorithm incorrectly matched 11 of the 252 unknown logos with logos in the database for a P_{fa} of 4%. These results are summarized in tables 2 and 3.

Table 2: Probability of Detection Summary

Database Matches	Hits	P_d
128	126	98%

Table 3: Probability of False Alarm Summary

Unknown Logos	False Alarms	P_{fa}
252	11	4%

3.2 Benchmarks

While evaluating performance measures for the recognition algorithm, we benchmarked the creation of the logo database and the processing of unknown logos using a Sun Microsystems SPARCstation 2. The major factor in both processes was the feature extraction time. In our tests, each logo required an average of 450 ms to extract the aspect ratio, busyness, power, and to compress the logo into a postage stamp. The average time per logo to create a database was 490 ms. The average time to compare an unknown logo against the database was 530 ms per logo (for a master database of 187 logos). Because of the structure of the database, the actual search time will only increase by the base two logarithm of the size of the database. For example, a 1024 entry database would approximately twice as long per logo to search as a 32 entry database. The search time is also dependent on the thresholds since the number of postage stamp comparisons (a relatively complex operation) is dependent on the database reduction the other features provide. Based on our observations, it is estimated that comparing a single logo against a 1000 entry database should only take an average of about 750 ms.

4 Future Goals

TRW ASG's logo recognition algorithms have been shown to provide a powerful and accurate method of identifying graphical logos or icons within a scanned image document. However, there remain areas of study and analysis not performed in the initial experiments that could provide insight into the algorithms and allow them to be refined to provide even better results. Two of these areas are described below.

4.1 Low Quality Images

For low quality images (which a large number of scan errors, speckle noise and image distortion) many of the characteristics that traditional logo recognition algorithms exploit may be obliterated. New techniques could be developed that concentrate on analyzing logo features that remain intact even for low quality images.

In addition, initial investigation has shown the TRW ASG logo recognition algorithms to perform better when the user-defined algorithm parameters are manually modified in relation to the noise within a document image. The incorporation of an image-quality measure could be exploited to improve the algorithms' performance by automatically adjusting the parameters to compensate for different noise levels.

4.2 Multiple logo versions

Our current logo recognition algorithm treats each logo in the comparison database independently. Using multiple versions of the same logo within the master database could improve logo recognition accuracy. Many times the same company or organization has similar but different versions of the same logo. Identifying these as being the "same" logo could be very useful to the logo recognition algorithm. A similar effect is seen when the same logo appears both large and very small. Many of the features of the small logo are lost and therefore matching it with the large one is not easy, especially when the document image is scanned at low resolutions. In cases where multiple copies of a logo are available, they could be used to compute averaged image information, thereby enhancing the recognition process.

Enhancement for Imaged Document Processing

Victor T. Tom Paul W. Baim
Atlantic Aerospace Electronics Corporation
470 Totten Pond Road
Waltham, Massachusetts 02154

Abstract

A set of tools for analyzing and correcting degradation in imaged text documents to improve OCR performance is described. A number of metrics have been developed that detect, identify, and characterize degradation types based on variations in fundamental characteristics of typical text. The current focus is on degraded 10 and 12 point Times Roman and Helvetica type rendered at 100x200 dpi. Statistical, morphological, and connected component techniques contribute to the analysis. Results from the analysis process are used to tailor a reconstruction sequence that normalizes the text image. OCR scores before and after normalization show an average 37% improvement in word error rate for thinned text but no improvement for thickened text.

1 Introduction

The performance limitations of OCRs against low resolution and/or degraded quality text are well known [1]. OCR performance rapidly degrades even when the image defects appear minimal to the naked human eye. One approach to improving OCR performance is to extend the training set of characters for OCR engines using a rigorous defect model to include all the possibilities [2]. Classification can then be performed using decision trees (preclassifiers) where up front classifiers are less accurate but only need to determine coarse categories and downstream classifiers are more accurate to perform the final classification. The drawback of this approach is the computational requirement.

A second approach, which we are investigating, is to modify the degraded data so that the filtered result looks like a nominal font, free of most image defects. This approach requires that we be able to characterize the typical image defects that would cause performance degradation in OCR and be able to filter them out. We make the important distinction that we do not attempt to quantify image distortions so that we can restore the imaged document to its original form, but instead, we only transform the characters to a nominal-looking font which can be interpreted by the OCR. In this work we assume that the text portion of the documents have already been blocked out and that we are only processing the text areas.

A set of modular software tools have been implemented and coded conforming to ANSI C standards. The tools currently run on SUN SPARC™ workstations and can process TIFF format images.

The following sections of this paper present examples of document distortions investigated, a system overview, preprocessing functions to remove global document defects, metrics used to quantify text characteristics, enhancement algorithms designed to improve text readability and a discussion of overall OCR performance results.

2 Text Distortions

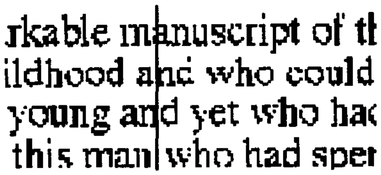
We used the DISTORT modeling software from MITEK systems to simulate scanning defects on 12 and 10 pt Helvetica and Times Roman text samples. The defects that we have been investigating include: vertical scanning defects, speckle, light bleed through, text blurring (thickening) and text thinning. Our focus was on low resolution documents sampled at 100x200 dpi.

A range of vertical scanning line defects was simulated for our analysis and testing. Since this is a sensor defect and not inherent in the document, scan line defects are always perpendicular to the orientation of the scanning direction, normally vertical. An example of a vertical line defect, which could be caused by a clogged detector, is shown in Fig. 1. Vertical lines were simulated with a range of probability and lengths resulting in both solid and broken line forms. Line widths were confined to one and two pixels.

Speckle was simulated using various speckle blob sizes and probabilities providing a range of speckle distortions. Figure 2 displays two examples of speckle, one denser and more uniform (top) and the other less dense but with a wider range of speckle size (bottom).

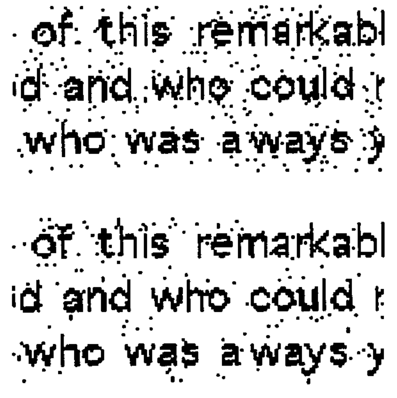
Examples of degraded 12 pt text are shown in Fig. 3 and 4. The examples in Fig. 3 and 4 include a magnified view of a pair of words that have been moderately and severely thinned and blurred. For the thinned case the degradation causes the characters to break. Fonts which exhibit tapering thickness strokes, such as Times Roman, are more susceptible to changes in character topology when subjected to thinning degradation.

The blurring mode creates thickening of character strokes, fringing at the bottom and left sides of characters, filled characters and joined adjacent characters. Some of these effects can be seen in Fig. 3. Times Roman is more susceptible to character joins in the vicinity of serifs.



rkable manuscript of tl
ildhood and who could
young and yet who ha(
this man who had sper

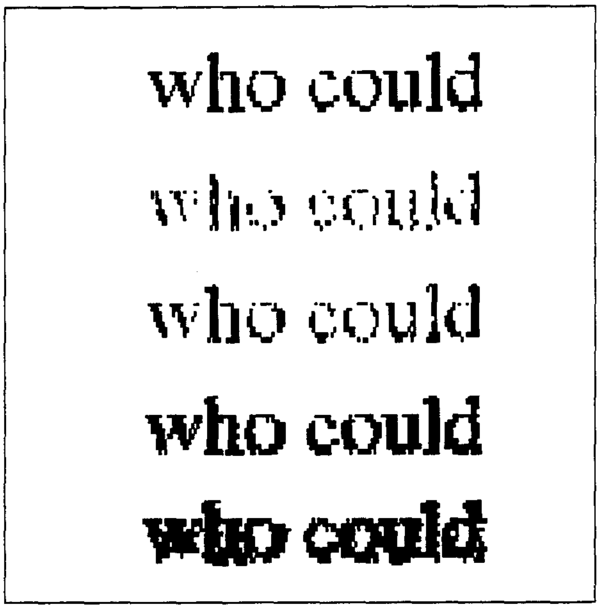
Figure 1: Vertical scan line defect.



of this remarkabl
id and who could t
who was aways y

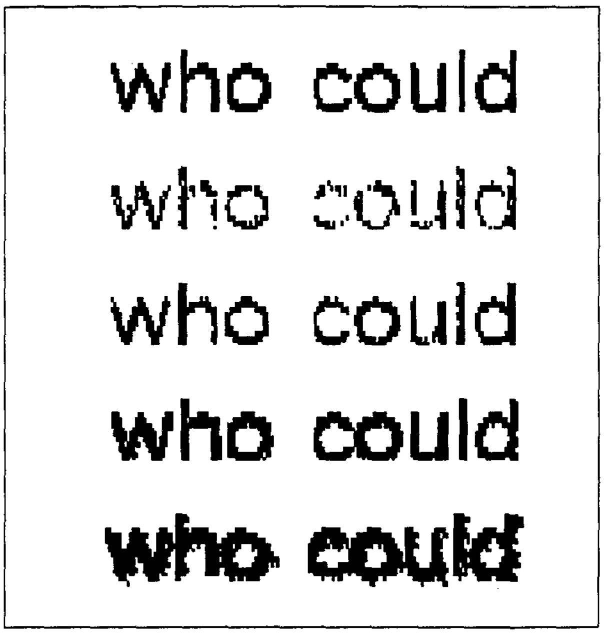
of this remarkabl
id and who could t
who was aways y

Figure 2: Two levels of speckle defects.



who could
who could
who could
who could
who could

Figure 3: Nominal, two thinned and two blurred versions of 12 pt. Times Roman text.



who could
who could
who could
who could
who could

Figure 4: Nominal, two thinned and two blurred versions of 12 pt Helvetica text.

3 System Overview

Our overall approach is based on identifying, quantifying and removing document degradation in successive stages. The processing flow is presented in Fig. 5.

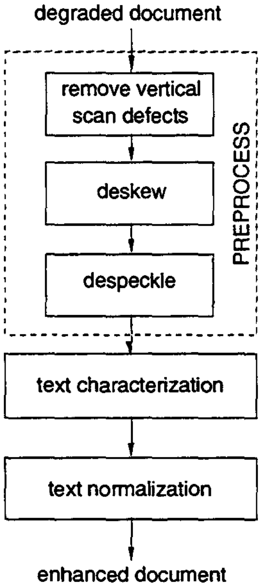


Figure 5: Flowchart for document enhancement.

The earlier stages of processing require the least amount of information about the document. As each level of processing is performed and degradation is removed, subsequent analysis steps have increased accuracy in determining text characteristics. Although the current results reflect this approach as being performed on an entire document, the software has been developed to facilitate processing portions of a document and allowing it to adapt to non uniform degradation.

3.1 Preprocessing Functions

Preprocessing is composed of three functions; vertical scan defect removal, deskewing and despeckling. Deskewing was not part of our research and is not described here.

The vertical scan defect remover is designed to remove thin, vertical black lines from document images. To detect the presence of vertical scan defects, a vertical histogram is computed over the entire document; that is, we add up the number of black pixels in each column. Vertical lines appear as anomalous spikes in the histogram as evidenced in Fig. 6. The median and standard deviation are computed for the profile. If any column value is greater than the median plus twice the standard deviation, and there are no more than two adjacent columns satisfying that criteria, then that column is declared to be a scan defect.

The processing for scan defects includes removing the column completely and then restoring the missing values using a form of *image continuation* [3] as shown in Fig. 7.

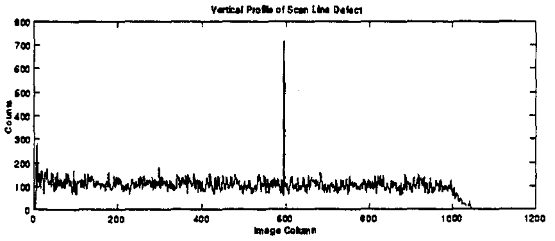


Figure 6: Vertical histogram for scanned document with vertical line defect.

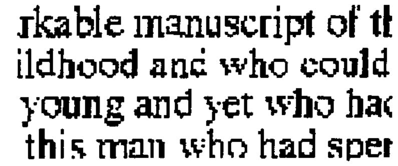

rkable manuscript of tl
ildhood and who could
young and yet who hac
this man who had sper

Figure 7: Vertical scan line defect removed from Fig. 1

The despeckler function measures the distribution of object sizes in a document image, determines an appropriate threshold, and then removes objects less than the threshold size. The algorithm first creates a histogram of object sizes, where each object is determined based on the 8-connectedness property. Speckled text documents display a peak at the extreme low end of the distribution as shown in Fig. 8, whereas unspeckled documents have few small sized objects other than punctuation marks. To determine the size threshold, we currently use the knee of the distribution. To remove speckles, all objects smaller than the size threshold are removed. Despeckled text examples are presented in Fig. 9. Speckle that is connected to valid characters are not eliminated at this point, but can be removed at a later defringing stage.

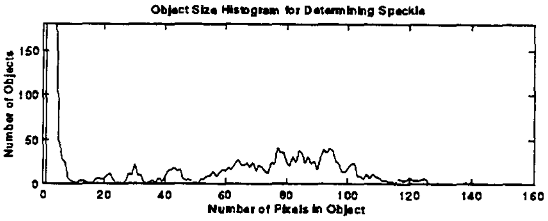
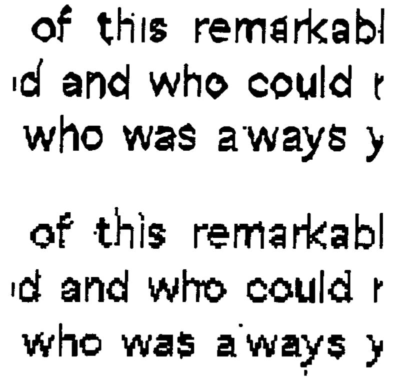


Figure 8: Speckle peak in object size histogram


of this remarkabl
id and who could r
who was a'ways y

of this remarkabl
id and who could r
who was a'ways y

Figure 9: Despeckled examples from Fig. 2.

3.2 Text Characterization

Image metrics were developed to: detect the presence of text defects; determine which appropriate filtering sequence to apply; and set filtering parameters to ameliorate those defects.

Text characterization is designed to quantify fundamental text characteristics in text images which have been blurred or thinned by poor contrast and resolution changes. These characteristics include the horizontal and vertical stroke widths, the amount of fringing in the vertical and horizontal directions, the median height of the text lines in the image, the density of black pixels in the text lines, and the density of "holes" in the text lines. Holes are defined as completely enclosed regions inside character loops.

Stroke width measurements are determined from a peak analysis of run length histograms. The dominant width of horizontal strokes corresponds to the location of the first major peak in the vertical run length histogram. The average horizontal stroke width equals four pixels for the data shown in Fig. 10. Conversely, the average width for vertical strokes (three pixels) is computed from the horizontal run length histograms (Fig. 11).

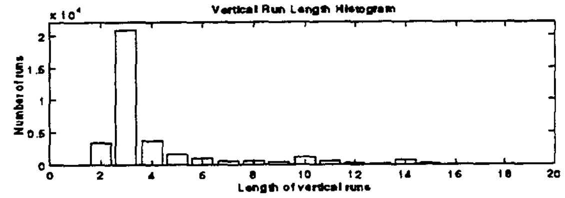


Figure 10: Average horizontal stroke width indicated in vertical run length histogram.

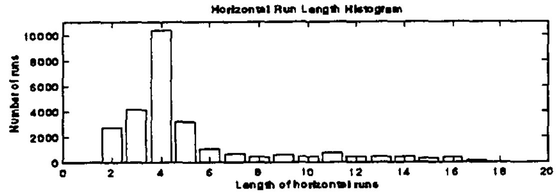


Figure 11: Average vertical stroke width indicated in horizontal run length histogram.

To detect fringe effects in thickened text, a separate measurement based on the run length histograms is used. Fringes are typically one pixel wide and show up in the first bin of the run length histogram. Fringes can be detected by measuring the ratio of the first bin to the bin corresponding to the stroke width. Fringes in thinned text show up as white fringes on the edges of characters. By using reverse contrast, a similar fringe measurement can be made.

A coarse assessment of whether text has been thickened or thinned can be inferred from the density of black pixels and holes within each line of text. Text line heights and locations are determined by thresholding the horizontal histogram, which is computed by summing up the pixels in each row. Using the location of text lines we compute the density of pixels and holes. Holes are determined by detecting regions within characters that are not connected to the background as shown in Fig. 12.

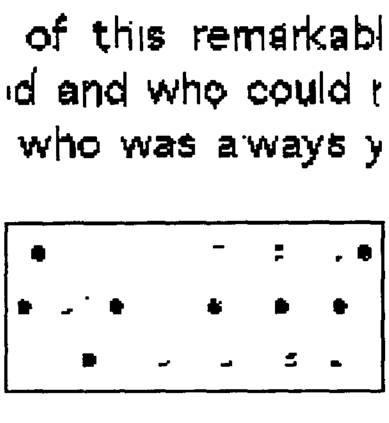


Figure 12: Example text (top) and corresponding character holes

Samples of thinned and thickened text are plotted versus pixel and hole density in Fig. 13. The symbol *N* indicates a sample of undegraded 12 pt Helvetica text. Normal looking text clusters near *N*. Thickened text displays higher pixel density and lower hole density due to the gradual filling of holes. Thinned text exhibit lower pixel density and low hole density due to character breakage. Slightly thinned characters can have low pixel density with high hole density if character breakage does not occur. Based on these observations, a partitioning of the pixel vs. hole density space was determined in order to classify severely thinned and thickened text samples. Samples falling in the lower left are considered to be severely thinned, whereas samples falling in the lower right are likely to be severely thickened. Additional stroke information was used in conjunction with pixel and hole density to further determine normal looking text and moderately thinned or thickened text. These classifications guide the selection of the text normalization process.

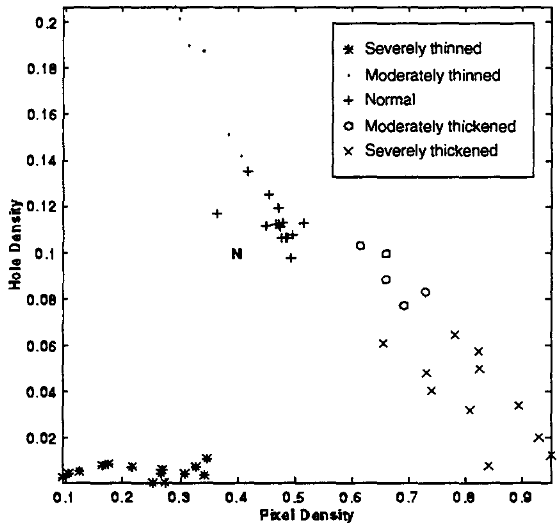


Figure 13: Coarse text characterization samples plotted versus hole and pixel density

3.3 Text Normalization

In this section we discuss the ensemble of text normalization algorithms that we use to modify text characteristics. Algorithms include character bridging, separation, defringing, thinning and thickening.

Text normalization is accomplished for each of the five classes of text characterization decisions. The enhancement steps for each of these regimes is shown in Table 1. Although the component enhancement steps for the severely degraded cases are the same as for the moderately degraded cases, the degree of filtering (bridging, defringing, etc.) are different and determined by the text characterizations.

Table 1: Enhancement steps for each characterization of text.

Text characterization	Enhancement steps
severely thinned moderately thinned	• bridging • defringing (white) • sttroke normalization
normal	• none
moderately thickened severely thickened	• character separation • defringing • stroke normalization

All of the enhancement filters are based in principle on morphological filters [4]. Character bridging is based on morphological closing filters. The extent of the bridge is governed by whether the text is determined to be severely or moderately thinned. Character separation is achieved by hit-or-miss filters designed to detect narrow character joins near the midheight line of the text. Currently this operation is fairly conservative and could be improved using any number of character separation algorithms [5]. Defringing is based on hit-or-miss filters where the filter kernels are matched to the expected fringe elements (white fringes correspond to the thinned case). The defringing filter is similar to the optimal FAX restoration filters [6].

Stroke width normalization is performed last. The computed line height is used to estimate a nominal vertical and horizontal stroke width. Morphological erosions or dilations are performed to decrease or increase the stroke widths to achieve the desired nominal measurements.

Examples of automatically processed text are shown in Figs. 14 and 15. As seen in these examples, all of the text exhibit similar stroke widths and fewer character defects than in the original degraded examples.

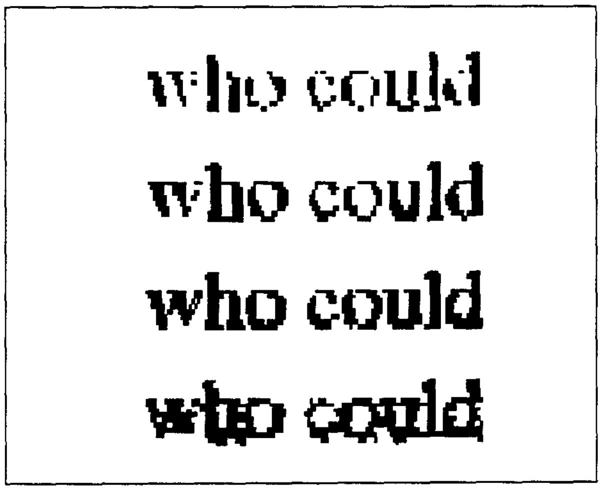


Figure 14: Normalized Times Roman text.

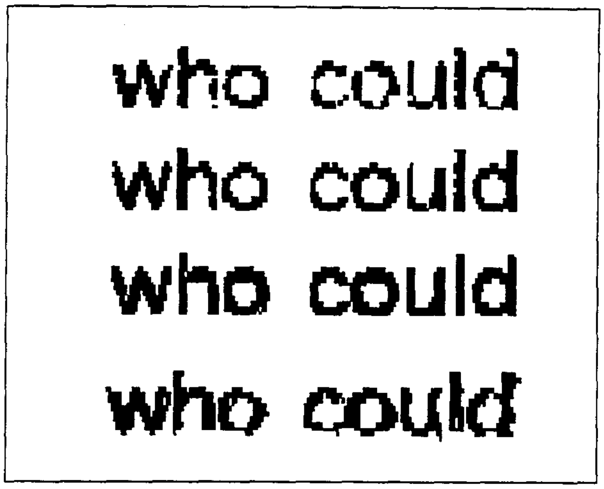


Figure 15: Normalized Helvetica text.

4 OCR Experiment Results

A Calerra OCR engine running on a SUN SPARC™ workstation was used to evaluate the effectiveness of our image enhancement approach. Sample 10 and 12 pt Times Roman and Helvetica text passages were prepared and the DISTORT software was used to generate a wide range of distorted samples for evaluation. The word error scores were recorded for the distorted and the enhanced samples and are plotted in Figs. 16-20.

To facilitate understanding of the regimes in which enhancement is effective, we plotted the scores according to the four categories determined by the text characterization procedure. From Figs. 16 and 17, one can see that the normalization for moderately and severely thinned text has a significant impact on error performance. On average, we achieved a 37% reduction in word errors over all thinned text samples. Some samples that did not seem to improve contained 10 pt characters, which may have been distorted beyond recognition.

Thickened text, on the other hand, did not fare as well. There was no measurable reduction in word errors for either the slightly thickened or severely thickened category. Improvements can be made in refining our algorithms for separating joined characters as well as for thinning thickened characters [7].

Despeckling also tended to improve OCR performance. Modest amounts of speckle rapidly degrade OCR performance. As seen in Fig. 20, significant word error reduction is achieved when despeckling is applied to 12 pt text. The lack of improvement for 10 pt text can be explained. Digitization of 10 pt characters often results in broken characters, whose small segments resemble large sized speckle. The current algorithm is too generous when it sets its size threshold and that is currently under review.

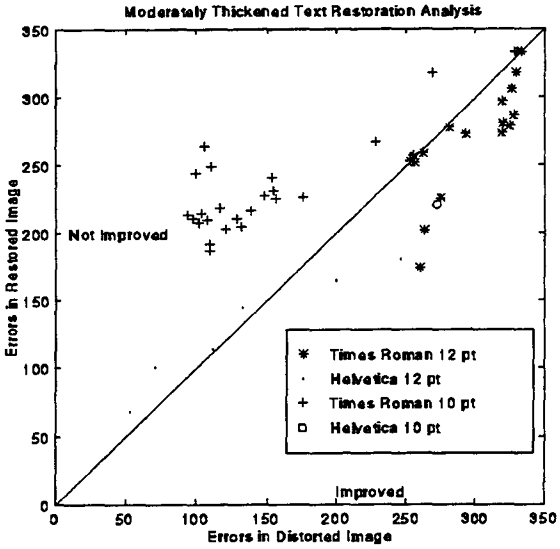


Figure 16: Comparison of OCR performance for moderately thickened text

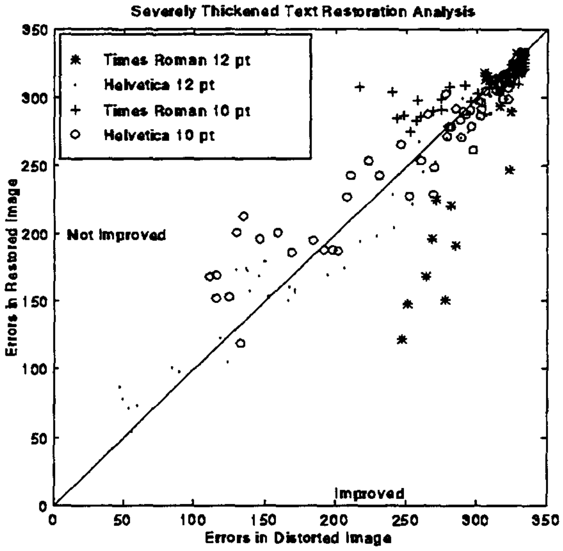


Figure 18: Comparison of OCR performance for severely thickened text.

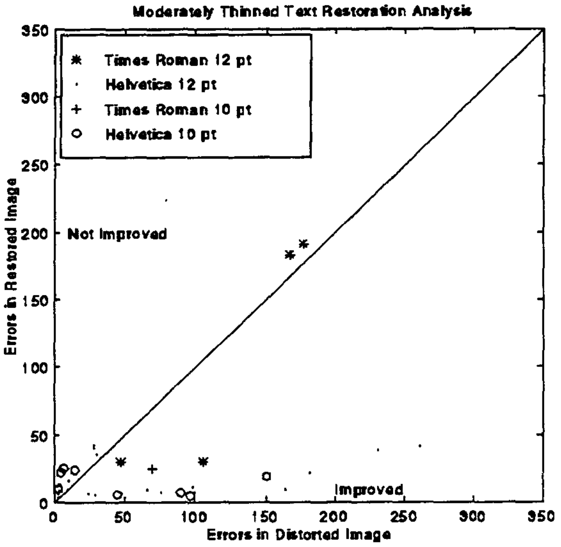


Figure 17: Comparison of OCR performance for moderately thinned text.

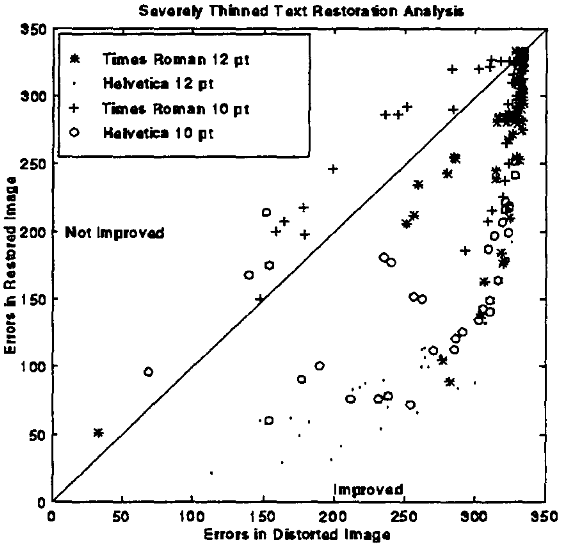


Figure 19: Comparison of OCR performance for severely thinned text

An additional explanation of why the enhanced thickend text has more errors can be due to the fact that Times Roman font has tapered character segments. What we classify as moderately thickened is actually thinned in one dimension and thickened in the other. For example, the cluster of 10 pt Times Roman examples in the top center of Fig. 16 have horizontal character segments are thin and broken, but the vertical strokes are thickened. In the present restoration scheme, no bridging of character breaks is performed since these examples have been characterized as moderately thickened.

This suggests a characterization scheme that factors in the two dimensions separately and a restoration scheme which is different in the way it handles vertical and horizontal strokes.

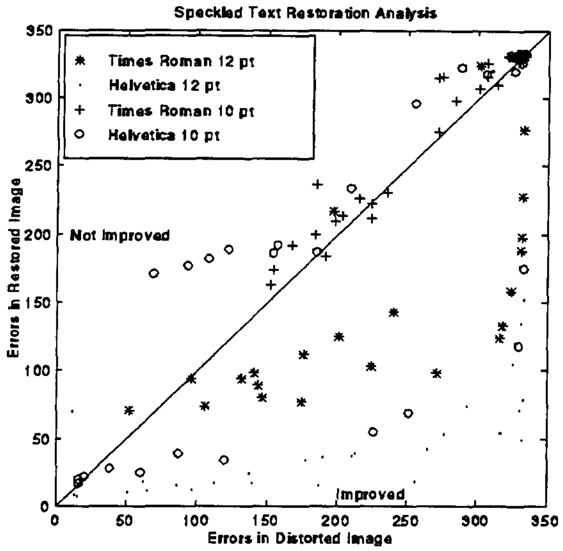


Figure 20: Comparison of OCR performance for speckled text.

5 Summary and Future Work

A versatile set of tools for measuring and modifying text characteristics has been described and implemented. An overall system has been assembled to automatically enhance degraded documents of varying fonts and defect level. Experiments with commercial OCR software indicate significant performance improvements possible for documents with speckles or thinned text when they are automatically enhanced. The results for thickened text currently show no improvement, but with additional refinement in the way we handle the horizontal and vertical dimensions separately that we should see improvement in future. We also plan to implement additional published character breaking and thinning algorithms.

Future work in this research is aimed at determining the ability of this enhancement approach to adapt to different portions of a non uniformly degraded document. We will determine a minimum size for a text block in order to generate a statistically significant character measurement. We will also examine the ability of this technique on a non Roman alphabet.

Acknowledgments

The authors would like to thank Kat Daley and Beato Cheng for providing the algorithm implementations and experimental runs. We also thank Alan Ratner for his support and guidance in this program.

References

[1] *Proceedings of the IEEE*, Special Issue on OCR, July, 1992.

[2] H.S. Baird, Document Image Defect Models and their Uses, in *Proc. 2nd International Conference on Document Analysis and Recognition*, Tsukuba Science City, Japan, Oct. 1993, 62-67.

[3] N. Billawala, P.E. Hart and M. Peairs, Image Continuation, in *Proc. 2nd International Conference on Document Analysis and Recognition*, Tsukuba Science City, Japan, Oct. 1993, 53-57.

[4] E.R. Dougherty, *An Introduction to Morphological Image Processing*, (SPIE Optical Engineering Press, Bellingham, Washington, 1992).

[5] Y. Lu, On the Segmentation of Touching Characters, in *Proc. 2nd International Conference on Document Analysis and Recognition*, Tsukuba Science City, Japan, Oct. 1993, 440-443.

[6] J.C. Handley and E.R. Dougherty, Optimal Nonlinear Fax Restoration, *Proc. in SPIE:* **2181**, Document Recognition, San Jose, CA, (1994) 232-235.

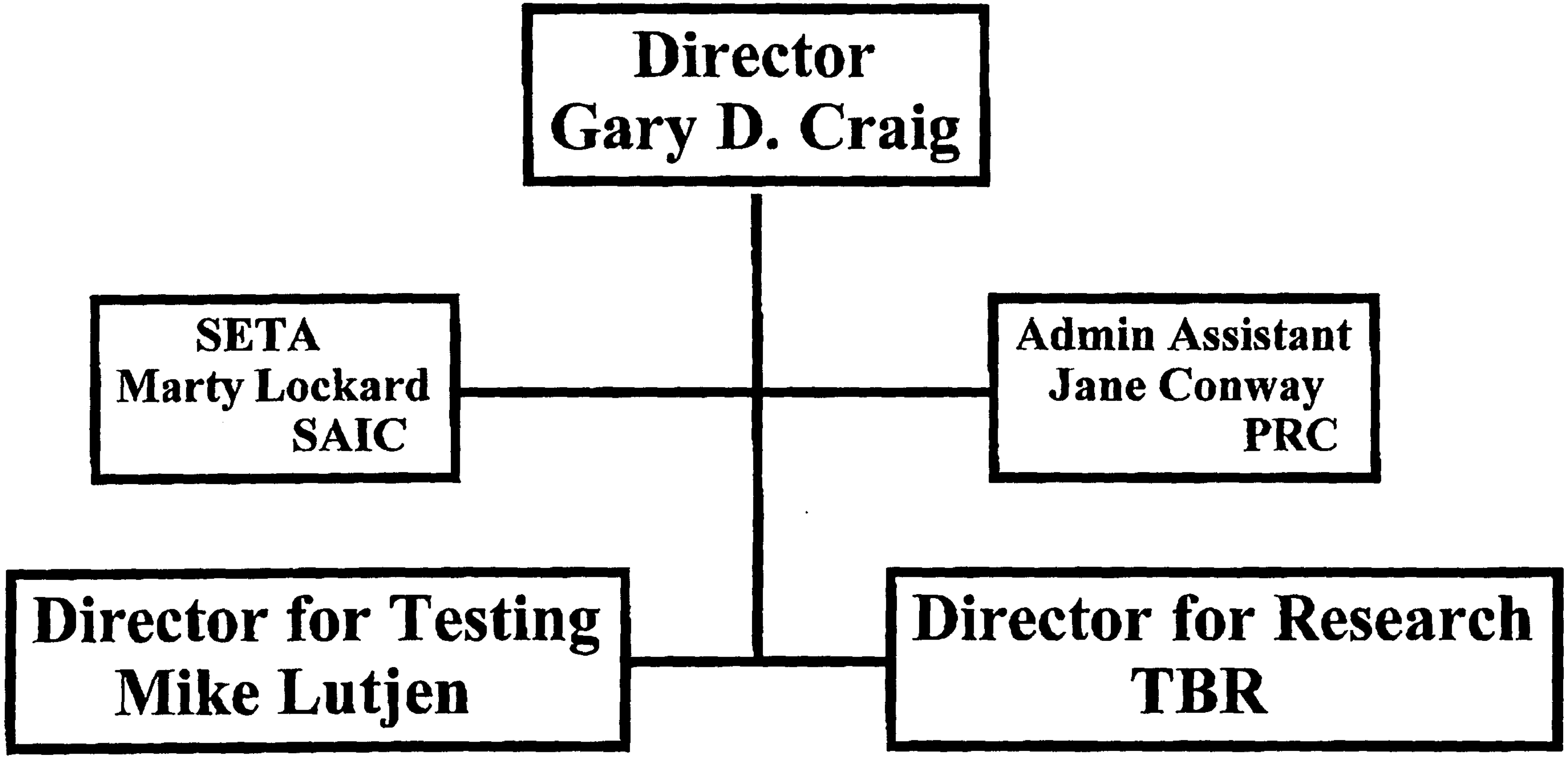
[7] L. Lam, S.W. Lee and C.Y. Suen, Thinning methodologies - A comprehensive survey, *IEEE Trans.PAMI*, **14**, no.9, 1992, 869-885.

Evaluation

Federal Intelligent Document Understanding Laboratory

(FIDUL)

Federal Intelligent Document Understanding Laboratory



Director
Gary D. Craig

SETA
Marty Lockard
SAIC

Admin Assistant
Jane Conway
PRC

Director for Testing
Mike Lutjen

Director for Research
TBR

Tysons International Plaza II
1921 Gallows Rd.,Suite 800
Vienna, VA 22182

Phone # (703)827-2220

Fax # (703)827-5730

Mission Statement

The mission of the ORD Federal IDU Laboratory is to conduct research, develop test methodologies, assess product development, promote standards, and foster technology transfer for Intelligent Document Understanding Technologies.

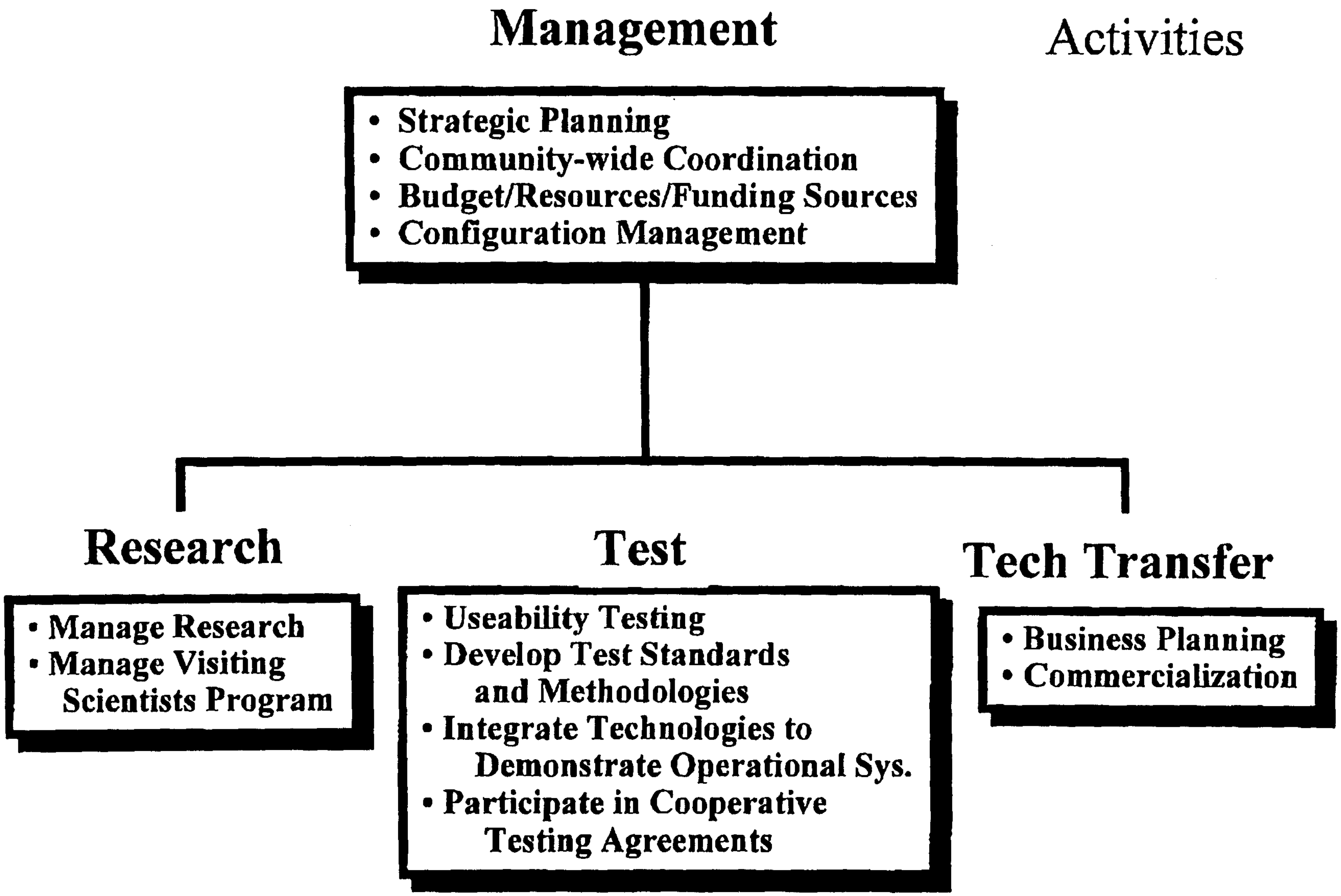
STRATEGY FOR FEDERAL IDU LABORATORY

Objectives

- Maintain world-class research program that advances state-of-the-art IDU technology.
- Establish a technical assessment program that:
  - Emphasizes useability testing,
  - Evolves requirements,
  - Guides research,
  - Provides decision data to transfer technology,
  - Involves end users,
  - Forms partnerships with other testing facilities.
- Promote standards for IDU technology.
- Develop a process and establish partnerships to transfer affordable IDU technologies to end users

Business Areas

Activities



Management

- Strategic Planning
- Community-wide Coordination
- Budget/Resources/Funding Sources
- Configuration Management

Research

- Manage Research
- Manage Visiting Scientists Program

Test

- Useability Testing
- Develop Test Standards and Methodologies
- Integrate Technologies to Demonstrate Operational Sys.
- Participate in Cooperative Testing Agreements

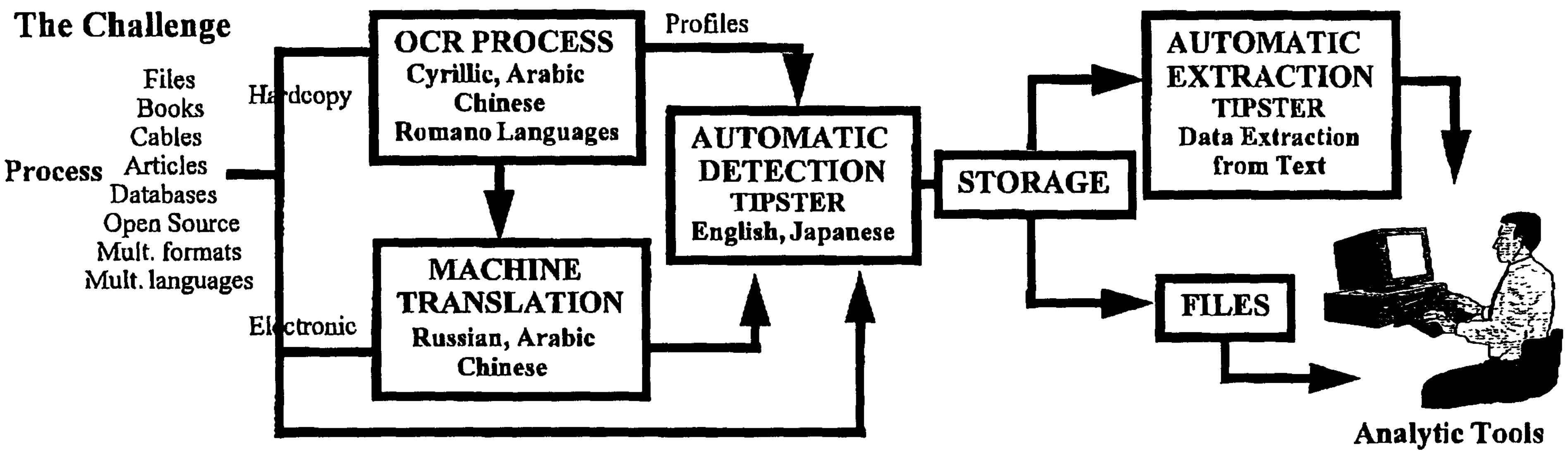
Tech Transfer

- Business Planning
- Commercialization

Information Sciences Research Division

Information Sciences Research Group

STRATEGY FOR FEDERAL IDU LABORATORY



The Challenge

Process

Files
Books
Cables
Articles
Databases
Open Source
Mult. formats
Mult. languages

Hardcopy

Electronic

OCR PROCESS
Cyrillic, Arabic
Chinese
Romano Languages

MACHINE TRANSLATION
Russian, Arabic
Chinese

Profiles

AUTOMATIC DETECTION
TIPSTER
English, Japanese

STORAGE

AUTOMATIC EXTRACTION
TIPSTER
Data Extraction from Text

FILES

Analytic Tools

Evolving the User Environment

- Directed Research
  - User Focused Testing
    - End-to-End Process
      - Collaborative Effort

Program Plan

	FY- 95	FY-96	FY-97
Management	-ID Strategic Partnerships -IDU Architecture Requirements Plan -CM Plan -Security Plan -Community Coordination -Conferences & Symposiums -Assess Research & Test Programs	-FY97 Strategic Plan/Budget -Requirements Tracking -Configuration Management -Community Coordination -Conferences & Symposiums -Assess Research & Test Programs -Develop Funding Sources	-FY98 Strategic Plan/Budget -IDU Virtual Lab Extension Strategy -Configuration Management -Community Coordination -Conferences & Symposiums -Assess Research & Test Programs -Reassess Lab Mission
Research	-Arabic OCR Demo Prototype -Visiting Scientists Prog. Plan -IDU Technology Data Base Plan	-Chinese OCR Demo Prototype -Visiting Scientists (TBR) -IDU Architecture TEMs -Initial IDU Technology Data Base	-Cursive OCR Demo Prototype(TBR) -Language Independent Character Recognition Engine(TBR)
Test	Develop Test Program -Establish Test Facil. -Develop Test Methodology & Validation 1) Cyrillic 2) Chinese	-TIPSTER Test Support -Arabic OCR Testing -ECI Chinese OCR Test -Establish Testing Partnerships -Useability Testing	-Cursive OCR Test (TBD) -MT Test Support (TBR) -Testing Conferences & Symposiums
Tech Transfer	-Tech Transfer Course Attendance -Survey Tech. Transfer Strategies	-ROI Definition/Strategy Plan -Business Strategy/Plan -Establish Partnerships for Tech. Transfer	-FIDUL Tech/Transfer Conferences (TBD) -Applications Database

Overview of Document Image Analysis Research at ISRI

T. Nartker J. Kanai K. Taghva S. Rice
Information Science Research Institute
University of Nevada, Las Vegas
Las Vegas, Nevada 89154

1 Introduction

The Information Science Research Institute (ISRI) at UNLV was formed in 1990 to conduct applied research and testing programs focused on OCR and Information Retrieval. The OCRWM program of the U.S. Department of Energy had, and still has, need for improved technology to convert large quantities of machine-printed documents for use in text retrieval databases.

In 1989, the state-of-the-art of technologies for converting machine-printed documents, and especially scientific and technical documents, was not well understood. The overall recognition accuracy that could be expected of existing devices was unknown. The relationship between marked characters in the output and the best accuracy achievable after manual correction was unknown. The comparative accuracy of the automatic zoning ability of existing devices was unknown. Although trade journals frequently published articles showing the results of comparative tests, they seldom used more than 20 or 30 pages of test data.

At the same time, the relationship between OCR accuracy and retrieval effectiveness was unknown. No experimental data, for any size database, were available to indicate how precision and recall were affected by OCR accuracy. No data were available concerning how different retrieval models were affected by OCR accuracy. Finally, no OCR systems were available that made use of "global" knowledge from documents (and collections of documents) as an aid to error correction.

ISRI has established programs that have shed light on all of these topics. Our programs are:

1. an annual OCR technology assessment test program,
2. a program of applied research in document analysis, and
3. a program of applied research in information retrieval.

There is much interaction between these programs. In this presentation, we give an overview of some of our achievements as well as an indication of current activities.

2 Test Program

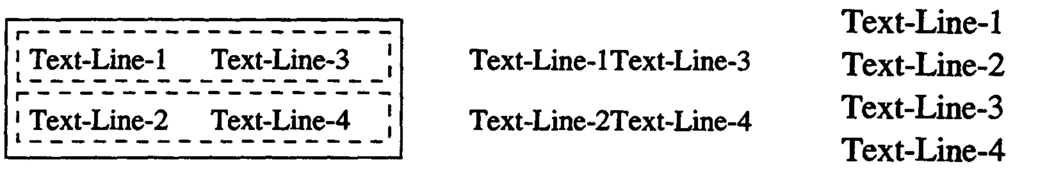
Each year, ISRI conducts an in-depth test of the accuracy of existing OCR technologies. The test is both free and voluntary, and most commercial OCR vendors, and several prototype developers, have participated. Each year, we attempt to test more devices using more data and more measures of performance [12–14, 11]. We believe our tests have contributed greatly to the general understanding of the state-of-the-art.

As part of this program, we have concentrated on developing new measures of OCR performance and on preparing new test data. We are modifying our testing environment to work with wide characters (i.e., using Unicode) and are now beginning to test foreign language OCR systems. We also conduct studies of OCR problems via careful examination of the errors detected by our tests.

2.1 Metric Development

One of the most important abilities of an OCR system is the ability to automatically zone pages. By zoning, we mean finding the text blocks on a page and determining the correct reading order. Although all page-reading OCR systems accept zone coordinates for each text block on a page, manual determination of such coordinates is expensive and time-consuming. For this reason, most systems will attempt to provide proper zone coordinates for each page automatically.

This ability can be especially valuable when some pages to be processed contain single column text and some contain multiple column (newspaper style) text. For multi-column text, it is important to properly decolumnize text paragraphs in the output (see Figure 1).



(a) Incorrectly Zoned Page (b) Generated Text (c) Correct Text

Figure 1: Zoning Error

For scientific documents that also contain much tabular information, it is important that tables retain their structure. Thus, a metric that measures a system's ability to properly decolumnize multi-column text while not decolumnizing tabular information is needed.

To measure how well different devices perform this task, we developed a metric we call the "Cost of Automatic Zoning" [5]. Appendix A shows a series of graphs taken from our 1995 test that illustrate very clearly which devices most accurately recognize multi-column text and tabular information at the same time [11].

We have also developed measures called "non-stopword accuracy," "marked character efficiency," "accuracy by character class," and others. It is our general opinion that many more image analysis metrics are needed.

2.2 Ground-Truth Data Preparation

In order to conduct a test program that expands each year, we have undertaken the task of preparing test data. We have established a laboratory to select, scan, zone, enter, and archive test data. Page 1 of Appendix B shows the steps we employ in preparing "Truth" data for testing. Page 2 shows the current status of our English language databases and page 3 the status of several foreign language databases.

We are also preparing test data for other research centers. Beginning this Fall, in a cooperative project with the DFKI in Kaiserslautern, we are preparing a database of German- language documents.

2.3 Foreign Language Testing

As part of a project for the DoD, we have created a version of our experimental environment that is capable of testing OCR devices in any foreign language. Appendix C is a page from our 1994 annual report that shows the image of a paragraph of Chinese text, the Ground- Truth Chinese characters, the OCR-generated characters for this paragraph, and a listing showing the position of each OCR error and the character substitution made [7].

We are currently extending our set of software tools to support Unicode and we are learning to prepare Unicode Ground-Truth files in our data preparation laboratory.

In addition to a test of Chinese devices last year and a test of Spanish language devices this year, we are working on a test of Japanese and German for next year.

2.4 Problem Characterization

We think that one of the most important contributions we can make is in problem characterization via careful examination of the errors detected by our tests. Page 1 of Appendix D shows a tabulation of the causes (determined by human judgment) of OCR errors in a test performed in 1991 [9].

In this test, we studied the 3,596 character errors produced by an ISRI-designed "voting" algorithm on a test sample of 132 page images. The algorithm produced about 50% fewer errors than the best device that participated in the vote. It was essentially used as a filter to separate the hardest images to recognize. Although the results are based on human judgments, the proportion of broken and touching characters on the images of these errors was striking.

In a cooperative project with George Nagy of RPI, we are currently studying a new collection of "difficult images" extracted from the data used in our 1995 tests. In these studies, we are attempting to provide a taxonomy of image problems and to understand the kinds of information that humans use to solve difficult recognition problems. Page 2 of Appendix D shows examples of image snippets taken from our 1995 annual report [11].

3 Document Image Analysis

A case study showed that rekeying documents is usually more cost effective than editing OCR output unless a minimum OCR accuracy of 95 to 98% is achieved [3]. Moreover, 67% of the total conversion cost is spent on correcting OCR errors. To reduce conversion costs by improving OCR accuracy, we are currently focusing on the following problems:

- Determination of the quality of document images that affects OCR errors.
- Adaptive document image processing techniques for improving the quality of document images.
- Characterization of (image) preprocessing techniques.

Since a large number of document images are archived in an information retrieval system, we are also investigating methods to reduce the size of document images without affecting their readability and OCR systems' ability to recognize them. We are mainly focusing on the trade-off between sampling and quantization in digitizing document images.

3.1 Page Quality Measure

Image degradation, such as touching and broken characters, seems to be the most important source of OCR problems [9, 2]. The studies suggest that the performance of an OCR system could be predicted by measuring page image quality.

A performance prediction module will be a key component in an efficient document conversion system as shown in Figure 2. It will be used to assign incoming documents into one of the following three classes: *good quality* documents for OCR, *mediocre quality* documents for image restoration, and *poor quality* documents for manual rekeying. (An adaptive image restoration technique will be described in the next section.

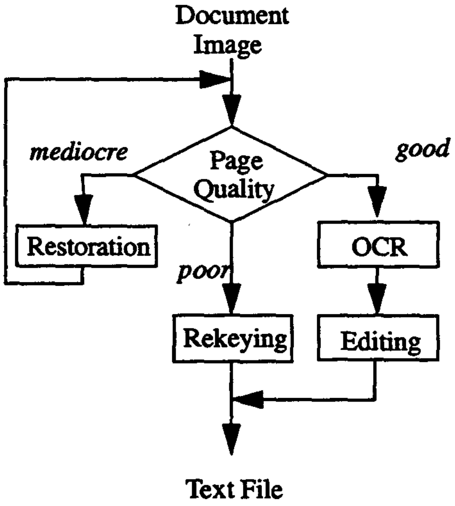


Figure 2: Document conversion with a page quality module

In our feasibility study, the following assumptions were made to limit the scope of the research.

- Page are printed in black and white (no color).
- Page Images have been segmented, and text regions have been correctly identified. The image-based prediction system extracts information from text regions only.

A small set of sample pages with low character accuracy was carefully inspected, and the following observations were made.

- **Observation 1:** Pages with characters whose strokes are thick tend to have many touching characters. Another by-product of these characters is that the loops in letters like a and e often get filled up completely or present only a minimal white portion in the center as shown in Figure 3. Thus, a measure that could capture the existence of these "minimally open" loops would detect problems related to touching characters.

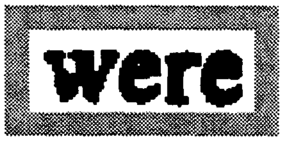


Figure 3: Distorted and Restored Images

- **Observation 2:** Broken characters are usually fragmented into small pieces, and character fragments could have almost any shape as shown in Figure 4. A measure that could weight the existence of these character fragments would detect problems related to broken characters.

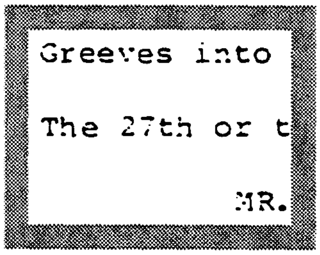


Figure 4: Broken characters

- **Observation 3:** Pages with "inverse video" (white letters on black background) or artistic typesetting tend to produce more OCR errors.

Based on these observations, three simple heuristic rules were developed to classify a page image as either *good* or *poor*. We did not attempt to identify mediocre pages in this study. The training data utilized included 12 clean pages and 12 degraded pages. These pages were extracted from the DOE data sets. For 21 pages, the median accuracy was computed from the output of eight OCR systems [12]. For the remaining 3 pages, median accuracy was computed from the output of six newer systems [14]. Since the highest character accuracy obtained from the degraded pages was 88.76%, *good* corresponds to an expected accuracy above 90%. On the other hand, *poor* corresponds to an expected accuracy below 90%.

To detect minimally open loops (Observation 1), a *White Speckle Factor* (WSF) was defined. A white speckle is any 8-connected white component whose size is less than or equal to 3 pixels in height and width. It is defined as:

$$WSF = \frac{\#whiteboundingrectangles < 3by3}{\#whiteboundingrectangles}$$

This metric weighs the amount of white-speckle present. It is expected that image quality goes down as this ratio goes up. Because of the small sample training data, the threshold value for identifying poor quality pages was manually determined to be 0.1. Thus, the first rule is as follows:

Rule 1: if $WSF \geq 0.1$ then *poor*

To measure the amount of broken characters in a given page (Observation 2), a *Broken Character Factor* was defined. In general, the sizes and shapes of character fragments vary widely. Thus, their bounding rectangles will have many different widths and heights. When a frequency distribution of the bounding rectangles is plotted as a 3-D histogram, such as Figure 5 the bounding rectangles of character fragments appear near the origin. Thus, this region was defined as the *broken character zone* as shown in Figure 6.

It is important to note that the broken character zone is designed to collect all small black connected components. These small components are mostly charter fragments but can also be "dots", such as the dot of "i" and a period, and other small legal characters in small type sizes. Since these dots will fall inside of the broken character zone, a density-based measurement is sensitive to the distribution of characters in the page. Therefore, the *Broken Character Factor* (BCF) uses area coverage rather than density to make a decision.

To measure the degree of covering of this zone, it is divided into square cells, at a rate of one per square pixel. The bounding rectangles of black connected components are allocated to these cells according to their width and height. After these cells are filled by the connected components, the *Broken Character Factor* is computed as:

$$BCF = \frac{\#ofcellsoccupied}{\#ofcells}$$

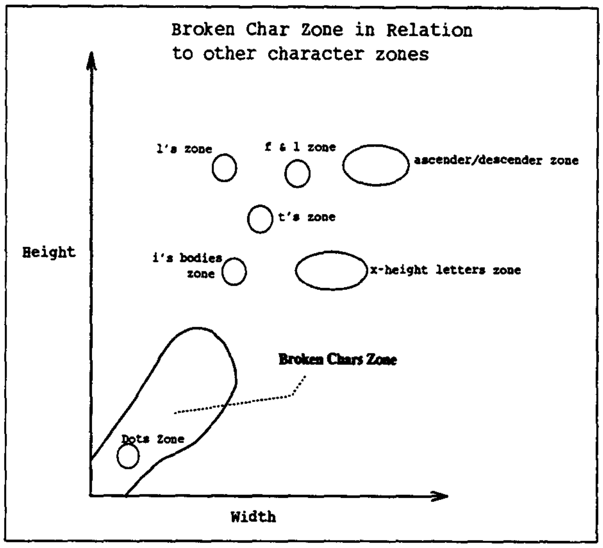


Figure 5: Broken character zone and other character zones

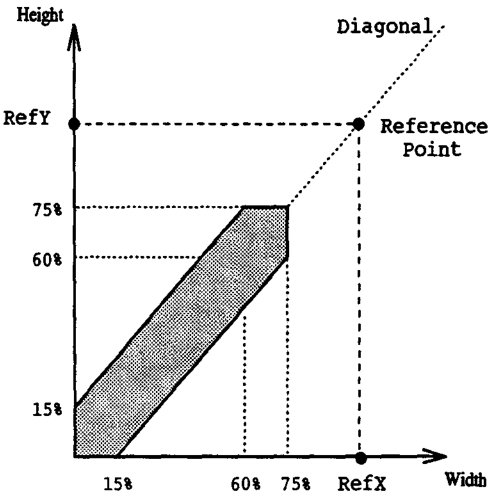


Figure 6: Definition of Broken Character Zone

To eliminate the effects of font sizes, the average height and the average width of bounding rectangles are used as a reference point, and the shape of the *broken character zone* is defined as shown in Figure 6. From observations on the training data, an area coverage of 70% or more is a very strong indicator of the presence of many broken characters in the page. Thus, the second rule is defined as

Rule 2: if $BCF > 0.7$ then *poor*

The third rule that uses the number of white connected components and their sizes as features was also defined to detect inverse video regions (Observation 3). In an inverse video region, white connected components correspond to characters. If either the average height or the average width of white connected components exceeds 30 pixels (approximately 7 points), they are highly likely to be characters. Moreover, the number of black connected components in the region should be small because of the background. Thus, the ratio of the number of black connected components to the number of white connected components also provides useful information. The third rule is defined as:

Rule 3: if max(Avg. White Width,
Avg. White Height) > 30 pixels
and
$\frac{\#blackconnectedcomponents}{\#whiteconnectedcomponents} < 1.5$
then *poor*

This prediction algorithm classifies a given page as poor if at least one of these three rules is activated.

Two sets of test data were utilized. The first set is a subset of Sample-2 data base [14]. It consists of 460 pages that were selected at random from a collection of approximately 2,500 scientific and technical documents (approximately 100,000 pages). Each page was digitized at 300 dpi using a Fujitsu M3096M+ scanner to produce a binary image. Since 21 pages in this data set were used to train the image-based prediction system, the remaining 439 pages were used to test it.

The second set consists of 200 pages selected from 100 magazines that had the largest paid circulation in the U.S. in 1992 as reported by *Advertising Age* magazine [1]. For each magazine, two pages were selected at random. Each page was digitized at 300 dpi using a Fujitsu M3096G scanner. The binary images were generated using a fixed threshold of 127 out of 255 by this gray scale scanner.

The performance of this prediction system was measured using the median character accuracy of six systems obtained in the *Third Annual Test of OCR Accuracy* [14].

Each character insertion, deletion, or substitution needed to correct the OCR generated text was counted as an error. Each reject character was also counted as a substitution error in this calculation. Character accuracy is defined as:

$$Characteraccuracy = \frac{n - \#errors}{n}$$

where n is the total number of characters in the ground truth data.

Table 1 shows the confusion matrix for Sample-2 data set. The classifier misclassified 15 *poor* quality pages as *good* and 53 *good* quality pages as *poor*. Thus, its error rate was 15%.

The 15 "*poor* → *good*" misclassifications were carefully examined. The page images did not present substantial degradation. All but four of these problematic pages contained numerical tables in them. The OCR output for many of these tables were illegible and generally useless. Yet, from an image defects point of view, it could find no evidence justifying a poor label.

Table 2 shows the confusion matrix generated from 200 pages obtained from magazines. The error rate was 13.5%, but the system did not make any "*poor* → *good*" misclassifications. These results suggest that it is feasible to develop a system for predicting the character accuracy of a given page by a given OCR system.

True ID	Recognized	
	Good	Poor
Good	159	53
Poor	15	22

Table 1: Confusion matrix for Sample 2 (439 pages)

True ID	Recognized	
	Good	Poor
Good	159	27
Poor	0	14

Table 2: Confusion matrix for Magazines (200 pages)

We are currently improving the classifier by experimenting with new features and non-parametric pattern recognition techniques.

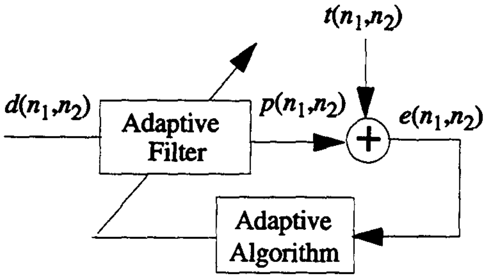


Figure 7: Adaptive system used to model inverse distortions

3.2 Adaptive Document Image Restoration

Although the literature has identified and classified many of the common distortions that cause OCR errors, mathematical models of these distortions have not been well developed. If an OCR system cold identify the distortions degrading the text image and the corresponding distortion model is available, an inverse distortion filter could be designed to restore the text image. The OCR system should then be able to recognize the restored text images more accurately. Because of the myriad of possible inverse distortion models that would be required to implement such a system, we proposed to restore distorted character images causing OCR errors by adaptively generating an inverse distortion model [16].

Figure 7 shows a closed loop adaptive system that can be used to restore distorted images. The image, $d(n_1, n_2)$, corresponds to the distorted characters that the OCR system can correctly recognize, and the image, $p(n_1, n_2)$, is the output of the adaptive restoration filter. The restoration filter implements a two dimensional transformation that approximates the inverse model of the distortion. The image, $t(n_1, n_2)$, called the training image, is an ideal undistorted version of the image, $d(n_1, n_2)$. In general, the ideal undistorted character images are not available. However, if the OCR system can output character labels, confidence values and font features, the ideal undistorted images of correctly recognized characters can be synthesized as shown in Figure 8. It is also possible to adaptively generate the undistorted characters from characters in the image using the algebraic operations.

To train the restoration filter, the error signal, $e(n_1, n_2)$, which is the difference between the training signal, $t(n_1, n_2)$, and the output of the adaptive filter, $p(n_1, n_2)$, is generated. The adaptive algorithm uses a measure of this error signal to adjust the structure of the restoration filter, so that it approximates the inverse of the distortion in some optimal sense. When the adaptive algorithm has sufficiently minimized the measure of the error signal, the processed image, $p(n_1, n_2)$, should approximate the training image, $t(n_1, n_2)$. If the distortion affecting the unrecognized characters is similar to the distortion affecting the characters in the image, $d(n_1, n_2)$, then the adaptive filter should be able to restore the unrecognized distorted character images. Thus, the OCR accuracy of the page should improve.

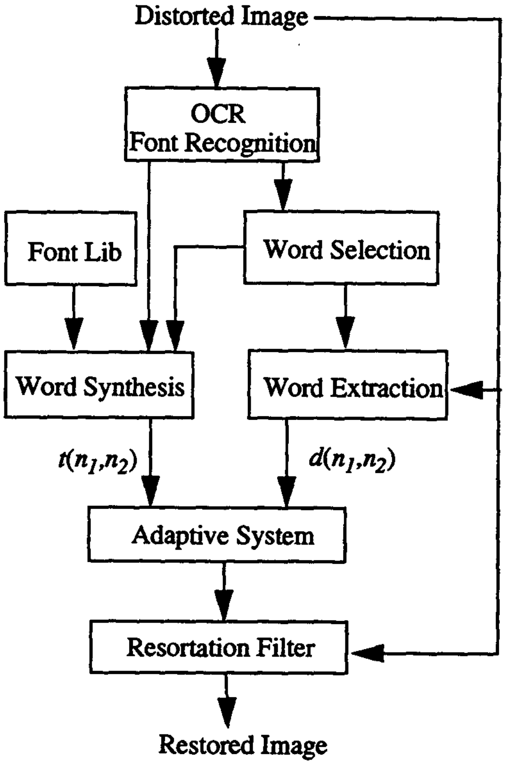


Figure 8: Practical adaptive document image restoration system

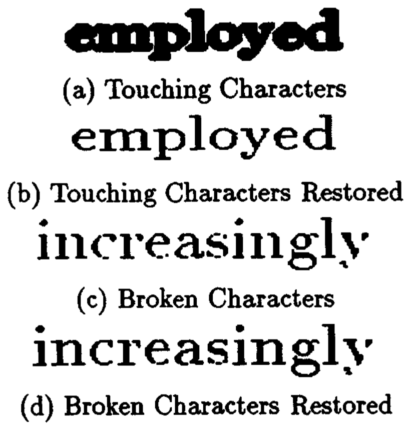
To approximate the inverse distortion model effectively, a variety of architectures is examined. The adaptive linear combiner and the adaptive recursive filter were determined to be inadequate for this application. On the other hand, a feedforward neural network with two layers of neurons was able to model distortion that caused either touching characters or broken characters.

Several distorted documents were restored by this method. Examples of distorted images and restored images shown in Figures 9. Experimental results suggest that this technique can improve OCR accuracies of 70-80% up to 95-98%.

3.3 Characterization of Image Processing Algorithms

The objectives of this research are to develop methods for characterizing image processing algorithms used by document image analysis systems and appropriate test data sets. We are currently focusing on binarization algorithms and skew estimation algorithms.

The goal directed approach proposed by Trier and



(a) Touching Characters

(b) Touching Characters Restored

(c) Broken Characters

(d) Broken Characters Restored

Figure 9: Filled loops

Jain [24] is used to study binarization algorithms. The performance of an algorithm is evaluated based on character accuracies obtained from resulting binary images.

In a study, four contemporary OCR systems and 50 monochromatic hard copy pages containing 91,649 characters were used to study histogram-based binarization algorithms [4]. These pages were digitized at 300 dpi and 8 bits/pixel, and 36 different threshold values (ranging 59 to 199 in increments of 4) were used. The resulting 1,800 binary images were processed by all four OCR systems. all systems made approximately 40% more errors from images generated by Otsu's method [10] than those of the ideal histogram method. Two of the systems made approximately the same number of errors from images generated by the best fixed threshold value and Otsu's method.

The performance of a skew detection algorithms is determined by comparing its output with the corresponding manually prepared ground truth data. We are investigating conditions (features) that affect the performance of several skew detection algorithms.

3.4 Trade-Off Between Sampling and Quantization

Lee et. al. showed that the theoretical trade-off between sampling and quantization in signal processing [8]. They also suggested the goal-directed analysis of the trade-off. We are following their suggestion to find the optimal combination of sampling and quantization for document image analysis systems.

In addition to reducing the size of data, storing documents as gray-scale images avoids the following problems. Currently, most documents are stored as 200-600 dpi binary images. To display such images on a high resolution monitor, they must be reduced to 75-100 dpi, and reduced documents are usually difficult read. Moreover, photos and color information are usually lost during the scanning process. In other words, gray-scale images provide better viewing experience. .

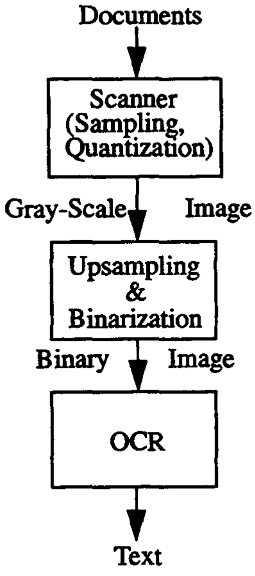


Figure 10: Practical adaptive document image restoration system

The remaining question is that effects of low resolution gray-scale images on OCR accuracy. Since contemporary OCR systems process 300-400 dpi binary images, we are investigating the performance of OCR system on binary images generated from gray-scale images using a combination of upsampling and binarization techniques shown in Figure 10 [6].

A page image containing 1090 characters were processed by the method. The results shown in Table 3 suggest that we can generate a high quality binary image for OCR from a 100 dpi gray-scale image

We are currently investigating the optimal combination of sampling and quantization to reduce images sizes with the minimum loss in character accuracy.

DPI	75	100	150	300
Samples	8	8	8	1
Accuracy	84.77%	98.99%	99.82%	99.91%

Table 3: Character Accuracy

4 Post Processing/Information Retrieval

Until quite recently, the effect of OCR errors on retrieval was unknown. Most projects for the conversion of hard copy documents to electronic form, either included a tedious manual correction step or left the errors and used the raw OCR text for retrieval purposes. To demonstrate what effect these errors might have, we, at ISRI, have conducted several experiments [22, 20, 21, 18] with a collection of LSS documents that showed average ***precision and recall*** were not affected by OCR errors. Precision and recall are standard measures used to determine the effectiveness of a retrieval system. Comparisons were made between the 99.8% correct versions of documents and the raw OCR text of the same set. To help support our claims, we ran an analogous experiment in conjunction with the University of Massachusetts at Amherst [18]. Unfortunately, ***averages*** don't always reveal what may occur for individual queries given to a system; our experimentation also concluded that if documents are highly degraded by the OCR process, they may not be returned by the IR system given any query.

Observations made during our experimentation led to the design of a prototype post-processing system (PPS) that improves the usability of OCR text by correcting OCR errors. After running the raw OCR text through the PPS, we found that precision and recall improved and additionally, we were able to increase the retrievability of some of the highly degraded documents.

The knowledge we have gained from the experiments we have conducted on the interaction between OCR and IR has led us to design the following systems:

1. **PPS**, improve the quality of the text output by the OCR device.

2. **Autotag**, to automatically tag logical entities within the documents' text.

3. **IR system**, to exploit the information provided by the previous systems and in turn, increase the effectiveness of retrieval of OCR-generated text.

Each of these systems are at various stages; the PPS is the most complete, a prototype of Autotag has been built but requires more research, evaluation, and design implementation before its completion is realized, and the IR system is only in the design stages. In the following subsections, we briefly describe some of the ideas we have implemented.

4.1 PPS and Autotag

The Post-Processing System is built on the philosophy that for every misspelling in the document, there are other occurrences of the same word correctly recognized by the OCR device. These correctly recognized occurrences can be used to correct the misrecognized forms. The PPS divides the words of a document into two lists: the misspellings list and the correctly spelled words list. Two routines in the PPS system, ***Phrasing*** and ***Clustering***, attempt to correct words in the misspelling list.

misspelling	correction	routine
facilities	facilities	clustering
Subiect	Subject	clustering
sguare	square	clustering
C∼wernment	Government	phrasing
Ceolo∼ic	Geologic	phrasing

Table 4: Corrections made by the post-processing system.

misspelling	correction
Di ametral	Diametral
Col umbus	Columbus
Materi al	Material
Washi ngton	Washington
Davi s	Davis
Pi tman	Pitman

Table 5: Misspellings corrected by the post-processing system.

Phrasing: For each misspelling, this routine tries to identify a phrase that might contain a correct form of the the misspelling. If such a phrase is found, the correction is made.

Clustering: For each misspelling, the system uses approximation matching to identify words from the correctly recognized list that may be corrections for the misspelling. If a single word can be selected that is sufficiently close to the misspelling, then the replacement is made.

Table 4 contains a list of misspellings and their corrections made by these two routines. From this table, we observe that the OCR device confused `i` with `j` in the second row of the table. As the PPS makes corrections, it also keeps track of the nature of these errors. This information can be used for learning routines that adapt themselves to the nature of the errors in a particular document.

Table 5 shows another list of misspellings corrected by PPS. The learning routine will observe that the OCR device breaks the word after the character `i`. This information can then be used to make another attempt at correcting other misspellings with the same characteristic.

In many of the PPS routines, we are unable to determine the correct replacement for the misspelling. In this case, we keep "candidate" corrections or ***possibilities*** for the misspelling. These can be used by the retrieval component of our information system to improve document recall.

Although an OCR system may produce misspellings, it can also provide useful information that can enhance the text it generates. This concept is the basis for our Autotag system. Autotag employs meta data provided by an OCR device, such as spacing, word location, typeface and other font information, to determine a document's logical structure. For example, Autotag can locate the author and title of a document if it exists. Other logical structures (like sentences, paragraphs and sections) that may be meaningful at retrieval time are also tagged.

The readers are referred to [17], [19] and [23] for more information on PPS and Autotag.

4.2 Retrieval System

The retrieval component of our information system will take advantage of both a Boolean and vector-based query language interface. With OCR text comes some uncertainty about the occurrences of terms since there may be misspellings. To incorporate this uncertainty into our system, we plan to give weights to "possibilities", where possibilities are candidate corrections for the misspellings generated by routines in PPS. For example, the misspelling "rnountain" may have "fountain" and "mountain" as possibilities. Based on the information available in the learning routine of PPS, we calculate the probabilities $P(mountain \mid rnountain)$ and $P(fountain \mid rnountain)$. These probabilities represent the certainty of the misspelling being either mountain or fountain.

The standard Boolean operators such as and will be extended to reflect these probabilities. For instance, a search for documents containing terms A and B, will identify documents which have the correct forms of both A and B or documents containing A and B as possibilities for some misspellings. The combined probability of A and B will determine the relevance of a returned document to a query.

This notion of probability is naturally extended to a ranked retrieval system with vector-based query languages based on the vector space or Bayesian retrieval models[15].

The retrieval engine will use a *mapping* generated by Autotag to relate the ASCII text and corresponding images. This mapping will provide detailed information on the geometry of words, sentences, paragraphs, titles, authors, and other document entities. By utilizing this geometry information, the system will display images with searched entities highlighted on the image. Hence, a user will view the actual image with all its artwork and graphics and still have all the capabilities which come with searchable text. Figure 11 shows the display of our planned IR system.

References

[1] Ad age 300. *Advertising Age*, June 14 1993.

[2] Mindy Bokser. Omnidocument technologies. *Proceedings of the IEEE*, 80(7):1066–1078, 1992.

[3] L. A. Dickey. Operational factors in the creation of large full-text databases. In *DOE INFOTECH Conference*, Oak Ridge, TN, May 1991.

[4] J. Kanai and K. Grover. A preliminary evaluation of histogram-based binarization. In *Document Recognition II, SPIE Proceedings Series*, volume 2422, pages 205–213, 1995.

[5] J. Kanai, S. V. Rice, T. A. Nartker, and G. Nagy. Automated evaluation of ocr zoning. *IEEE Tranactions on Pattern Analysis and Machine Intelligence*, 17(1), January 1995.

[6] J. Kanai and P. Stubberud. Upsampling document images. In *Proceedings of COMCON5*, June 1995.

[7] Junichi Kanai, Yucheng Liu, Stephen V. Rice, and Thomas A. Nartker. A preliminary evaluation of chinese OCR systems. Technical Report 94-04, Information Science Research Institute, University of Nevada, Las Vegas, April 1994.

[8] D. Lee, T. Pavlidis, and G. W. Wasilkowski. A note on the trade-off between sampling and quantization in signal processing. *Journal of Complexity*, 3:359–371, 1987.

[9] T. A. Nartker, R. B. Bradford, and B. A. Cerny. A preliminary report on UNLV/GT1: A database for ground-truth testing in document analysis and character recognition. In *Proceedings of the First Symposium on Document Analysis and Information Retrieval*, pages 300–315, Las Vegas, NV, March 1992.

[10] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on SMC*, SMC-9(1):62–66, 1979.

[11] Stephen V. Rice, Frank R. Jenkins, and Thomas A. Nartker. The fourth annual test of OCR accuracy. Technical Report 95-04, Information Science Research Institute, University of Nevada, Las Vegas, April 1995.

[12] Stephen V. Rice, Junichi Kanai, and Thomas A. Nartker. A difference algorithm for OCR-generated text. In *Proceedings of the IAPR Workshop on Structural and Syntactic Pattern Recognition*, Bern, Switzerland, August 1992.

[13] Stephen V. Rice, Junichi Kanai, and Thomas A. Nartker. An evaluation of OCR accuracy. Technical Report 93-01, Information Science Research Institute, University of Nevada, Las Vegas, April 1993.

[14] Stephen V. Rice, Junichi Kanai, and Thomas A. Nartker. The third annual test of OCR accuracy. Technical Report 94-03, Information Science Research Institute, University of Nevada, Las Vegas, April 1994.

[15] G. Salton. *The SMART Retrieval System, Experiments in Automatic Document Processing.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1971.

[16] P. Stubberud, J. Kanai, and V. Kalluri. Adaptive restoration of text images containing touching and broken characters. Technical Report 95-05, Information Science Research Institute, University of Nevada, Las Vegas, April 1995.

[17] Kazem Taghva, Julie Borsack, Bryan Bullard, and Allen Condit. Global correction of ocr generated errors using an inverted file based text retrieval system. In *Symposium on Advanced Information Processing and Analysis*, Tysons corner, Virginia, 1993. Advanced Information Processing and Analysis Steering Group.

[18] Kazem Taghva, Julie Borsack, and Allen Condit. Evaluation of model-based retrieval effectiveness with OCR text. *ACM Transactions on Information Systems.* (to appear).

[19] Kazem Taghva, Julie Borsack, and Allen Condit. An expert system for automatically correcting OCR output. In *Proceedings of the IS&T/SPIE 1994 International Symposium on Electronic Imaging Science and Technology*, pages 270–278, San Jose, CA, February 1994.

[20] Kazem Taghva, Julie Borsack, and Allen Condit. Results of applying probabilistic IR to OCR text. In *Proceedings of the Seventeenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 202–211, Dublin, Ireland, July 1994.

[21] Kazem Taghva, Julie Borsack, and Allen Condit. Effects of OCR errors on ranking and feedback using the vector space model. *Information Processing and Management*, 1995. (to appear).

[22] Kazem Taghva, Julie Borsack, Allen Condit, and Srinivas Erva. The effects of noisy data on text retrieval. *Journal of the American Society for Information Science*, 45(1):50–58, January 1994.

[23] Kazem Taghva, Allen Condit, and Julie Borsack. Autotag: A tool for creating structured document collections from printed materials. Technical Report 94-11, Information Science Research Institute, University of Nevada, Las Vegas, December 1994.

[24] O. D. Trier and A. K. Jain. Goal-directed evaluation of binarization methods. In *Proceedings of Performance versus Methodology in Computer Vision*, pages 206–217, Seattle, WA, 1994.

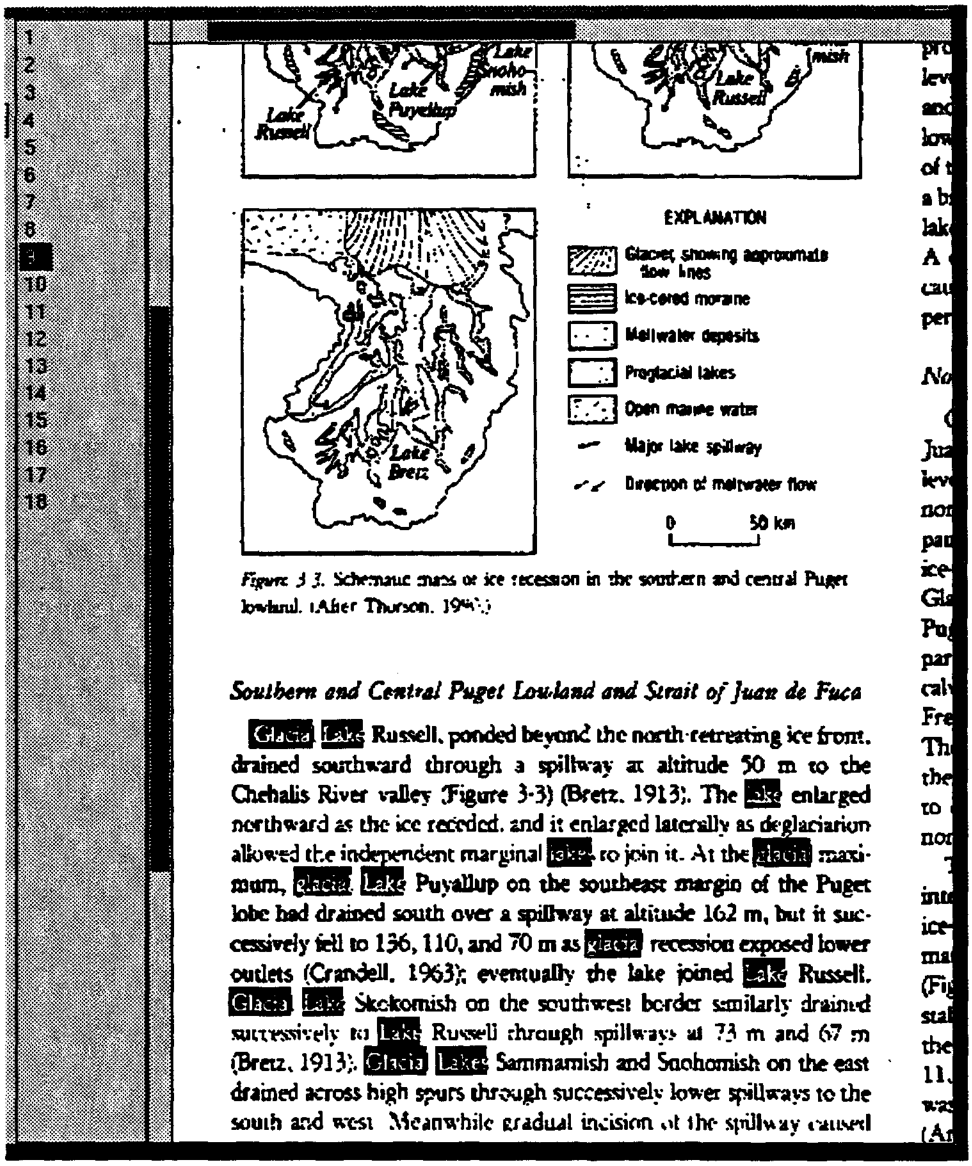


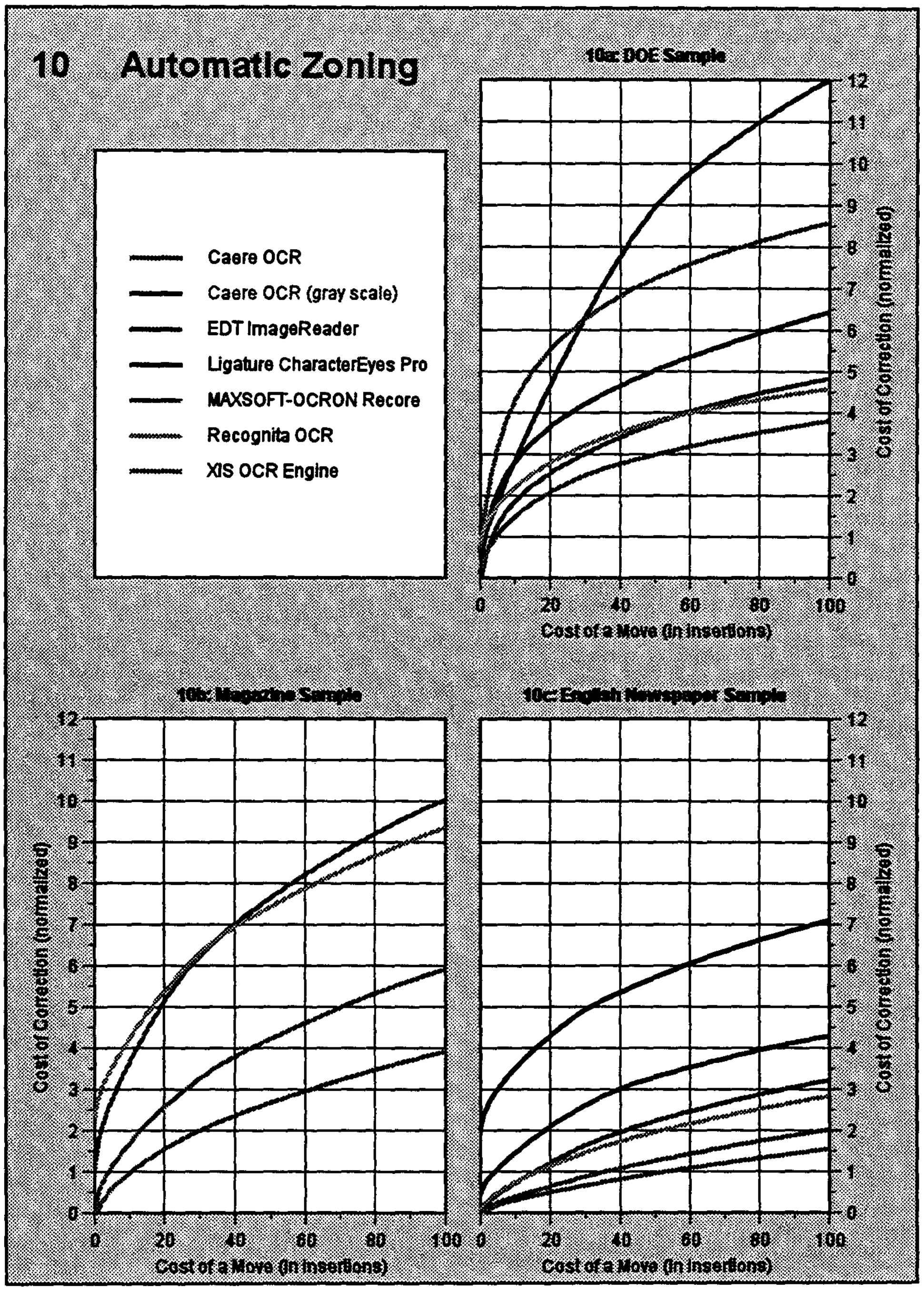
Figure 3 3. Schematic maps of ice recession in the southern and central Puget lowland. (After Thorson, 198·)

Southern and Central Puget Lowland and Strait of Juan de Fuca

Glacial Lake Russell, ponded beyond the north-retreating ice front. drained southward through a spillway at altitude 50 m to the Chehalis River valley (Figure 3-3) (Bretz. 1913). The lake enlarged northward as the ice receded. and it enlarged laterally as deglaciation allowed the independent marginal lakes to join it. At the glacial maximum, glacial Lake Puyallup on the southeast margin of the Puget lobe had drained south over a spillway at altitude 162 m, but it successively fell to 136, 110, and 70 m as glacial recession exposed lower outlets (Crandell. 1963); eventually the lake joined Lake Russell. Glacial Lake Skokomish on the southwest border similarly drained successively to Lake Russell through spillways at 73 m and 67 m (Bretz, 1913). Glacial Lakes Sammamish and Snohomish on the east drained across high spurs through successively lower spillways to the south and west. Meanwhile gradual incision of the spillway caused

Figure 11: IR system's user interface displaying original document image

Appendix A: Cost of Automatic Zoning



Appendix B: Preparation of Ground-Truth Data

1. Acquire document sample
   Choose document class
   Identify source for documents
   Choose selection strategy and method of acquisition
   Obtain documents

2. Select page sample
   Choose page (or partial page) sampling strategy
   Sample and prepare pages
   Choose zone types

3. Train data entry personnel
   Select & acquire tools
   Train data entry staff

4. Scan pages Determine scanning variables (i.e. threshold & page placement)
   Scan all pages (usually at 200, 300, 400, binary & 300 greyscale)
   Verify images
   Quality control

5. Archive document & page collections

6. Manually zone images

7. Prepare Truth text
   Prepare multiple manual entries
   Prepare text from ISRI voting algorithm
   Resolve multiple manual entry & voting differences

8. Archive images,
   zone information,
   all manual & voting truth input, &
   ground-truth text

ISRI English Ground-Truth Databases

(As of 6/30/95)

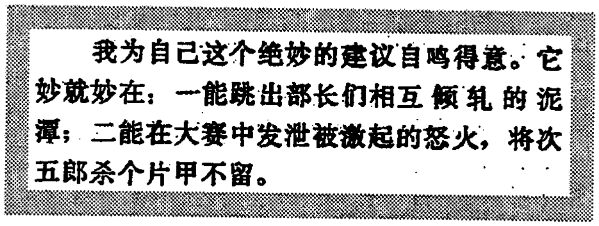
	Document Sample			Page Sample			T.S.	Scan Pages			Arch.	Zone Images		Prepare Truth		Arch .
Name	Defined	# Docs.	% Com	Defined	# Pages	% Com	Com.	# Imgs	Resol.	% Com		# Zones	% Com	# Character	% Com	
DOE Sample1	Yes	250	100	Yes	240	100	Yes	132	3b	100	Yes	242	100	278,786	100	Yes
DOE-Sample1 Synthesized	Same	250	100	Synthe-sized	132	100	Yes	242 (x 9)	Synthe . At 3b	100	Yes	242 (x 9)	100	278,786 (x 9)	100	Yes
DOE-Sample1 SynPrintedScand	Same	250	100	Syn-Printed	132 (x 9)	100	Yes	242 (x 9)	3b	100	Yes	242 (x 9)	100	278,786 (x 9)	100	Yes
DOE Sample2	Yes	2500	100	Yes	460	100	Yes	460	3b	100	Yes	1313	100	817,946	100	Yes
DOE Sample3	Yes	2500	100	Yes	800	100	Yes	800	2,3,4b 3gs	100	Yes	2431	100	1,463,51 2	100	Yes
Magazine Sample1	Yes	100	100	Yes	200	100	Yes	200	2,3,4b 3gs	100	Yes	1414	100	666,134	100	Yes
US-Newspaper Sample1	Yes	50	100	Yes	200	100	Yes	200	2,3,4b 3gs	100	Yes	758	100	492,080	100	Yes
BusinessLetter Sample1	Pages	200	100	Yes	200	100	Yes	200	2,3,4b 3gs	100	Yes	1508	100	319,756	100	Yes
DOJ-Microfilm Sample1	Film Images	2000	100	Yes	200	100	Yes	200	2b	100	Yes	704	100	471,755	100	Yes
FAX1 BusinessLetters	Same	200	100	Yes	200	100	Yes	200	Fine & Stand.	100	Yes	1357	100	319,756	100	Yes
Industry Annual Reports	Yes	75	100	Yes	300	100	Yes	300	2,3,4b 3gs	100	No	1703	100		30	No
Magazine Sample2	Yes	75	100	Yes	60	20	Yes	60	2,3,4 b 3gs	20	No	163	20		10	No
96 Mystery Sample	Yes															
Commerce Business Daily	Yes															
Supreme Court Slips	Yes	10	100	Yes	100	100	Yes	100	2,3,4 b 3gs	100	No	219	100	186710	33	No

ISRI Foreign Language Ground-Truth Databases

(As of 6/30/95)

	Document Sample			Page Sample			T.S.	Scan Pages			Arch.	Zone Images		Prepare Truth		Arc h.
Name	Defined	# Docs.	% Com	Defined	# Pages	% Com	Com.	# Imgs	Resol.	% Com		# Zones	% Com	# Character	% Com	
Chinese-test Sample	Yes	N,M, B,GA	100	Yes	57	100	No	57	3b 3gs	100	...	57	100	48,378	100	
Japanese-test Sample	Yes	27	100	Yes	73	100	No	4	2,3,4b 3gs	5	...	29	...	4,400	...	
Span-Newspaper Sample1	Yes	36	100	Yes	144	100	Yes	144	2,3,4b 3gs	100	Yes	569	100	348,091	100	Yes
China-Newspaper Sample1	Yes	19	100	Yes	95	100	Yes	95	3b 3gs	100	No	614	100	81,080	100	
Japan-Newspaper Sample1.Verticle	Yes	30	100	Yes	60	100	Yes			0	No				30	
Japan-Newspaper Sample1.Horizon	Yes (same)	30	100	Yes	57	100	Yes			0	No				0	
Cyrillic-Newspaper Sample1	No	...	...	No	...	...	...									
Farsi-Newspaper Sample1	Yes	45	100	No	...	...	...									
GermanBusLetter Sample1	No	140	75	No	150	75	...									

Appendix C: Chinese OCR Errors Identified



我为自己这个绝妙的建议自鸣得意。它
妙就妙在：一能跳出部长们相互倾轧的泥
潭；二能在大赛中发泄被激起的怒火，将次
五郎杀个片甲不留。

Figure 2: Digitized Paragraph

我为自己这个绝妙的建议自鸣得意。它
妙就妙在：一能跳出部长们相互倾轧的泥
潭；二能在大赛中发泄被激起的怒火，将次
五郎杀个片甲不留。

Figure 3: Correct Text

我为自己这个绝妙的建议自鸣碍意。它
妙就妙在：一能跳出部长们相互倾轧的泥
潭：二能在大家中发泄被激起的怒火，将次
五郎杀个片甲不留。

Figure 4: OCR-Generated Text

我为自己这个绝妙的建议自鸣{1}意。它
妙就妙在：一能跳出部长们相互倾轧的泥
潭{2}二能在大{3}中发泄被激起的怒火，将次
五郎杀个片甲不留。

	Correct	Ocr Output
{1}	{得}	{碍}
{2}	{；}	{：}
{3}	{赛}	{家}

Figure 5: OCR Errors Identified

Appendix D: Characterization of OCR/Image Problems

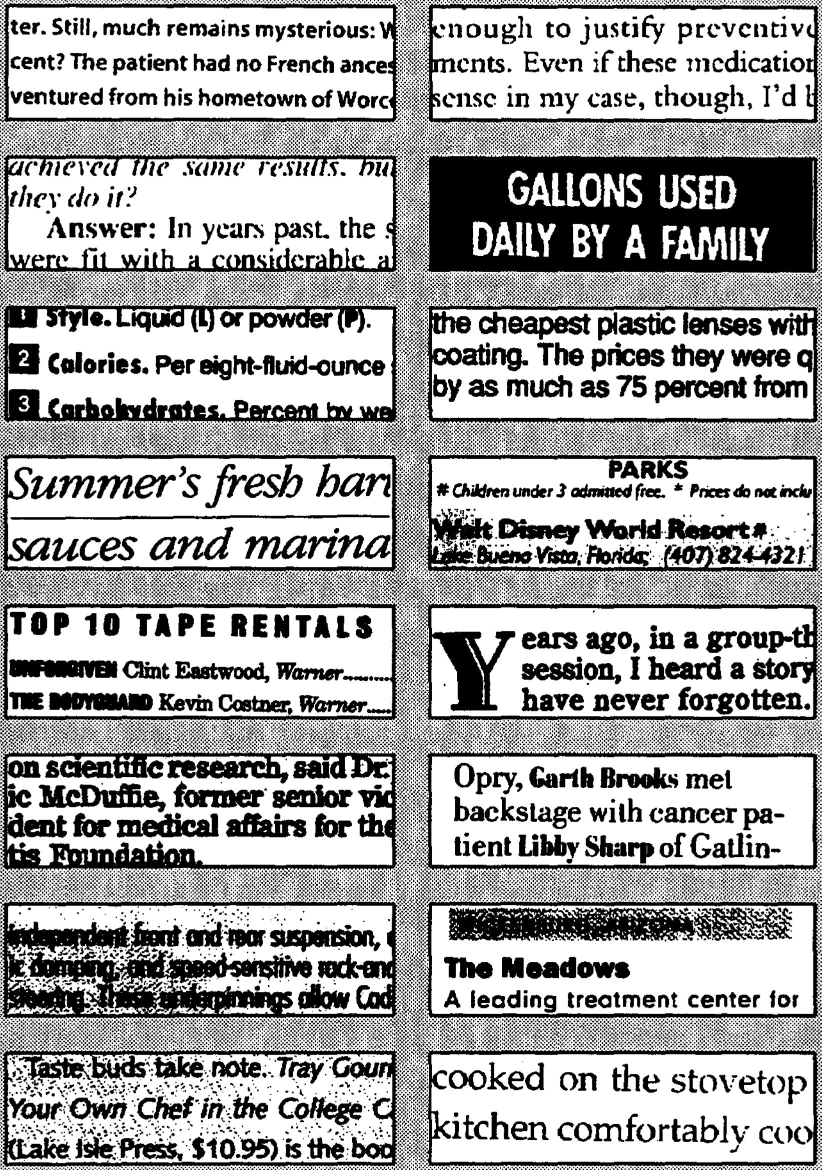
Problem Category	Number of Errors	Fraction of Total Errors
Broken characters	1872	52.1
Touching characters	734	20.4
Noise / speckle	122	3.4
Skew (or curved baseline)	49	1.4
Broken & touching	186	5.2
Broken & noise	9	0.3
Broken & skew	33	0.9
Touching & noise	2	0.1
Touching & skew	10	0.3
Total	3017	83.9
Similar symbols (1,l O,0)	207	5.8
Wrong case	12	0.3
Stylized characters	46	1.3
Introduced spaces	79	2.2
Dropped spaces	39	1.1
Unknown cause	196	5.5
Total	579	16.1
Grand Total	3596	100

Table 1: Distribution of Estimated ASCII-OCR Problems

Problem Category	Number of Occurrences
Subscripts	157
Superscripts	127
Total sub & superscripts	284
Greek letters	69
Degree symbol	27
Bullet	20
Times symbol	17
Long hyphen/dash	13
Other (21 different problems)	64
Total Non-ASCII	210

Table 2: Number of Occurrences of Subscripts and Non-ASCII Characters

Figure 3: Examples from Page Quality Group 5, Magazine Sample



ter. Still, much remains mysterious: W
cent? The patient had no French ances
ventured from his hometown of Worc

enough to justify preventive
ments. Even if these medicatio
sense in my case, though, I'd l

achieved the same results. bu
they do it?
Answer: In years past. the s
were fit with a considerable a

GALLONS USED
DAILY BY A FAMILY

1 **Style.** Liquid (L) or powder (P).
2 **Calories.** Per eight-fluid-ounce
3 **Carbohydrates.** Percent by we

the cheapest plastic lenses with
coating. The prices they were q
by as much as 75 percent from

Summer's fresh har

sauces and marina

PARKS
*# Children under 3 admitted free. * Prices do not inclu*
Walt Disney World Resort #
Lake Buena Vista, Florida; (407) 824-4321

TOP 10 TAPE RENTALS
UNFORGIVEN Clint Eastwood, *Warner*........
THE BODYGUARD Kevin Costner, *Warner*.....

Years ago, in a group-th
session, I heard a story
have never forgotten.

on scientific research, said Dr.
ic McDuffie, former senior vic
dent for medical affairs for the
tis Foundation.

Opry, **Garth Brooks** met
backstage with cancer pa-
tient **Libby Sharp** of Gatlin-

independent front and rear suspension,
ic damping, and speed-sensitive rack-and
steering. These underpinnings allow Cod

[illegible]
The Meadows
A leading treatment center for

Taste buds take note. *Tray Gour*
Your Own Chef in the College C
(Lake Isle Press, $10.95) is the boo

cooked on the stovetop
kitchen comfortably coo

DPI—Declassification Productivity Initiative

Thomas Curtis
Department of Energy
NN-523
19901 Germantown Road
Germantown, MD 20874

Abstract

The Department of Energy (DOE) Office of Declassification (OD) is responsible for executing the Department's technical and policy responsibilities related to the identification of technology and other information that may affect national security. The office carries out a multi-faceted international and domestic program to identify classified and unclassified but sensitive information, technology, and activities which may contribute to the proliferation of nuclear weapons by other countries or subnational groups and applies appropriate limitations as may be prescribed by treaties, laws, Executive orders, DOE orders, and administrative practices. The promulgation of general policy guidance and specific guides for classification and declassification are a major responsibility of OD.

DOE is faced with an huge inventory of classified documents. So far, at least 130 million pages have been identified as historically significant. Motivated by the end of the cold war and a new spirit of openness, the Department seeks to release the unclassified portion of this information to the American public. By the direction of the Congress, OD has embarked on the Declassification Productivity Initiative (DPI) a concerted effort to use technology to improve declassification productivity.

To accomplish this OD has identified goals to increase production volume, reduce backlogs, and speed declassification processing. OD seeks to increase the use of automated tools and systems, improve cooperation with other agencies, and acquire technology from industry and academia to improve productivity. DPI will support the development of standards to provide for inter-operability with other agency declassification automation systems and tools. DPI will increase productivity and accuracy by:

(1) Developing an in depth understanding of declassification by studying, documenting, analyzing, and modeling the cognitive processes, knowledge used in declassification, and the structures of the documents to be declassified; and

(2) Using the knowledge developed above to reengineer processes, design and develop knowledge-based automated tools, and implement procedures and techniques to improve the productivity and accuracy of the declassification process.

Almost all the documents of interest for declassification processing are only available in paper form. These documents may be more than 40 years old; include carbon, Xerox, and facsimile copies; and range from very poor to fair condition. Conversion of these documents to computer usable bit map and then text forms represents a major early challenge for DPI. Further challenges also exist for information which has been stored as microfilm, photographs, and on other media.

Foreign OCR

Open Source Document Analysis Requirements

Jerry Kowalski
FBIS

Arabic Character Recognition

John Trenkle **Andrew Gillies**
Erik Erlandson **Steve Schlosser**
Environmental Research Institute of Michigan
P.O. Box 134001
Ann Arbor, MI 48113-4001
email: schlosser@erim.org

Abstract

The scripted nature of the Arabic written language poses new problems for automatic text recognition systems. We describe three separate approaches to recognizing Arabic text which have been integrated into a single recognition system, and present some recognition results for that system. Characteristics of the algorithm training and testing data are summarized, along with the contents of a published CD-ROM which contains truthed Arabic text imagery from our data collection and truthing effort.

1 Background

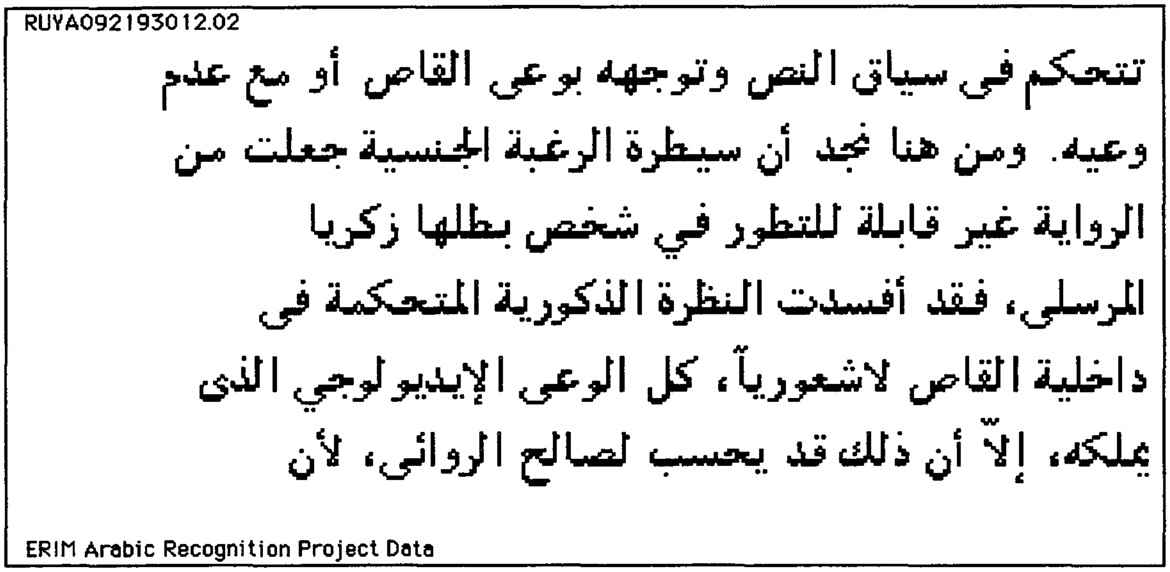
The task of recognizing Arabic text imagery poses difficulties that are not encountered when attempting to recognize text written in a Latin alphabet. Arabic differs from European languages in that it is purely a script language; as with cursive handwriting, there is typically no separation between the characters in a word. The absence of character separation in Arabic text renders it unsuitable for recognition approaches based on character-level segmentation. An example of Arabic text is shown in Figure 1.

Another problematic feature of Arabic script is the frequent presence of ligatures, in which two (or more) adjacent characters are written as a third, entirely different, character. Clearly, character-level segmentation cannot be performed on two characters written as a ligature.

In response to these characteristics of Arabic text, new recognition approaches must be developed which do not rely on the ability to explicitly separate each character from its neighbors in a text image. We have explored three such recognition approaches which do not use character segmentation. These three systems are described in the sections that follow.

2 Approach

Three recognition approaches have been developed in parallel and integrated as subsystems into a single Arabic text recognition system. The first recognition subsystem is based on sliding neural networks, which scan continuously over a word image to search for characters. The second subsystem is based on character



RUYA092193012.02

تتحكم فى سياق النص وتوجهه بوعى القاص أو مع عدم
وعيه. ومن هنا نجد أن سيطرة الرغبة الجنسية جعلت من
الرواية غير قابلة للتطور في شخص بطلها زكريا
المرسلى، فقد أفسدت النظرة الذكورية المتحكمة فى
داخلية القاص لاشعورياً، كل الوعى الإيديولوجي الذى
يملكه، إلاّ أن ذلك قد يحسب لصالح الروائى، لأن

ERIM Arabic Recognition Project Data

Figure 1: Machine Printed Arabic Text

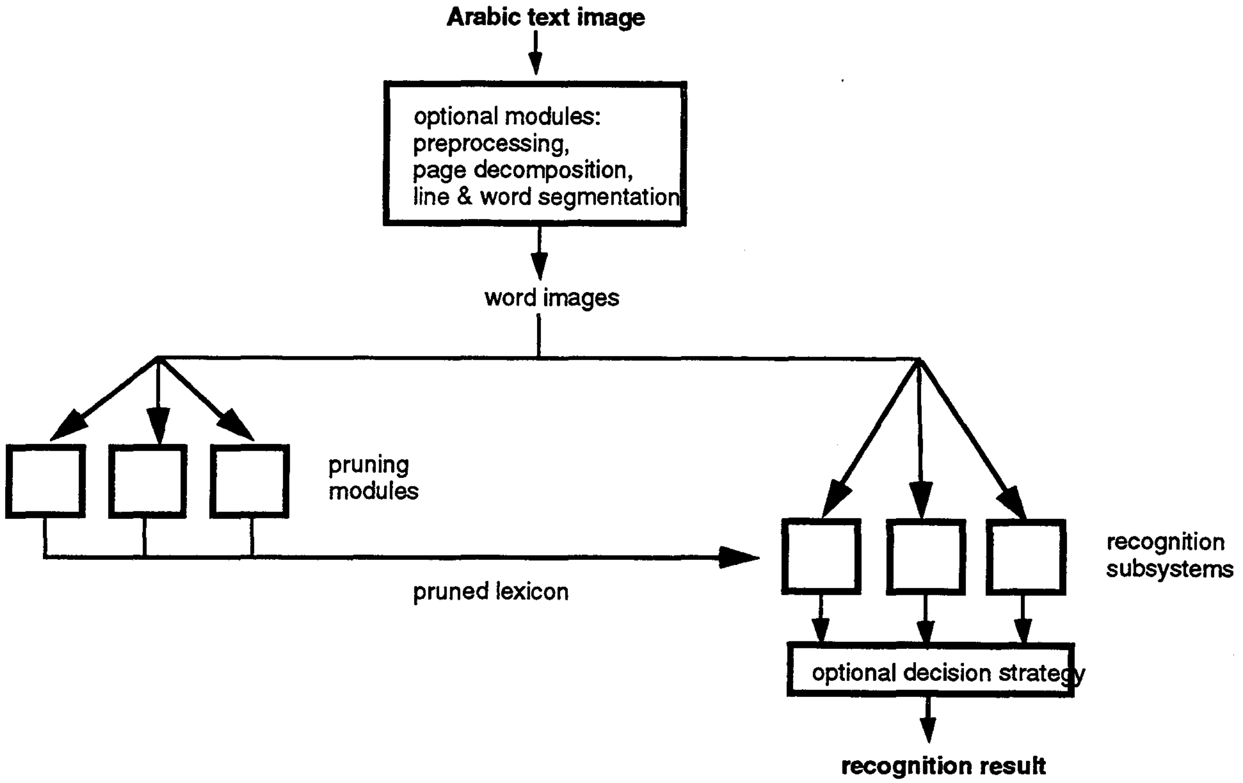


Figure 2: Overview of Arabic Recognition System Architecture

*over*segmentation, which breaks a word image into sub-character segments and reassembles those segments into characters. The third subsystem computes features on the word-image as a whole, and does not consider character-level information in any way. All three systems make use of a lexicon and lexicon-pruning techniques to constrain the recognition process.

2.1 System Architecture

The Arabic recognition system is designed to accept a variety of input imagery, ranging from word images to entire page images. Modular preprocessing steps are applied as necessary to reduce any imagery to a sequence of word-images. The system's page-decomposition module can extract lines of Arabic text from a full page image. Separate line-segmentation and word-segmentation modules may also be applied to preprocess smaller and more simplified text imagery. Once word-images have been extracted from the input image, they are passed into one or more of the recognition subsystems. If more than one subsystem is applied, then recognition results are combined using a decision strategy to produce a single recognition response. Figure 2 outlines the architecture of the Arabic recognition system.

2.2 Recognition Subsystems

Each of the three Arabic recognition subsystems uses a different approach to recognizing word images without character-level segmentation. The three subsystems are referred to as the oversegmentation system, the sliding neural network system and the whole-word recognition system.

2.2.1 Oversegmentation

The oversegmentation system is based on the process of segmenting a word-image at local extrema in the image contours. Segmenting in this fashion breaks the image up into pieces which are *smaller* than individual characters. These segments are combined into all possible unions of one, two, three or four segments. Each of these unions is scored by a neural network trained on whole-character images, and these scores are saved in an array. A dynamic programming algorithm is then applied to these scores to determine the best-cost way to assemble those unions into a spelling of a word from the word lexicon. The word lexicon is pruned for each word so that only a fraction of the possible words need be tested in this way. The word with the highest best-cost dynamic programming score is presented as the oversegmentation system's top choice. Figure 3 shows an array of segment unions created by the oversegmentation system. The word image from which the segments were extracted is in the upper left corner

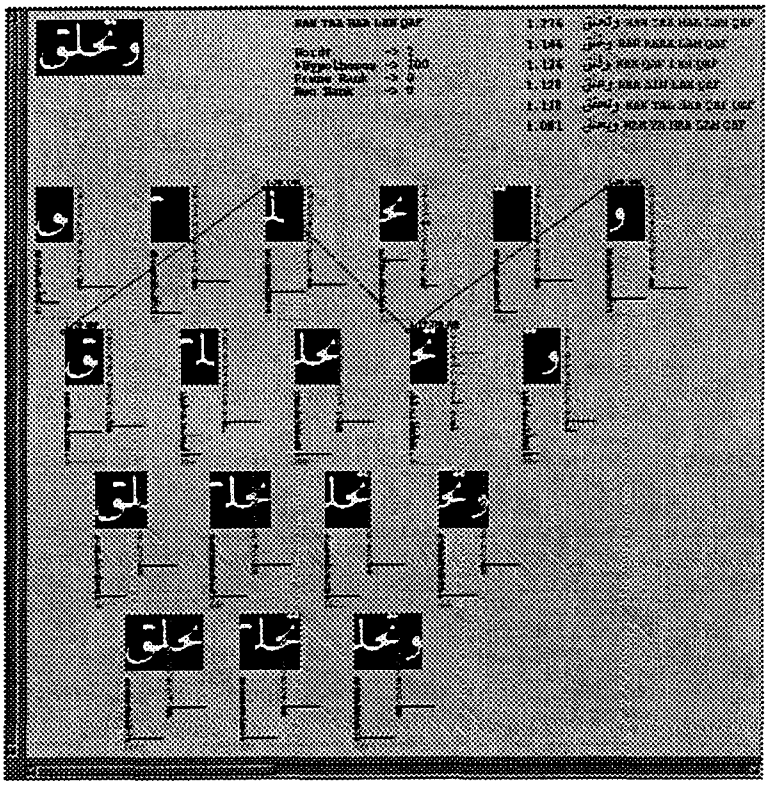


Figure 3: Segment Unions from Oversegmentation Subsystem

of the figure. The best-cost path through the segment array is shown as a dark line connecting four of the segment unions.

2.2.2 Sliding Neural Network

The sliding neural network system utilizes neural networks in a two-stage approach. The first stage is to detect probable locations of character centers and the second is to recognize characters at those probable locations. During both stages, the neural network acts as a window which slides across the word image and produces a recognition signal at each location along the image. This technique obviates the need for first trying to segment each character. The character hypotheses from the second stage are then combined with the aid of a dynamic programming algorithm, which uses a pruned list of words from the lexicon to constrain its search for the most probable word hypothesis. An example of the two-level application of sliding neural networks with recognition-signals is shown in Figure 4.

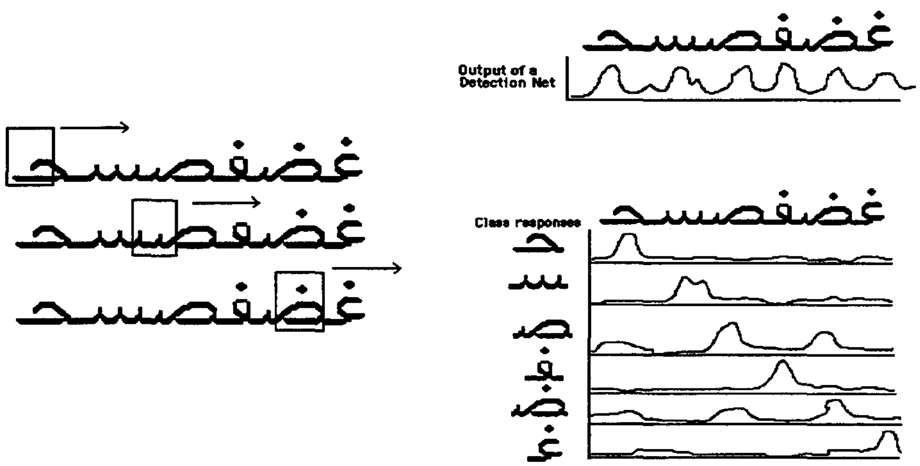


Figure 4: Signals from Detection and Recognition Sliding Neural Networks

2.2.3 Whole Word Recognition

The whole word recognition system is a feature-based approach that uses word-level features and a spatial-matching scheme. The feature set is based on binary image morphology operations, and includes such objects as line-segments at a range of orientations, holes, image skeleton endpoints and junctions. For this system, word-images are collected with their truth, and the feature set is computed for each word-image, along with noise models of that image. The various feature sets for each word are then stored in a database. An unknown word is recognized by computing its feature set and then comparing that set against a pruned subset of the database to determine the word which matches most closely with the unknown word. Some examples of features computed on a word image are depicted in Figure 5.

2.3 Lexicon Use

As was mentioned in the above sections, each recognition subsystem makes use of a lexicon of known Arabic words to constrain its recognition. The current system uses a lexicon of roughly 50,000 words compiled from our early truthing effort. Furthermore, this lexicon is pruned for each word-image so that a minimum number of possibilities need be considered by the recognition systems. Lexicon pruning has the double benefit of improving the recognition results and

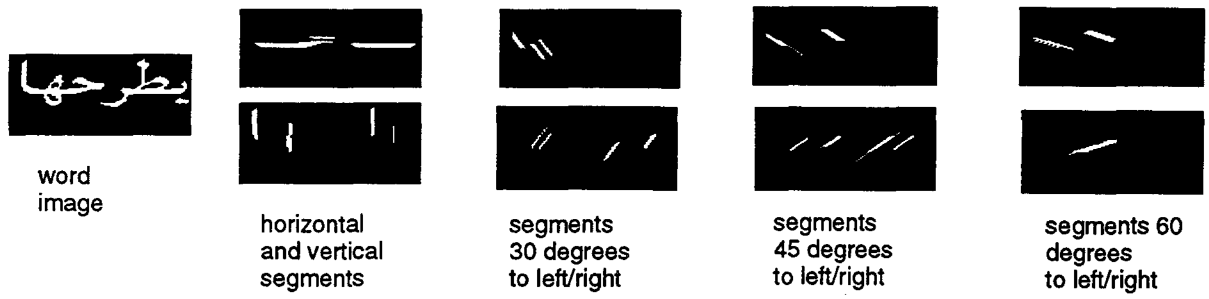

يطرحها
word image
horizontal and vertical segments
segments 30 degrees to left/right
segments 45 degrees to left/right
segments 60 degrees to left/right

Figure 5: Example of Morphological Features Extracted by Whole-Word Recognition

increasing the speed of the system.

Several pruning techniques have been implemented and incorporated in the Arabic system. Each of these methods is essentially an abbreviated version of a full-scale recognition system. The purpose of the abbreviated version is to quickly eliminate words in the lexicon which are unlikely to be the correct choice, and let a full recognition system discriminate between the remainder of the choices. The current system includes pruners based on morphological feature extraction, oversegmentation and sliding neural networks. Hybrid schemes employing combinations of these methods have also been implemented.

3 Training Data

Each of the recognition subsystems was trained on a set of data collected during the early phases of this project. Roughly 700 pages of Arabic text were digitized into greyscale imagery. These pages were collected from 45 documents from the University of Michigan's Arabic library. The text was chosen to include a range of fonts and sizes, including typewriter as well as type-set characters. Digitization produced varying degrees of noise quality, encompassing both high quality and noisy text.

The digitized pages were sectioned into sub-images called zones, and these zones were truthed using an Arabic word processor. The truth text from the word-processor was converted from extended ASCII into Unicode for use by the Arabic recognition training systems. Zone images were binarized and line, word and character information was extracted to be used in training. Binarized word images were also extracted and saved for the purpose of convenient testing.

4 Recognition Results

The Arabic recognition system can be configured to use the full cross-product of pruning systems, recognition systems and their hybrids, which provides dozens of possible system results. Results for a few of the most commonly used system configurations are described below to give a flavor for the system's performance. All results were generated from a random sample of 8436 word images taken from the training data.

4.1 Subsystem Results

The sliding neural net subsystem obtains its best results in conjunction with an oversegmentation pruner based on digram extraction. Its word recognition rate on the test set is 70.0% This is also the fastest configuration using the sliding neural network system.

The oversegmentation system obtains its best results using a hybrid pruner based on the oversegmentation and morphological feature systems. Its recognition rate in this configuration is 80.7%. The system's fastest configuration is with the digram-based oversegmentation pruner, which gives a recognition rate of 77.4%

The whole-word recognition system achieves its fastest and highest recognition performance with the digram-based pruner. The system's recognition rate in this configuration is 65.0%

4.2 Hybrid System Results

The recognition subsystems can be combined using decision strategies to obtain some increase in recognition performance. The highest recognition performance is obtained using a combination of the sliding neural network and oversegmentation systems, which obtains a recognition rate of 81.4% in conjunction with the hybrid morphological and oversegmentation pruner.

5 Arabic Text Database

One of the byproducts of the training data collection and processing was a large body of Arabic text imagery with detailed truth information. Because Arabic text imagery and truth data are not easy to obtain, it was decided to publish this training data as a CD-ROM for other Arabic recognition researchers to use.

The result of our publication efforts is a CD-ROM containing the binary zone images from all 700 of the digitized pages. For each zone image there is a truth file which provides the locations of all lines, words and characters in that zone. Character locations were extracted semi-automatically, and therefore are given with a confidence measure to help compensate for the error rate of the automatic extraction.

In addition to the full set of binary zone images, a subset of the original greyscale page images was also included in the CD-ROM, to facilitate the development and testing of full-page preprocessing applications.

The authors may be contacted for more information on obtaining this CD-ROM, which is available for a nominal fee.

6 Status

Our work on the Arabic recognition system is continuing on several fronts. We are in the process of planning a second data collection and truthing effort to extend our training data to a broader range of document styles, for instance newsprint documents. In addition, a new approach to automatically generate training data is being developed, which will allow text imagery to be generated artificially with the exact location and content of the text. This technique will allow us to reduce the impact of truthing errors on our training algorithms.

We are also expanding our recognition capability to incorporate lexicon-free recognition systems. We have obtained preliminary results from a lexicon-free recognition system based on oversegmentation. We also plan to explore a system based on recognizing subwords, which are the sub-strings of a word that are connected in the Arabic script language.

Finally, we are expanding our page decomposition module so that it can be applied to a wider range of document styles and print sizes.

Bibliography

Al-Enami, Samir and Usher, M., On-line recognition of handwritten Arabic characters, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(7) (July 1990) 704-710.

Almuallim, H. and Yamagushi, S., A method of recognition of Arabic cursive handwriting, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-9(5), Sept. 1987, 715-722.

Amin, A., Masini, G., and Haton, J.P., Recognition of Arabic words and sentences, in *Proc. of 7th International Conference on Pattern Recognition*, Montreal, Quebec, 1984.

Amin, A., and Masini, G., Machine recognition of multifont printed Arabic texts, in *Proc. of the 8th International Conference on Pattern Recognition*, Paris, France, 1986.

Badie, K. and Shimura, M., Machine recognition of Arabic cursive scripts, *Trans. Inst. Electron. Commun. Eng. Japan*, Vol. E65(2), Feb. 1982, 107-114.

Beeston, A.F.L., Written Arabic-An Approach to the Basic Structures (Cambridge University Press, 1968).

El-Dabi, S.S., Ramsis, R. and Kamel, A., Arabic character recognition system: a statistical approach for recognizing cursive typewritten text, *Pattern Recognition*, 23(5) (1990) 485-495.

El-Khaly, F. and Sid-Ahmed, M.A., Machine recognition of optically captured machine printed Arabic text, *Pattern Recognition*, 23(11) (1989) 1207-1214.

Trenkle, J.M. and Vogt, R.C., Word Recognition for Information Retrieval in the Image Domain, in *Proc. Second Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas, NV, April 1992, 105-122.

Arabic Character Recognition using Integrated Segmentation and Recognition

Mosfeq Rashid **Gale Martin**
Microelectronics & Computer technology Corp. (MCC)
3500 W. Balcones Center Drive
Austin, Texas 78759
email: mosfet@mcc.com, galem@mcc.com

Abstract

We are applying of Integrated Segmentation and Recognition technique developed for handwritten and degraded machine printed English to recognize machine printed Arabic. In this system, the large input window is used by the neural network based classification to avoid problems introduced by premature segmentation. The system integrates mechanism to move the input window appropriately by emulating of saccadic movements of human eye. It takes into account the confidence level of how well a character has been recognized and contextual information in the input window. Several attributes of this system suitable to connected and cursive nature of Arabic writing, especially in case of degraded prints we often encounter in our database. In moving from English to Arabic, we augmented our with additional features to optimize the recognition process: introduction of modular approach, and application of Gabor wavelet based preprocessing. Our goal is not only develop a high performance system, but also to explore the use of increasing knowledge of biological vision system. Our experiments indicate that the success of Integrated Segmentation and Recognition algorithm is applicable to Arabic script.

1 Introduction

We human have the intrinsic ability to effortlessly decipher the content of documents of many different qualities. Although automated document processing has come a long way, the technology has yet to achieve our level of performance. The poorly understood mechanism behind computation in biological system is quite different from that of digital computers. A better understanding of this computation, nevertheless, is gradually emerging, particularly for the early stages of sensory processing. Some of these mechanisms do not fit well with our present computational paradigm. But we may be able to effectively adapt the principles behind the biological implementations for building better artificial systems.

In an effort to apply a few of the sophisticated principles behind biological vision system, we have developed Integrated Segmentation and Recognition technique for optical character recognition. Our emphasis is to take a line of text and recognize its content. In order achieve high performance document processing system, several other aspects of the system has to optimized in addition to recognition. Many of these are domain specific. The knowledge of the composition of the documents can be used to improve the performance of the line extraction phase. Similarly at the post-processing stage using domain knowledge to do dictionary lookup or apply other constraints can markedly improve the performance. These issue are more fully addressed in *The Second Census Optical Character Recognition Systems Conference* which brought together several current state-of-the-art systems, including Integrated Segmentation and Recognition, in a competitive forum [1].

We have successful used this system to build high performance systems for recognizing handwritten and degraded machine printed English before our current effort to use it with Arabic. We integrate various aspects of processing around a neural network based classification and prediction mechanism. The system then uses other aspects of biological vision systems to integrate segmentation and recognition. The two principles we apply in our system are the saccadic eye movement and Gabor- wavelet orientation sensitive edge detectors.

2 Traditional Approaches

Traditional techniques for Arabic has been used with some time with limited success [2]. Until recently most of the efforts were quite limited. Now powerful techniques using neural networks along with the application of other algorithms at pre/post-processing stages are under development [3].

There are several problems if one uses a conventional approach to character recognition systems [4–6]. In such systems the task is broken into several manageable subtasks each feeding the next one (figure 1). This reduces the number of free input parameters at each stage of processing, simplifying the implementation. Segmenting a character from its background focuses subsequent operations on a single character (or parts of the character which are

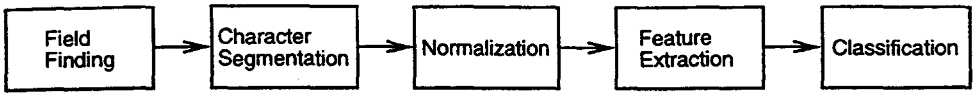


Figure 1: Conventional pipelined architecture.

combined later), and feature extraction reduces a high-dimensional character image to a compact feature list. Reducing the number of free parameters input to a classifier also tends to improve generalization accuracy, assuming a fixed-size training set.

At the same time significant contextual information is lost in this pruning. This arbitrary division does not guarantee that the necessary information is preserved from one stage to the other. Without enough context it is difficult to determine whether a line segment is a character or a part of a character. Contextual information in Arabic is more important because location provides additional cues for classifying a character. One example is the inability of the system to determine where the baseline is located. Another is the common technique for simplifying the feature extraction phase by skeletonizing the image of the character; this often removes relevant information necessary for classification. Additionally, inability to separate out a character overlapping or touching other characters is also a serious limitation of this approach. It is also valid when a part of a character is used and combined later for classification. In such situation, the segmentation phase incorrectly passes several characters confusing the following stages. The dots and diacritics widely used in Arabic further increases the complexity.

In the real world fading out, missing pixels, and other degradations, introduced by copying, fax machines or other environmental factors break the characters and the connections. The segmentation algorithm often fails in these cases by confusing boundaries of characters. In addition, the extracted features no longer correspond to the feature set of the system. Generating a feature set is also a time consuming process. The increasing opportunity to use a wide variety of printing styles is becoming more common with the availability of inexpensive and powerful computers. The inflexibility of generating new features, along with other drawbacks, makes its limitations more obvious, especially when one can use more generalized adaptive approach.

In conventional systems classification fails because the decomposition task is unable to incorporate the necessary interdependence between segmentation and classification. Correctly segmenting a character from its background often requires first recognizing it, and correctly recognizing it requires first segmenting it. The pipeline architecture of conventional character recognition systems is fundamentally flawed because it does not allow recognition and segmentation decisions to interact.

The problem is actually more general than the interdependence of segmentation and recognition. Ideally, recognition decisions should interact with all forms of preprocessing and postprocessing. This includes contrast enhancement, size-normalization, de-skewing, rotation, and even field finding; the degree of inter-dependence vary for different types of processing. For example, the best contrast level for viewing a handwritten character is the level that makes recognition easiest, not some preset, constant value. Similarly, finding and isolating fields of characters is best done by recognizing that a string of characters exists at a given location. For example, conventional line segmentation algorithms, based on density histograms, fail because they do not distinguish between irrelevant visual noise and relevant fine-grained details. Similarly knowledge at the word level can help the recognition process. Once again, the pipeline architecture of conventional character recognition systems is fundamentally flawed because it assumes that other processing stages are necessarily independent of recognition decisions. The inter-dependence of pre/post-processing and recognition decisions becomes more critical for printed Arabic scripts than for many other types of machine print recognition, because of wide range of acceptable variations with possible degradation of quality.

3 Principles of Integrated Segmentation and Recognition

Our first effort was devoted to integrating feature selection and classification. Initial applications of neural networks simply replaced the classifier component in the pipeline architecture with a neural network [7, 8]. We demonstrated that a neural network can be trained to classify characters directly from images of presegmented, size-normalized characters [9]. This approach frees the system designer from the task of feature selection. The net automatically and simultaneously optimizes feature selection and classification.

This previous success, of using an image array as the input to a net, suggested that it might be possible to integrate character segmentation and recognition within one function that is learned by a neural network [10], shown in Figure 2. Within this scheme the neural net is trained to recommend segmentation transformations as well as recognize characters, and then during run-time, the recognition decision drives the decision of whether the present segmentation is incorrect and hence a new attempt at segmenting is necessary.

3.1 The Saccadic subsystem

The system, called Saccade, provides a different way of thinking about automatic character recognition

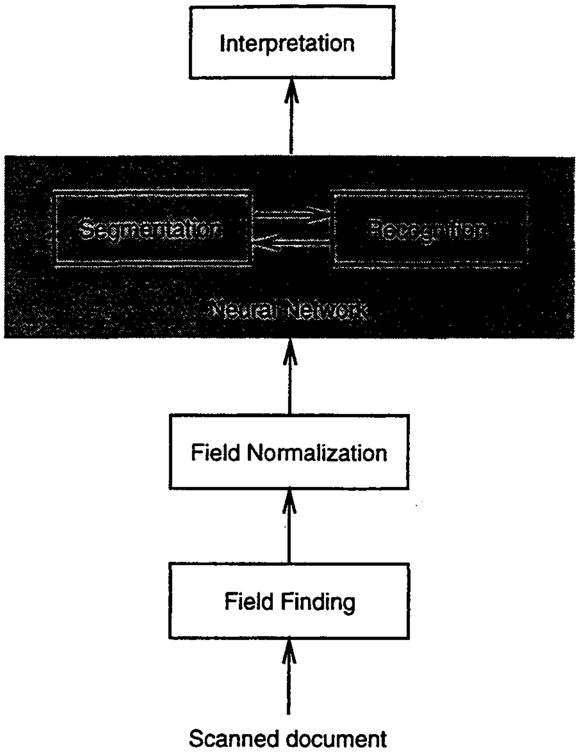


Figure 2: Integrated Segmentation & Recognition approach.

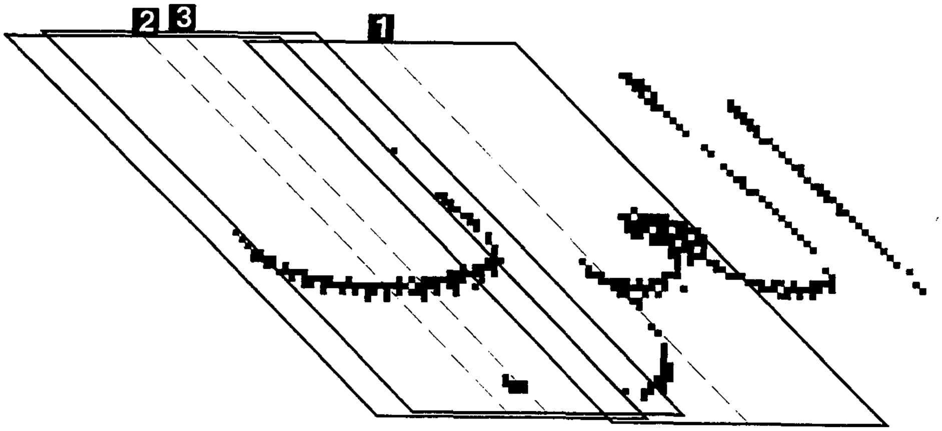
and segmentation. It assumes that segmenting characters into little boxes is the wrong way to think about the problem. Rather, segmentation should be thought of as a centering operation involving a focus of attention. A character is correctly segmented if it appears in the center of a large window, regardless of what else appears in that window. In addition, the approach assumes that a neural network should be able to learn more than to simply recognize what is in it's input window; the net should be able to learn to control what comes into it's input window. The approach takes a cue from the ballistic and corrective saccades (eye movements) of natural vision systems. Saccades make it possible to efficiently move from one informative area to another by jumping. The eye typically initiates a ballistic saccade to move the center of focus to the general area of interest, followed by one or more corrective saccades for fine-grained position corrections. Thus, the approach involves training a neural net to center a character in its input window, recognize it, and jump to center the next character in it's input window, and so on. This operation applies to both non-touching and touching characters.

During training, the input to the net comes from a two-dimensional window that scans horizontally across a field of characters, as shown in figure 3. The field image is first size-normalized to the height of the input window. The input window is chosen to be wide enough to contain several characters. As the input window scans across the field, the center of the window will pass over the centers of characters and over points between characters. The field image is first size-normalized to the height of the input window. The network is trained to answer four questions at each point in the scan:

1. Is there a character centered in the input window?
2. If so, what character is centered in the window?
3. How far off-center is the character?
4. How far is the next character to the left?

These questions can be answered during training because, prior to training, the field images are labeled with the horizontal center positions of each character in the field, as well as with the categories of the characters. Training involves pairing each input window with a 3-part target output vector representing what is centered in the input window, the distance by which the centermost character is off-center, and the distance from the center of the input window to the next character to the left. If this distance exceeds half the window width, (i.e., the center of the next character to the left is not visible in the input window), the distance value is set to a maximum of half the window width. Thus, during training the net learns to compute an input-output mapping function, where at any given point in the scan, the input corresponds to what is in the scanning window, and the target output is the 3-part vector, described above, corresponding to what is in the scanning window. The present form of the system separate the step prediction and classification into two separate networks in a modular fashion, which we will describe later. Note that the net does not simply learn the average distance between characters, since this is likely to vary with different characters and different type of formatting. The net is trained on very large database of fields. As a result, the net learns to use the perceptual cues in the input window, to estimate the distance it should jump to correctly center the next character to the left (ballistic jump), and to make fine-tuned position corrections to correctly center the character closest to the center of the window (corrective jumps).

During run-time or testing, the system uses the computed values in the character classification and distance fields of the output vector and some heuristics to navigate horizontally along a character field, jumping from one character to the next, and occasionally making a corrective saccade to improve its ability to classify a character. When the net recognizes a character, it executes a ballistic saccade to the next character. It obtains the distance to jump by reading the next character distance field. When the net fails to recognize a character, it executes a corrective saccade to better center the character. It obtains the distance and direction to jump by reading from the current character distance field. The system works correctly for non-touching, integral characters as well as for touching and broken characters.



1 'ray' is recognized at this position with sufficient confidence; the system attempts to jump to following character.

2 The character 'bay' is not at center of the input window for best recognition; the systems attempts to make a corrective jump.

3 The optimally positioned to recognize 'bay'.

Figure 3: Saccadic jumping mechanism.

3.2 Modular saccade for Arabic

In modifying the design of the system for machine printed Arabic character recognition, we decided to pursue a more modular approach while maintaining the advantages of our earlier Saccade system. In this system, instead of one network doing classification and stepping, there are two separate neural networks doing the two tasks. This allows us to have modularity in an integrated system which helps to better optimize of each task and their interaction. The integration of the response of the single neural network in previous system involved several empirically developed heuristics to enable effective scanning of the line and to determine positions at which recognition should be performed. The modular approach make it possible to embody a significant portion of these heuristics in the neural networks, reducing the complexity of the overall system.

Initially the classification network is trained to determine the character class and whether the character is of initial, medial or terminal form. The number of output nodes of the classification network is 39 and they are marked as: *Ḥamza, Alef, Waw, Beh, Teh, Theh, Jeem, Hah, Khah, Dal, Thal, Reh, Zain, Seen, Sheen, Sad, Dad, Tah, Zah, Ain, Ghain, Feh, Qaf, Kaf, Lam, Heh, Yeh, Tehmarbuta, Alef-maksura, Diacritical-mark, Unknown, Space, Between-symbol, Overlap, Initial form, Medial form* and *Terminal form.* The Diacritical-mark node indicates the presence of vertically overlapping diacritical mark (which includes hamza), Unknown node represents characters which are not represented by the first 31 nodes representing known character classes, Space node indicates inter-word spacing, Between-symbol node represents the region between two symbols, Overlap indicates two characters (includes diacritical mark) are vertically overlapping, and the last three node represent the form of the classified characters.

The step prediction network has 25 output nodes: the first indicate that corrective jump is necessary, the second redundantly (*Between-symbol* node of the classification network) indicate whether the network is positioned between two characters symbols, the third indicate when to classify, and the rest of the nodes encode jumping distance. Of the nodes reserved for encoding, 2/3 are used for ballistic jump and 1/3 are used for corrective jump. We use a coarse coding using triangular (convex) function to encode the distance value. The redundancy introduced by this method makes prediction more error tolerant.

3.3 Gabor-Wavelet subsystem

The input window we used for Arabic was larger than that of English. Consequently, the dimensionality of the input to neural networks were higher making the training process more difficult. Although we had good classification performance, the performance of the stepping mechanism was well below what we achieved in the case of English handwriting. Our initial assumption that the modularity we introduced in case of Arabic may have made the learning task more difficult proved to be false. Our experiments with the earlier architecture did not improve the situation.

We decided to take a new approach of applying Gabor wavelets to raw input image. A 2D Gabor wavelet is a Gaussian modulated by a sine or cosine wave, and it is general complex form is [11]:

$$G(x,y) = \exp[-\pi\{(x-x_0)^2\alpha^2 + (y-y_0)^2\beta^2\}] \\ \times \exp[-2\pi i\{u_0(x-x_0) + v_0(y-y_0)\}] \quad (1)$$

The 2D shape of the Gaussian component of $G(x,y)$ is determined by α and β and its shift by x_0 and y_0. u_0 and v_0 specify the frequency components in two directions.

We use a simpler form with two basic wavelets with circular Gaussian. We can scale, rotate and shift these wavelets across the image to obtain a set of wavelets that allow reasonable reconstruction of the transformed input image.

$$G_{\sin}(x,y) = \sin(\omega x)e^{-(x^2+y^2)/\sigma^2} \quad (2)$$

$$G_{\cos}(x,y) = \cos(\omega x)e^{-(x^2+y^2)/\sigma^2} \quad (3)$$

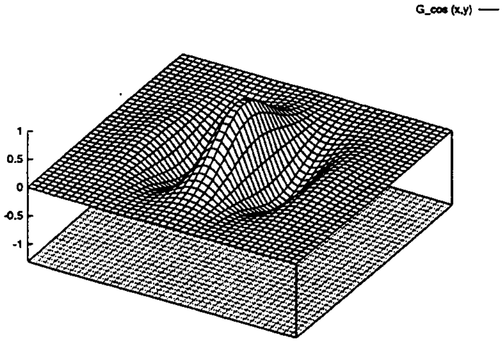


Figure 4: Plot of $G_{\cos}(x,y)$.

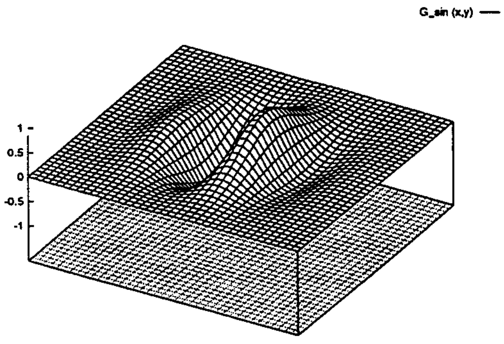


Figure 5: Plot of $G_{\sin}(x,y)$.

There is evidence that this approach can improve in optical character recognition type applications [12]. Gabor wavelets match the response characteristics of early stages visual processing in the cortex [13]. And there are several good technical reasons to have such wavelets:

- They reduce the dimensionality of the input.
- They add noise immunity to the input image.
- They are local both in space and frequency, allowing the option to maintain efficient local receptive field structure of the neural network.
- They avoid loss of information from scaling down the input image.
- They provides a degree scale, translation, and rotation invariance.
- Within a limited range two wavelets can interpolate between the two extremes of their properties (scale, translation and rotation) [12].
- The neural network is not burdened with learning generic features that these wavelets provide.

On the other hand, being non-orthogonal in nature, Gabor wavelets are computationally not as efficient as other commonly used wavelets. The advantages, however, outweighs this shortcoming, considering the evidence that evolution picked a similar approach with built-in redundancy [11].

Our first attempt is to use this mechanism with step prediction network to achieve a performance level comparable to that of the classification network. We selected three Gabor wavelets of different frequency (ω) and scale (σ). Both sine (equation 2 & figure 5) and cosine (equation 3 & figure 4) wavelets are then replicated and shifted across the input image, which makes the total count 50. Each of these have four different orientations making the total number of input features 200. This selection was based on manually looking at the reconstruction of the transformed input images. Automating the selection process together with training the neural network may allow us to obtain a more compact and efficient set of wavelets. In addition, having Gabor wavelets at the preprocessing stage allow the possibility of achieving our long term goal of integrating size-normalization, rotation and contrast change.

4 Experimental Status

The system is trained using a data set of pages containing more 100,000 Arabic characters. Each character was labeled to mark their center and the bounding box. We divide the data into three sets: 80% of the data for training, 10% for testing and the other 10% of validating the final performance of the system. First we trained the classification network so that to get best possible score with the test set. At the peak point we validate the performance with the validation set.

We picked twelve different neural networks with three different input window sizes and four different hidden node configurations. After initial batch of training runs, we found that taller input window size and larger number hidden nodes contribute to rapid training performance improvement. The impact of various local receptive field size of the first layer of the neural network was not significant. After extensive training and testing we chose three best performing networks.

The following table shows the classification performance result of some the network we have trained. The first column for each result show the only character recognition accuracy, and the second score includes the other symbols of the classification network. In the following results it is interesting to note that the *Net-1* is the smallest classification network; *Net-2* is 16% bigger and *Net-3* is 35% bigger than *Net-1*. This show that the smaller network possible with the use Gabor wavelets can improve the performance even further.

Table 1: Classification performance.

Networks	Test results (%)		Validation results (%)	
	Characters	*Overall*	*Characters*	*Overall*
Net-1	90.39	89.08	88.16	87.40
Net-2	89.50	87.11	87.14	85.02
Net-3	86.77	84.08	83.78	81.69

In our experimentation with Gabor wavelets we have found that changing the size of the input window as much as 25% of the input window even when we do not proportionally change the ω and σ. We only proportionately changed the spacing. This property can make the system tolerant to size variations without much adaptive tuning of wavelet parameters. During training the step prediction error rate is significantly lower error than we could achieve with several of our earlier effort. We will provide result when we have extensively tested the stepping subsystem as it moving on its own.

5 Conclusion

The step prediction network is at its early stage of development. The classification network performing well, and we hope to have further improvement when we use Gabor wavelets system with the classification network. We are planning to expand the scope of the Integrated Segmentation and Recognition system by using a larger Arabic character data base with a wider variation of fonts.

Acknowledgments

The research is presently funded by DoD contract number MDA-904-93-C-7203.

References

[1] J. Geist, R. A. Wilkinson, S. Janet, and et. al., *The Second Census Optical Character Recognition Systems Conference*, (1994).

[2] P. Ahmed, and M. A. A. Khan, Computer Recognition of Arabic Script Based Text — the state of the art, in *Proceedings of the 45th International Conference and Exhibition on Multilingual Computing* (1994), 2.2.

[3] K. M. Hassibi, Machine printed OCR using neural network, in Proceedings of the 45th International Conference and Exhibition on Multilingual Computing (1994), 2.3.

[4] H. Gorian, M. Usher, S. Al-Emami, Off-line Arabic character recognition, *Computer*, (July 1992), 71–74.

[5] F. El-Khaly and M. A. Sid-Ahmed, Machine recognition of optically captured machine printed arabic text, *Pattern Recognition*, **23** (1990), 1207–1214.

[6] S. S. Dabi, R. Ramis and A. Kamel, Arabic character recognition system: a statistical approach for recognizing cursive typewritten text, *Pattern Recognition*, **23** (1990), 485–495.

[7] D. J. Burr, A neural network recognizer, *Proceedings 1986 International Conference of Systems, man, cybernetics.*, (Atlanta, GA), 1621–1625.

[8] J. S. Denker, W. R. Gardner, H. P. Graf, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, H. S. Baird, And I. Guyon, Neural network recognizer ofor hand-written zip code digits, in *Advances in Neural Information Processing Systems 1*, D. S. Touretzky ed. (Morgan Kaufmann Publisher, San Mateo, CA, 1992) 323–331.

[9] G. Martin, and J. Pittman, Recognizing hand-printed letters and digits. , in *Advances in Neural Information Processing Systems 2*, D. S. Touretzky ed. (Morgan Kaufmann Publisher, San Mateo, CA, 1990).

[10] G. Martin, M. Rashid, D. Chapman, & J. Pittman, Learning to see where and what: training a net to make saccades and recognize handwritten characters, in *Advances in Neural Information Processing Systems 4*, J. E. Moody, S. J. Hanson, and R. P. Lippman eds. (Morgan Kaufmann Publisher, San Mateo, CA, 1992) 441–447.

[11] J. Daugman, Complete discrete 2-D transforms by neural networks for image analysis and compression, *IEEE Trans. on Acoustics, Speech and Signal Processing.* **36**(7) (1988) 1169–1173.

[12] A. Shustorovich, A subspace projection approach to feature extraction: the two-dimensional Gabor transform for character recognition, *Neural Networks* **7** (8), 1295–1301.

[13] D. Hubel, and T. Wiesel, Sequence regularity and geometry of orientation columns in the monkey striate cortex, *Journal of Computational Neurology*, **158**, (1974), 267–293.

Feature Selection Given Large Numbers of Classes and a Large Feature Universe

Ivan Bella   **Garrett Macey**

Loral Federal Systems, 700 N. Frederick Ave., Gaithersburg, MD. 20879, USA

Abstract

An Automated feature selection method based on Kudo and Shimbo [1] was implemented to improve a character recognition engine. The approach selects a subset of features from the universe of features such that recognition accuracy against a target data set is optimized. The method was altered to deal with large numbers of classes (thousand+) and a large universe of features (hundreds). This allows the collection of disparate feature sets up front, without regard to back-end problems of over-training or over-sampling of the data space.

The feature set is narrowed to the number of features that provide the best within-class consistency and cross class discrimination. The approach frees the researcher from intuitively selecting the features used in a classifier. Instead, any interesting features are added to the feature universe and the selection method determines which features, if any, are replaced from the currently used set.

Experimental results are provided, along with a discussion of computational feasibility for various sized problems. A parallel implementation is discussed. A comparison of accuracy is provided for the selected heterogenous feature subset versus complete homogeneous feature sets. Discussion is provided on the effectiveness of the method when applied to various datasets such as Japanese characters or English characters.

1 Introduction

This paper examines the implementation of an automatic feature selection algorithm based upon the work of Kudo and Shimbo [1,2]. The feature selection algorithm was pursued to allow the automatic selection of a reasonable subset of a large feature universe without a priori knowledge of the source language characteristics. The segmentation and feature generation for the individual characters is not considered within the scope of this work.

There is a vast body of research into recognition of character data [3,4,5,6,7,8]. With each advance and language studied a new set of features is proposed to handle specific problems or attributes of the source data set. The addition of new features is not sufficient to develop a better image character recognition (ICR) engine. The discrimination rate of an ICR engine will often increase with the addition of a few features. However, peaking phenomena are common when a large number of features are considered. After this peak, discrimination rates actually decrease with the addition of new features. Additional features also increase the computation time required to perform the recognition.

The task of defining an appropriate feature set is generally the responsibility of the researcher. The selection of features is driven by a priori knowledge about the source data set.

An automatic feature selection method was pursued to reduce the loss of accuracy due to peaking phenomena while allowing the examination of a broad feature universe. The automated feature selection method was pursued to support the development of a tool to create ICR engines. Given a training dataset, a feature set could be generated that performed well against that data source without a priori knowledge of the language characteristics. In this manner a researcher could quickly develop an appropriate feature set to handle Kanji, Cyrillic, English, or any other character set of interest given a reasonable set of training data.

2 Goals

This primary goal of this investigation was to increase the accuracy of an existing ICR tool. However, accuracy is not the only measure of importance to an ICR engine. There were four other areas of interest in relation to this study. Each of these four areas is presented in decreasing order of importance.

Automation: The universe of features used in character recognition is very large. The selection of an appropriate feature space is dependent on the characteristics of the source data. The feature selection algorithm was designed to allow the researcher to examine a wide feature space without a priori knowledge of an efficient feature set.

Comparison Efficiency: The final ICR engine is designed to handle pages of text and, therefore, a large number of individual characters. The evaluation model needs to support rapid processing.

Performance: The automatic feature selection process is run once against a training set of data. On a small number of classes it is desirable that the operation be interactive. In an interactive environment, the researcher would select a small training set of data and generate a customized recognition engine that would be used against any remaining data. Against a large number of classes the algorithm's performance would prohibit interactive usage. When a large number of classes are considered the training data would be generated and saved. The feature selection algorithm would then be started and allowed to run until termination. Upon termination the generated template library would be used for further recognition of source data.

Generation Efficiency: The ICR process will need to generate the feature vector for the characters within the source document. This generation time should be minimized to support high volumes of source material.

3 Background

This paper discusses an implementation of the algorithm described by Kudo and Shimbo [1,2]. The details of the algorithm are provided only by reference. Cursory knowledge of the approach is required to understand the implementation. A synopsis of the base algorithm has been provided in section 3.2.

The terminology and notation used for referencing the data elements within this paper are consistent with the terms used in the referenced work [1]. A summary of the major terminology is included in section 3.1.

Finally, the feature selection algorithm was implemented within the framework of an existing ICR engine development tool. A few of the design decisions were influenced by this framework. A brief overview of the ICR engine is provided here for completeness.

The existing ICR tool handled character segmentation, normalization, feature generation, and final recognition. During feature generation the ICR tool would generate three types of features (see 6.1). On the Kanji test dataset these three feature sets represent 692 individual features. The recognition engine was based upon a template library. Source characters were compared against the templates in the library and the codepoint which minimized a distance function was selected. The existing ICR tool was past the optimal discrimination rate when all features were considered.

3.1 Terminology

Class : All samples of a given codepoint (character) within the training data set.
Subclass : The set of samples within a single class which share some common attribute(s).
Feature Vector : The set of measurements for the feature universe for a single sample ($\vec{v}$).
S+ : The class under consideration
S- : The set of all classes within the problem space other than the codepoint under consideration (S+)
Maximal : Used to describe a subset, where the addition of any of the remaining vectors from S+ would violate the common attribute(s) of the subclass.
Exclusive : Used to describe a binary vector. Exclusive indicates that the binary vector is distinct from all vectors within S- (eq. 1).

$$\vec{v} \wedge \vec{s} \neq \vec{v} \quad \forall \vec{s} \in S- \qquad (1)$$

G : A subset of S+, also known as a "view"
α(G) : The binary vector resulting from the logical conjunction of the samples represented by G
Ω(S+,S-) : The collection of all the maximal subsets of S+ which are exclusive against S-
MAX : Number of samples within the largest subclass occurring in Ω(S+,S-) (eq. 2).

$$Max(\Omega) = \max_{G \in \Omega} \frac{|G|}{|S+|} \qquad (2)$$

AVE : The average number of samples within the subclasses occurring within Ω(S+,S-) (eq. 3).

$$Ave(\Omega) = \frac{1}{M} \sum_{G \in \Omega} \frac{|G|}{|S+|} \qquad (3)$$
$$where\ M = |\,\Omega\,|$$

3.2 Basis of Algorithm

Kudo and Shimbo [1] discuss a six step procedure for reducing a feature space. The six steps consist of binarization, data reduction, identification of subclasses, feature evaluation, feature reduction, and peaking determination. Each step will be identified by a roman numeral to assist in later referencing.

I: The [1] algorithm is based upon the evaluation of hyper-rectangles and the ability of these rectangles to discriminate between classes. The boundaries of the hyper-rectangles are established upon a binary feature space where each bit indicates the relationship of the real feature value to a boundary within the minimum and maximum values for the feature across all samples.

II: Data reduction is the combining of the binary vector spaces so that the number of samples within a single class is within a range which is computationally feasible. [2] suggests a guideline for reduction of 40 to 60 samples within S+. The large number of classes and total number of samples required the reduction of both S+ and S- with stricter guidelines. The guidelines used within this implementation are discussed in section 4.3.

III: Identification of subclasses identifies maximal subsets of S+ which are exclusive against S-. This step is discussed in [2]. The referenced method is a recursive algorithm which examines subsets of S+. This is a computationally demanding process. Each iteration requires $\binom{N}{M}$ steps, where N is the number of samples under consideration and M represents the desired number of samples within the subset. The algorithm is designed such that N starts as the number of samples in S+ but is reduced on each iteration. At the same time M begins at 1 and increases.

IV: Feature evaluation determines within a single subclass the contribution of each feature towards the ability to discriminate the subclass from S-. The base algorithm examines each feature dropping any feature which does not contribute towards discrimination. A pair of statistics are generated for each remaining feature. The statistics represent a measure of how many samples from the class are within the subclass and a measure of how important each feature is to discriminating the subclass from S-. The values for each subclass are then combined to generate a set of values for the class. These class feature metrics are further merged into metrics evaluating the features contribution across the entire set of classes.

V: Feature reduction determines which features contribute the least across the entire set of classes. These features will be removed from consideration. The algorithm drops features if they fail to reach a predetermined significance level. If all features are found to be significant then the feature with the lowest contribution metric is discarded.

VI: Finally, peaking determination is the mechanism used to determine if the process (II thru V) should be repeated against the reduced feature space. The peaking determination is done by examining the rate of change within the significance metrics (MAX, AVE).

4 Implementation

A complete algorithm was provided within [1]. Our implementation varies from the base algorithm in a number of areas. These variations were prompted by the exact nature of the examined problem and the implementation environment. A brief overview of the implementation and areas where modifications were made is provided. The examination is divided into five sections. The first two sections, data design and algorithm, present the general structure of the implementation. This is followed by two sections, problem size and running time considerations, which discuss design decisions that were made due to the nature of the studied problem. The design decisions discussed in these sections did not alter the basic operation of the base algorithm as described in [1]. The final section, modifications to algorithm, details the changes that were made to the actual algorithm. The changes within this final section actually modify the behavior of the base algorithm. Within each section roman numerals will be used to reference back to the original algorithm as described in 3.2.

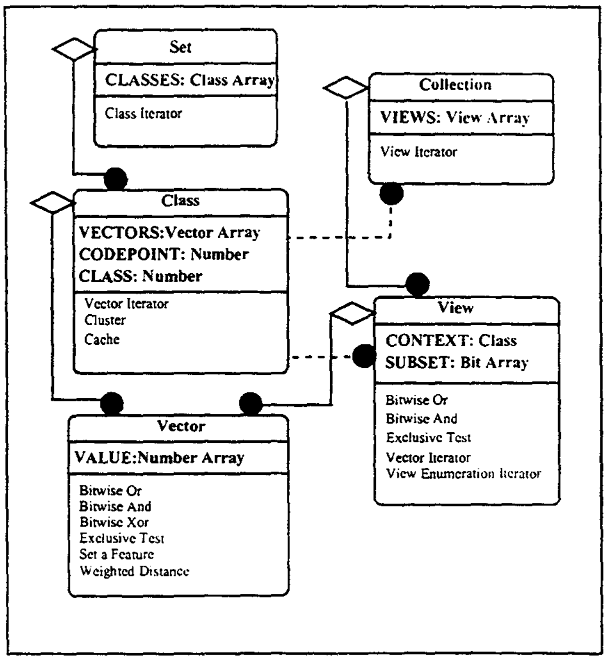


Figure 1: Data Diagram

4.1 Data Design

Figure 1 represents the data structure used within the implementation. The fundamental unit of data is the binary feature "Vector." Binary feature vectors are aggregated into a "Class" which represents a character or one of several representations of a character (i.e. S+). The classes are then aggregated into a "Set" which contains the entire universe of training data. A "View" is an aggregation of vectors within one class. And finally a "Collection" is an aggregation of views which all correspond to the same class. Each of these objects has a set of data attributes and functions use to operate on the data attributes.

The "Set" object has only one basic function which is to iterate through its classes. This function is used by the main routine to evaluate the current feature set against each class. It is also used within the reduction of features to determine if a vector within a class is exclusive against every vector in all other classes (i.e. S-, excluding the current codepoint).

The "Class" object has several basic functions. The first is the vector iteration function which is used mainly to determine if a vector is exclusive against all vectors in a particular class. The second function is used to cluster the vectors within a class down to a manageable number. Since the performance of this algorithm is exponentially related to the number of samples in a class as seen in the performance section, the vectors need to be clustered. This "clustering" is refered to as data reduction throughout the remainder of the paper. The final function is required to manage the data requirements of large data sets. The caching function allows a class to be written and read from a data device.

The "Vector" object is the fundamental object within the data hierarchy. The first few functions are basic operations on the binary feature vectors such as the bitwise or, and, and exclusive or. The "Exclusive" test is a fundamental routine that determines whether one feature vector overlaps another in the given feature space. The "Set a Feature" function is needed to set and clear the bits representing a particular feature within the binary vector. This function along with the exclusiveness test help to determine which features distinguish one vector from another. Finally, the "Weighted Distance" function is used to compare one vector to another given a set of weights or confidence values per feature.

The "View" object is used to contain a set of vectors (e.g. G). The goal is to find those sets of vectors that combine to form a binary vector ($\alpha(G)$) which is exclusive against the rest of the universe. Hence this object is used to group vectors and test their exclusiveness. The first two functions are used to group the vectors into one test vector. The "Exclusive test" is used to help determine a minimal set of views that are exclusive against the universe. The vector iterator is used to count vectors within the view and to test the vectors for various attributes. Finally, the enumeration iterator is used to create collections of views using the M choose N operator. This function is the one that drives the time performance of the algorithm.

The "Collection" object is simply a container for a set of views corresponding to one class (eg. $\Omega(S+,S-)$). It is used to contain minimal sets of views which are exclusive against the universe. Therefore the only operator required is a view iterator which is used to test the views for exclusiveness and whether or not they violate the "minimal set" rule.

4.2 Algorithm

The implementation of the distributed, iterative feature evaluation tool is represented in Figure 2. Given a properly segmented source data set and truth model, the real feature vectors are generated. Those features which best discriminate the source data are selected. The template library generated by the feature reduction is read from DASD and the source data set is processed. The generation of real feature vectors and subsequent feature set reduction are optional and only performed against the training data set. On subsequent source material, processing consists of reading the template library and performing the recognition. Of the four steps the focus of the current investigation is within the "select features" activity. All other activities were available within the existing ICR tool.

Within the select features activity, processing consists of three phases; data initialization, processing of classes, and result gathering. This structure was implemented to support the use of multiple processes against the same set of data as well as the ability to interrupt and restart processing.

Data initialization consists of three activities; "Build binary vectors," "Build S-," and "Start children." These activities are only performed by the parent process. Binarization (I) consists of converting the real feature vectors into binary feature vectors. The resultant vectors contain feature values for the entire feature universe. A mapping is maintained that identifies the current feature set. As each class is binarized it is placed into the set S-. Note that S- actually contains every class. During processing it is necessary to skip the codepoint of the class under consideration (S+) when it is encountered within S-. If the number of samples is larger than a user specified parameter the class is reduced (II) down to the desired number. The reduction that takes place for S- is based on Hamming distance and logical disjunction. The parent then initiates child processes to assist in the processing of the individual classes. The use of children and the number to use can be controlled by the researcher.

The parent and children processes then select classes for processing until each class has been processed using the current feature set. After selecting the class to be processed as S+ the process determines if reduction (II) is required. Reduction is recommended

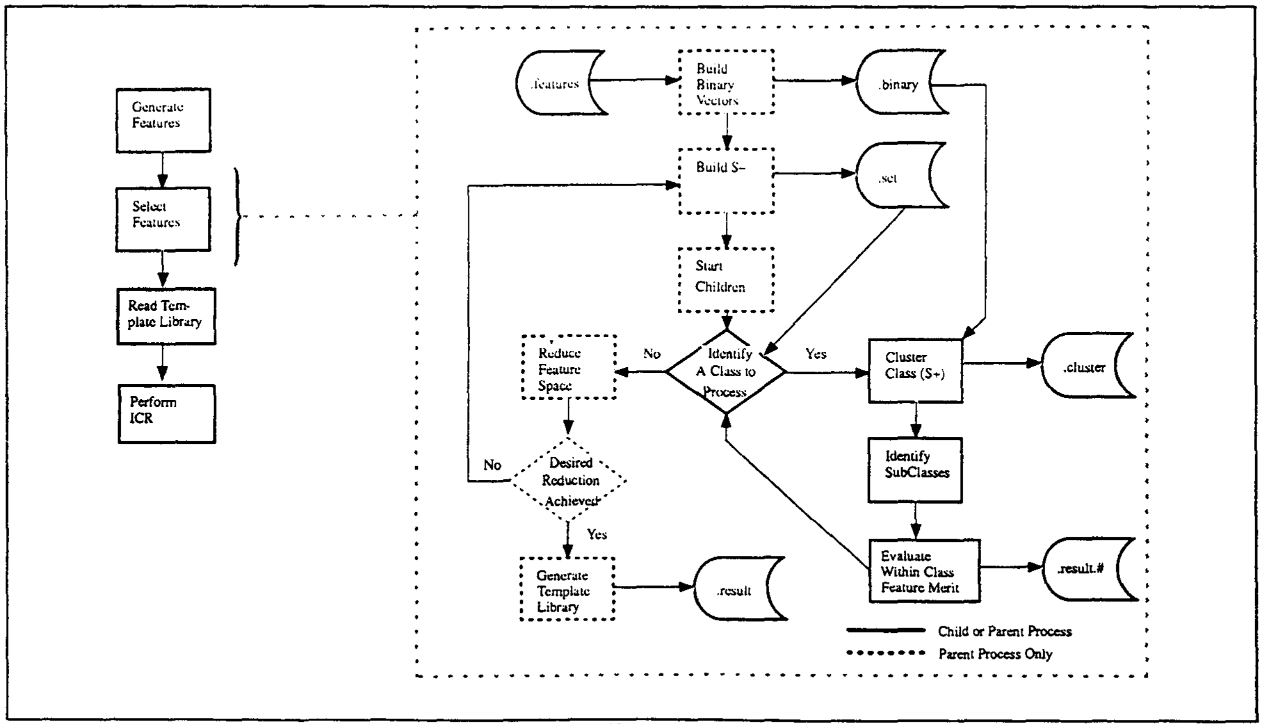


Figure 2: Flow Diagram

if more than 32 samples exist (allows a view to be manipulated using integer operations). The exact limit can be specified by the researcher. Once the class has been prepared for use as S+, maximal exclusive subsets are identified (III) by using a non-recursive version of the base algorithm [1]. For each of these subclasses the current feature set is evaluated (IV) based upon the features ability to discriminate against S-. The binary vector which represents the view has all redundant features cleared. The view, the resulting vector from the view, and the feature evaluations are stored to DASD.

Once each class has been processed against S-, the parent process combines the metrics (V) for each class so that a single metric is available that describes the contribution of each feature in the current feature set. Those features which do not contribute significantly (as configured by the researcher) are discarded. If all the features are significant, those that contribute the least will be removed. The researcher can configure a desired rate of feature removal. The structural indices [1] are then tested to see if the feature evaluation process should terminate (VI). Once the final feature removal has occurred the last set of results files are combined to generate the template library.

4.3 Problem Size Considerations

The largest problem space examined as part of this investigation involved 1,691 classes and 792 features. The scale of the proposed problems was taken into consideration during algorithm and data design. The implementation caches S-. The implementation takes advantage of machine architecture when optimizing sizes of data structures.

A major space concern within the implementation is the memory allocation required for S-. To address this space concern, each class within S- is written to DASD as part of the initialization of S-. A caching mechanism was built into the traversal of S-. A target memory utilization figure for S- is provided as part of the application initialization. If S- exceeds this allocation, a class is selected to be cached to disk. The class will be reloaded from DASD when required.

A second concern is the number of views that can be generated (III), either in Π- or during the saturation of Π-. This is a function of the number of samples within a class. A worst case bound on the length of Π- where N represents the number of samples within the current class is given in (4). Realistically, the worst case expected is more likely to be given by (5).

$$\binom{N}{M} \text{ where } M = \frac{N}{2} \qquad (4)$$

$$\binom{2K}{K} \text{ where } K = \frac{N}{3} \qquad (5)$$

The internal representation of a view requires a single bit for each sample in the class under consideration. The saturation of PI- (III) requires the frequent determination of the overlap (3) between two views. To speed this operation and to limit the total number of generated views a class was reduced to at most 32 samples. This allows a view to be stored within

$$V_1 \cap V_2 \neq \phi \qquad (6)$$

an integer.

Finally, space requirements are driven by the number of buckets used when binarization (I) of the real feature array occurs. Each real feature is converted into a binary representation containing both a set of positive and negative indicators. Each set of indicators is aligned on a byte boundary. For this reason, the optimal number of buckets is eight (number of bits in byte) or fewer. The ninth bucket would result in a doubling of the storage requirement for binary feature vectors. For all of the test results presented five buckets were used.

4.4 Running Time Considerations

Through examination of the implementation logic diagram (Fig. 2), a simple analysis of computation time can be performed. Of interest is the breakdown of processing within the "select features" activity. The core loop within the base algorithm is the processing of a single class against S-. This series of steps must be performed against each class. The "Identify SubClasses" activity represents the most computationally intense step within the entire process. However, the activities required to evaluate the class can be performed independently for each class once S- has been constructed using the current feature set. The implementation of the feature selection algorithm was designed to run on multiple processors simultaneously. A single process is responsible for binarization, generation of S-, reduction of the feature set, and generation of the final template library (**I, II, V, VI**). This parent process can then invoke child processes which merely process a single class against S- (**III, IV**). These child processes may be on the same machine or on separate machines. The coordination of classes between processes and the presentation of results are handled through a shared file system between the parent and all child processes.

The caching mechanism (see 4.3) also directly impacts the processing of a class against S-. The time required to fetch a class within S- from DASD is a significant portion of the total time required to check for exclusiveness against the class. To minimize caching, the set S- is implemented as a linked list of classes. When a vector is tested against S- for exclusiveness, the class which results in failure is moved to the head of the list. When processing a series of views from S+ against S- (III) each vector is likely to have similar characteristics and thus will fail against the same class within S-. This allows for caching of infrequently referenced classes.

Early experiments against the largest problem set indicated that the evaluation of a single class ranged from four minutes to a few hours. Given that there are ~1700 classes and that multiple iterations would be required, the implementation has to be interruptable. The implementation was designed to generate intermediate results for each class as processed. If restarted the current state is examined and work continues. The intermediate results were used to examine peaking phenomena within the feature set reduction model.

Finally, within the "Identify SubClass" activity (III) is a step which requires the enumeration of subset of S+. Given the restriction of S+ to 32 samples, this one piece of logic is bounded by $\binom{32}{16}$ steps with significantly fewer steps expected (see [2] for a complete discussion). Each step requires a test for exclusiveness against S-. Given the number of samples in S-, the implementation enforces a limit on the number of steps allowed. If the number of iterations is considered beyond computational feasibility the algorithm is terminated early. The class is not divided into subclasses and each sample is considered within the final view that composes the set Ω(S+,S-).

4.5 Modifications to Algorithm

The implementation required modification of the base algorithm. Modifications were made to enhance performance, utilize space efficiently, and to exploit existing tools.

Examining the flow diagram (Fig. 2) the activity "Identify SubClass" (III) requires the greatest amount of CPU time. The base algorithm for finding subclasses [2] performs at least one iteration through the recursive procedure ENUMSAT. The class S+ may not contain any subclasses. Predetermination of the existence of subclasses within S+ can completely remove the "Identify SubClass" activity. The existence of subclasses is easy to detect. If a vector generated from the view consisting of each sample in S+ is exclusive against S-, then each sample is exclusive against S-. In this case a single view consisting of each sample will be the only entry within the collection Ω(S+,S-).

The subclass algorithm [2] (III) was improved to reduce the total number of iterations required. Once a subclass divides, each resulting portion needs to be examined to determine if it is sufficient or if further subdivision is required. The same subdivision may be identified many times upon examination of the divisions. This duplication is redundant and in a large problem space requires significant processing time. The algorithm was modified so that it was iterative (vs. recursive). A work list is maintained that corresponds to the original recursive call. Redundant entries are not placed onto the worklist. This reduces significantly the number of traversals of S- required to determine if subsets were exclusive.

The loop within the flow diagram (Fig. 2) from the activity "Build S-" through the decision "Desired Reduction Achieved" reduces the feature universe. The base algorithm drops features (V) from the universe using two rules. First, features are removed if the feature contribution does not exceed a configurable threshold. Then, if no features were dropped in the first step a single feature is selected for removal. The implementation drops a configurable percentage of features on each iteration. This reduces the total number of iterations while sacrificing accuracy within the backward selection model.

The binarization (I) used to convert from a real feature vector to a binary feature vector was modified for the implementation. The modification improves the information content of the binary vector and makes more efficient use of the space required to store the binary vector. The base algorithm binarization method [1] uses the maximum and minimum values for a feature as two of the bucket boundaries used for generating the binary feature. This implies that a bit is required to indicate if a real feature measurement is greater than or equal to the minimum. Similarly a bit is required to indicate less than the maximum. Note that the greater than minimum bit will be true for all values, and that the less than maximum is true for all values which are not exactly equal to the maximum (perhaps only a single vector). To more efficiently utilize the space within the binary vector, the implementation defines (p + 2) thresholds using the base method and then transforms each real sample vector x using (7) (corresponding to (5) of [1]).

Finally, the reduction of S+ and S- (II) were done using an existing clustering tool. This tool uses a hierarchical clustering technique with Ward's method of updating the dissimilarity matrix [9]. The distance function used for the binary feature vectors is the Hamming distance.

5 ICR

The algorithm presented documents the evaluation of the feature set and the calculation of the feature contribution metrics. The ICR makes use of this data to recognize characters within the source data set. The data provided within the template library and the evaluation model are described in section 5.1. Section 5.2 will present an overview of the ICR process as a whole.

5.1 Character Recognition

The results from the feature selection mechanism are used to generate a template library. This template library is then used by the recognition engine to process source samples. The data contained within the results and the comparison process is provided for completeness.

The feature extraction algorithm produces the following data:

$$x \rightarrow \dot{x} = (x_{p+1}^{1}, . . , x_{2}^{1}, . . x_{p+1}^{D}, . . , x_{2}^{D}, x_{p+1}^{-1}, . . , x_{2}^{-1}, . . , x_{p+1}^{-D}, . . , x_{2}^{-D}) \qquad (7)$$

- *Bucket Ranges*: The information required to convert from a real feature value to the binary representation.
- *Feature Map*: Identification of the features that comprise the final selected feature set.
- *Class Data*: There are multiple data elements generated for each class. The complete set is: the codepoint of the class, the binary vectors and the related confidence values representing the maximal exclusive subsets ($\Omega(S+,S-)$) within the class, and a final set of confidence values for the class as a whole. The class wide confidence values are aggregations of the confidence values for each subset.
- *Final Confidence Values*: A contribution metric for each feature in the result set. This metric is the aggregation of the class contribution metrics.

This set of data is used to recognize each character within the source material.

The recognition of a source character is a three step process. The character image is converted into a binary feature vector using the bucket ranges. This binary vector is then compared against each of the template vectors generating a distance measure. Finally, the template which minimizes the distance measure is selected as the correct class for the source character.

The distance measure between a source binary

$$\vec{d}_i = (\vec{S}_i \veebar \vec{T}_i) \wedge \vec{T}_i \qquad (8)$$

vector (S) and a template vector (T) each composed of N features is given by (8) and (9). Each bit in a binary vector represents a rule denoting whether the value for the examined feature is greater or less than a bucket edge. Each feature (i) produces a feature distance measure (8) which represents how far the sample is from the edge of the templates valid range. The final distance is the feature distance multiplied by the

$$D = \vec{d} \; X \; \vec{C} = \sum_{1}^{N} (\vec{C}_i * \vec{d}_i) \qquad (9)$$

confidence value for that feature (C_i) (9). The final distance function (9) may be used with any of the three different confidence vectors: subset, class, and set.

5.2 The ICR Process

Given the feature selection algorithm and a base ICR tool it is now possible to develop and test ICR engines that are customized to the source data set. A small portion of the source data is selected as the training set. Using the base ICR tool this training set is properly segmented and a truth model established for each of the characters. Real feature vectors are generated for the feature universe under examination.

The real feature vectors are converted to binary vectors within the feature selection algorithm. The feature selection algorithm then processes each class to determine the maximal exclusive subsets and the corresponding contribution metrics. The feature selection algorithm continues to reduce the feature universe until a peaking determination is made. Once the final feature set is established the ICR template is generated that corresponds to the input training data.

Once the template is prepared the remaining source data can be processed by the ICR engine. This consists of reading the template library, segmenting the input data, and performing the recognition based upon the minimized distance measures.

The process may be repeated as often as necessary. Examination of alternative feature sets may be performed as new features are proposed by research efforts. The process would be repeated to generate new engines to support additional languages or data sources.

6 Results

Examination of the final results indicates that the algorithm can partially correct for a loss of accuracy due to a broad feature space where overtraining occurs. In the case of a small feature space, the algorithm can significantly reduce the feature set while sacrificing a small percentage in recognition accuracy.

Evaluation of the feature selection method was done by examining two distinct problem spaces. The first was a reasonably sized hand-print English language case. This dataset was used so that the experiments could be performed in a short period of time. The second dataset was a significantly larger machine-print Kanji case. A summary of the findings and final conclusions are presented in the remainder of this section. The details about the evaluation testcases are provided in sections 6.1, 6.2, and 6.3.

After the first iteration, further reduction of the feature space does not significantly degrade the recognition rate. In an environment where feature generation during the recognition of the source document is computationally burdensome, the algorithm provides a reasonable mechanism to reduce the feature universe and reduce recognition time.

The algorithm has a bias towards features which occur later within the feature universe. The effect becomes more dramatic the larger the feature universe. This problem is due to the base algorithm's method of calculating feature contribution metrics. If a subset of features which appear at the end of the feature vector are sufficient to satisfy exclusiveness against S-, then all prior features will evaluate to a zero contribution. This results in early elimination of beneficial features.

The algorithm is well suited to a decomposition of the feature space. Small sets of related features can be evaluated. The subsequent templates and feature contribution metrics can be merged. This assists in correcting the later feature bias while sacrificing the ability to detect redundancy between the selected feature subsets. This decomposition is also well suited to a distributed implementation.

The algorithm is limited in the ability to process a class containing a large number of samples. The approach to reduce the sample space by clustering resulted in a dramatic decrease in the discrimination performance. Similarly, merely selecting a smaller training set results in poor recognition as the algorithm will overtrain to the small set of data. The division of a large class into artificial smaller groups and combining the results is a reasonable alternative. The division of classes reduces the total computation time and allows distribution of the processing while not adversely impacting final recognition results.

The algorithm requires significant computing resources. The largest problem set (~1700 classes and ~700 features) was not solvable with available resources within a reasonable time period. The running time estimate for this problem was approximately 70 machine days. The ability to decompose the problem into smaller processes that can be executed in parallel on a distributed set of processors partially mitigates this problem. The restructured problem was executed in a single day using seven machines.

6.1 Feature Universe

The existing ICR tool was configured with three sets of features for these tests. Each set of features is based upon a normalized image size of 60 by 60 pixels. The first set of 192 features (Slice) is taken from the work of Nakamura and Takahashi [10]. The character is divided into 12 slices in four different look directions: horizontal, vertical, upward diagonal, and downward diagonal.Within each slice two by two pixel patterns are analyzed, categorized, and summed.

A second set of 100 features (Mesh) was generated using a simple mesh pattern. The mesh divides the character into a ten by ten grid. The count of black pixels is then summed within each grid.

The final feature set (LDC) consists of 400 features described in [11] as Local Direction Contributivity. This feature set consists of determining the black run length in all four standard directions at every pixel. These values are then averaged in a box area producing four feature values per box. The same ten by ten grid is used.

6.2 Few Classes, Many Features

The examination of the characteristics of the algorithm was performed against a small set of 36 classes. The data was taken from the ElectroTechnical Laboratory ETL-6 Database. The 36 classes consisted of the numeral and roman sets. 180 samples of each class were selected at random to generate the training data set. Of the remaining samples an additional 180 were selected as the test set.

Tables 1 and 2 provide the recognition accuracy of the feature selection model. Table 1 compares the accuracy of the new algorithm vs. the accuracy of the old recognition engine. The new model reduces the feature set and slightly increases the recognition accuracy when all features are considered. However, the new algorithm corrects only partially for the peaking phenomenon observed in the old model. This can be determined by the loss of accuracy in comparison to the old model when considering only the slice feature set. The results presented for the new model are after two passes through the feature selection algorithm.

Table 1: ICR Results

Testcase	Feature Width	Accy Test	Accy Train
Old - Slice+LDC+Mesh	692	96.72%	95.58%
New - Slice+LDC+Mesh	506	97.19%	99.91%
Old - Slice	192	98.50%	98.04%
New - Slice	113	96.84%	100%

Table 2 compares four possible recognition schemes using the new algorithm. Four pairs of figures are provided. Each pair represents the accuracy against the training data and against the test set. As part of the evaluation of a feature, the feature's contribution metrics are generated at the view level. The view level metrics are then aggregated into class level metrics. The class metrics are then aggregated into contribution metrics across the entire sample space. The ICR engine can use the feature contribution metrics from each of these three levels. The final pair of statistics use the raw distance measure without applying a contribution weight. The implementation considered the slice feature set as a single input dataset. The data presented is for a single pass through the feature selection algorithm. Table 2 indicates that the confidence metrics do not contribute to the recognition accuracy of the final engine. Note that for all subsequent tables and graphs accuracy numbers are presented using the view confidence values unless otherwise noted.

Table 2: ICR Results

Evaluation Model	Accuracy Test	Accuracy Train
View Confidence	87.64%	97.45%
Class Confidence	91.66%	99.97%
Set Confidence	95.04%	98.93%
Equal Confidence	96.12%	99.97%

As can be seen from Figure 3, the algorithm can quickly reduce the feature set with only a slight loss of accuracy. Each point in the graph represents an iteration through the feature selection algorithm. The graph only considers the slice feature set. As can be seen in Table 1, the total accuracy is slightly worse than the baseline results from the original ICR tool (Slice features). Examination of Figure 3 and Table 1 implies that the discriminants selected tended to include noise patterns common to the training samples but not universally present. The small number of samples per class and the large number of features which are greatly variable by noise accentuate this shortcoming.

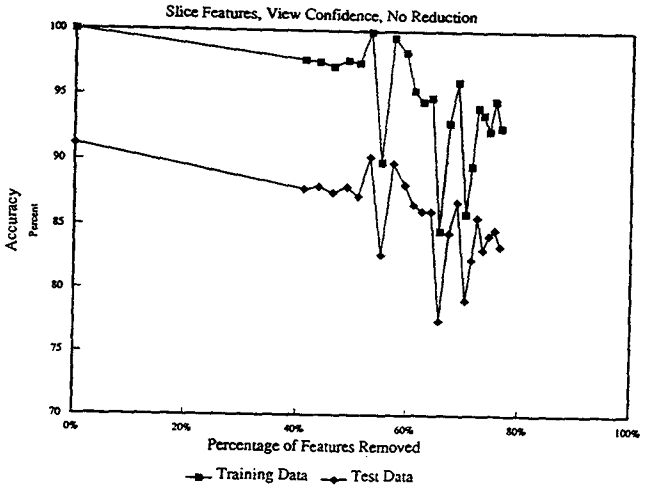


Figure 3: Peaking Phenomena

The algorithm is biased to features which occur at the end of the real vectors. Table 3 depicts the recognition accuracy of the three feature sets considered individually and as a set. The first three data points correspond to the accuracy of the recognition engine when each feature subset is considered by itself. The next three lines examine the entire feature space at one time with the features being presented in different orders within the real feature vector. The implemented algorithm was incorporated into the ICR engine in a manner that would allow the merging of independently evaluated feature sets. The final data point represents the three feature set considered separately and then merged prior to the recognition.

As shown in Table 3 the examination of each feature set individually and then merged results in a better recognition engine. It is important to note, however, that any correlation between individual features from different sets is lost. This information loss will result in a suboptimal feature selection. Table 3 also demonstrates the wide variation in recognition accuracy that can result merely due to feature ordering.

Table 3: Merging of Results, View Confidence

	Feature Width	Accy Test	Accy Train
Slice	113	87.9%	97.5%
Mesh	95	82.6%	96.8%
LDC	298	74.8%	99.9%
Slice, LDC, Mesh	214	87.5%	99.9%
Mesh, LDC, Slice	151	91.0%	100%
Mesh, Slice, LDC	292	74.8%	100%
Slice+Mesh+LDC	506	95.8%	99.9%

The algorithm is severely restricted on the allowable size of the positive sample set. The implementation restricts S+ to 32 samples and will apply a data reduction method to reduce the class down to this number if additional samples are provided. The number of samples can be further restricted by the researcher. Smaller number reduce computational time significantly (see Fig. 8). Each graph indicates the actual data reduction point for the particular trials represented. Since the running time is not linear in relationship to the size of S+, the ability to combine separate testcases into a final recognition algorithm was utilized to mitigate this problem. Figure 4 compares the accuracy of the algorithm when the slice feature set is examined as a single set versus the additive results of the sample space divided artificially into separate subclasses. For each of these iterations the classes within S- and S+ were data reduced (if necessary) down to at most 20 samples. The data was gathered by dividing the 180 samples into groups. The resulting template vectors and confidence values were merged to generated a final template.

Figure 4 indicates that independent runs against large classes are desirable up to the point that data reduction is required. The increase in recognition percentage is significant. However, the reduction of class size decreases the CPU requirements for generation of the feature metrics. The results indicate the seperation of the class into smaller subclasses does not adversely effect the recognition rates, as long as all samples are considered. The significant loss of accuracy occurs only once data reduction (using the clustering tool) takes place. The data reduction scheme needs to be examined to determine if it can be corrected. The current behavoir indicates that the more samples that need to be merged into a single representative sample using our scheme, the worse the final recognition rates

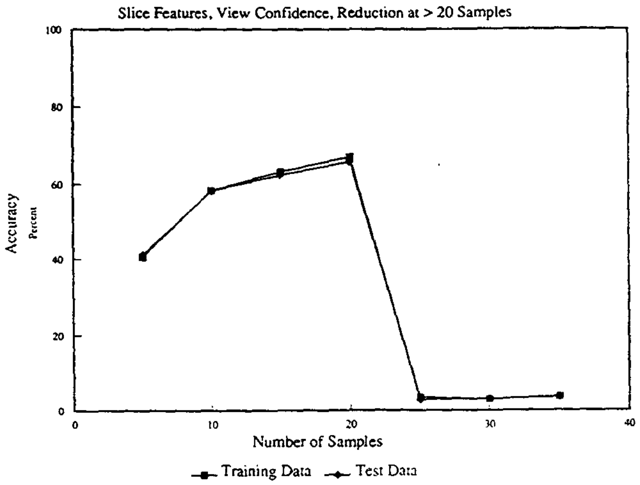


Figure 4: Division of S+

will be for the final ICR engine. It is beleived that a better method of representing the clustered samples will alleviate the problem.

Next, the optimal number of buckets to use when creating binary vectors was examined. Figure 5 presents the recognition accuracy of the slice feature set when the number of division points is modified. When the number of buckets was limited to two buckets the ability to distinguish samples within S+ from S- was greatly degraded. For this data point many classes became indistinguishable using the algorithm. The remaining trials (three through eight) indicate that the number of buckets does not significantly impact recognition performance. The number of buckets did, however, influence the algorithm's ability to reduce the feature space rapidly. With three buckets the first pass resulted in the removal of 39 features, eight buckets removed 83. The anomaly that appears at five buckets suggests that the algorithm is sensitive to the position of the bucket boundaries. A better binarization mechanism is a topic for further research.

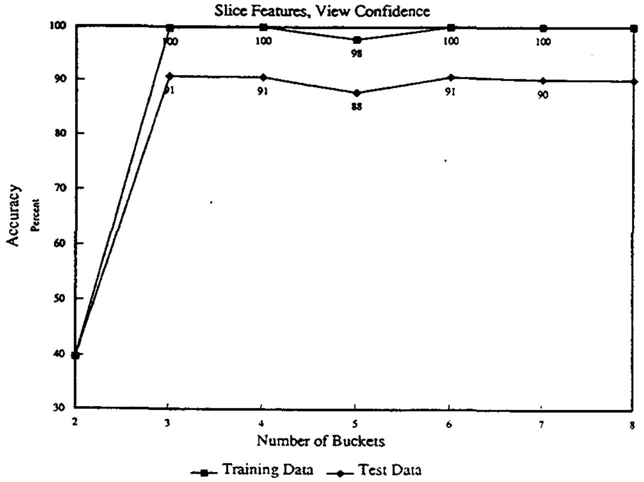
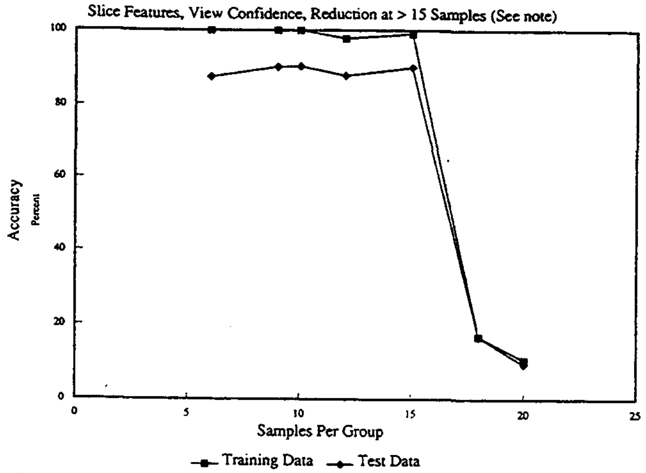


Figure 5: Accuracy vs. Number of Buckets

Finally, Figure 6 represents the accuracy of the algorithm based upon the number of samples within the classes. The first N samples were selected from the available 180 samples. The accuracy of the algorithm was then measured against these class sizes. The results indicate that better results are achieved with larger class sizes. Once again, the data reduction algorithm interferes with recognition results and accounts for the dramatic loss of recognition rates.

6.3 Many Classes, Many Features

The final objective of the study was the improvement of an existing ICR engine that handled a Kanji case. The desired Kanji case consisted of a 1691 classes. Data for this case was taken from the 1994 CEDAR Japanese Character Database from the Center of Excellence for Document Analysis and Recognition at SUNY Buffalo. The samples within the "Book" data set were used for this analysis. The total number of samples in this set is 58,970. Ten percent of each class was reserved to test the final engine. The remaining 90 percent was used as the training data set. Those codepoints that had only one sample were included in both data sets.



Note: Reduction was used when > 15 Samples available but reduced down to 12 Samples

Figure 6: Accuracy vs. Samples

The size of the Kanji case represented the major difficulty in processing the data. Figures 7 and 8 provide some performance metrics that give an indication of the total running time of the algorithm. Figure 7 presents the increase in running time for a single twenty sample class as the number of features examined increases. The three lines represent the minimum, average, and maximum processing time for the class. Note that the minimum time is acutally due to the enhancement to the algorithm with detects classes within which subclasses

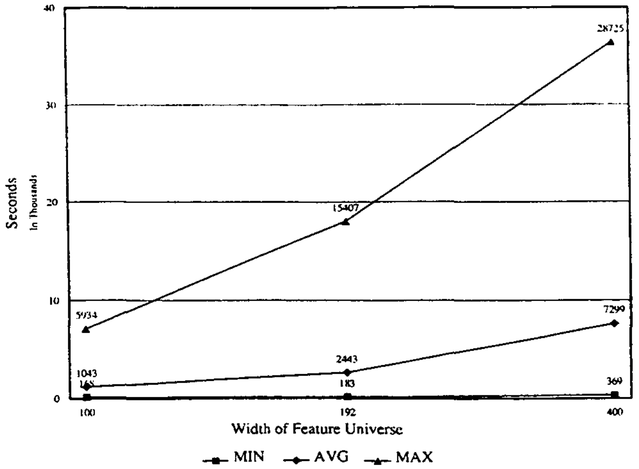


Figure 7: Perfomance vs. Feature Universe

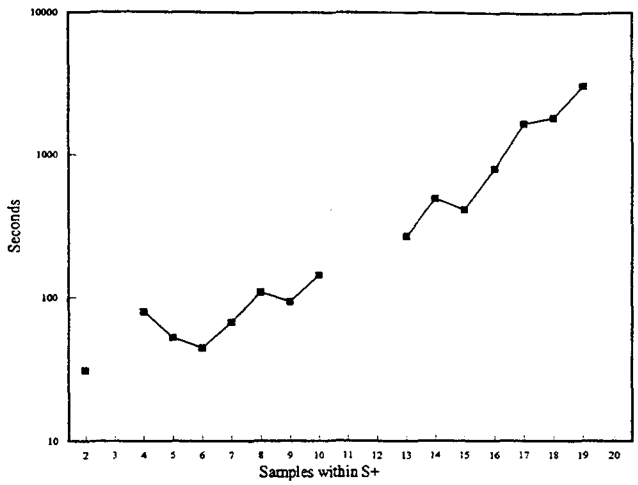


Figure 8: Performance vs Samples

will not be detected. Figure 8 represents the average processing times for a class based upon the number of samples within the class. Note that the scale on the figure is logarithmic.

Given these two performance indicators it was not deemed feasible to perform the evaluation of the entire 692 wide feature space using 32 samples per class. The slice feature set was performed and the class size was limited to 15.

The results of the final Kanji run are presented in Table 3. The feature reduction for this large case is similar to that of the smaller scenario. Similarly, the algorithm resulted in a lower recognition rate. This was expected as the slice feature set alone does not appear to be beyond the peaking point for the old method. The method is valid for the large case. Additional investigation is required to confirm the ability to merge results from different feature sets to improve the engine.

7 Future Direction

The current study indicates that the largest weakness of the base algorithm and implementation is a dependance on feature order. The algorithm is biased towards features presented later in the original real feature vector. The most dramatic case would occur if there were two perfect features for separating a class from all others. Either feature alone is sufficient for perfect discrimination. The algorithm would only detect the later feature and not attribute any significance to the first. It has been proposed that by repeating the within-class feature evaluation and temporarily dropping any features which were significant on earlier passes it might be possible to more accurately distribute feature contribution metrics.

Table 4: Kanji ICR Results

Evaluation Model	Feature Width	Accuracy Test
Old Method - Slice	192	97.7%
New Method - Slice	126	92.5%

Another area to examine is in the area of binarization. The current binarization scheme divides the range of values for a feature into evenly spaced thresholds. These thresholds do not take into account the distribution of the feature values within the range [minimum, maximum]. By setting thresholds that divide the range into sections that actually separate classes, it would be possible to enhance the final evaluation model.

A third area that is being improved is the data reduction algorithm for large classes. A clustering mechanism that does not interfere with the final recognition accuracy will significantly alter the results presented in Figure 4 and Figure 6. An effective data reduction tool should also reduce the influence of noise on the algorithm.

References

[1] M. Kudo and M. Shimbo, Feature Selection Based on the Structural Indices of Categories, *Pattern Recognition* **26** (1993) 891-901

[2] M. Kudo and M. Shimbo, Optimal Subclasses with Dichotomous Variables for Feature Selection and Discrimination, *IEEE Trans. Syst. Man Cybern.* **19** (1989) 1194-1199

[3] S. Mori, C.Y. Suen, and K. Yamamoto, Historical Review of OCR Research and Development, *Proceedings of the IEEE* **80** (1992) 1029-1058

[4] M. Kushnir, K. Abe, and K. Matsumoto, Recognition of Handprinted Hebrew Characters using Features Selected in the Hough Transform Space, *Pattern Recognition* **18** (1985) 103-114

[5] P. Chinnuswamy and S. Krishnamoorthy, Recognition of Handprinted Tamil Characters, *Pattern Recognition* **12** (1980) 141-152

[6] S. Hanaki and T. Yamazaki, On-Line Recognition of Handprinted Kanji Characters, *Pattern Recognition* **12** (1980) 421-429

[7] An Efficient Knowledge-based Stroke Extraction Method for Multi-Font Chinese Characters, *Pattern Recognition* **25** (1992) 1445-1458

[8] Recognition of Handprinted Characters by an Outermost Point Method, *Pattern Recognition* **12** (1980) 229-236

[9] A. Jain, Algorithms for Clustering Data, Prentice Hall (1988)

[10] Y. Nakamura and H. Takahashi, Character Recognition Apparatus, *IBM Technical Disclusure Bulletin* **28** (1986)

[11] Suchenwirth, Guo, Hartmann, Hincha, Krause, and Zhang, Optical Recognition of Chinese Characters 67-68

LITRE: Language Independent Text Recognition Engine

Michal P. Prussak, Laurence E. Bernstein, Ronald A.
Linyard, J. Shane McRoberts, Sheena W. Rice, Bart
Rothwell, Brian C. Sparks

Mitek Systems, Inc.
10070 Carroll Canyon Rd
San Diego, CA 92131
http://www.miteksys.com/browse/mitek
e-mail: mpp@miteksys.com

Abstract: *Text recognition systems have traditionally been developed to take advantage of specific features of the script or language they were developed to recognize. As a result such systems are hard to adapt to recognition of a new language, especially if that language uses a different script. In this paper we present LITRE, a character recognition system that has been developed to be independent of any particular language or script. The algorithms in LITRE are not tailored to any specific language, but they allow for their parameters be set externally for recognition of a specific script. Hence new languages can be added to LITRE without any programming changes. LITRE has been tested on English, Burmese, Lao and Thai and is currently being tested on Vietnamese. We also present how LITRE deals with complex scripts, which place individual characters not only beside each other, but also above or below each other.*

1. Introduction

Traditional text recognition systems have been developed to work on a single language or a single script, in order to take advantage of characteristics of that language or script. As the need to recognize a wider variety of languages increases, the ability to reconfigure a system to a new language becomes very important. We have developed an OCR system, LITRE, that separates a language independent recognition engine from a language dependent configuration specification, so that the language dependent portion can be modified without any programming or re-compilation.

Commercial and academic research in text recognition has centered on languages based on the Latin script, with some work also done on Cyrillic, Hebrew, Arabic and ideographic scripts. The design of LITRE allows for recognition of scripts with a higher level of complexity in character placement. Scripts such as Burmese, Lao and Thai allow the placement of a character or a tone mark above or below another, in addition to placing characters next to each other. Therefore, the segmentation algorithms must be capable of segmenting words into characters horizontally and vertically. The horizontal segmentation produces individual characters, which don't have any neighbors above or below, or stacks of characters - that is, characters that make up a vertical column. Next, the stacks of characters are segmented vertically into individual characters. Without the capability to segment character stacks into characters, it becomes necessary to consider all possible combinations that can make up a character stack, when recognizing a character stack segmented out of a word. In many scripts the number of such combinations may be too great to allow reliable discrimination between character stacks by the recognition engine.

The LITRE system recognizes zones of text, which are assumed to be extracted from the page and pre-processed by a separate system. LITRE segments the text zone into lines, words and glyphs. A glyph is usually a character, but we permit the glyph to be two or more characters combined or a part of a character that can be recognized as a single entity and later reconstructed into individual characters. The segmented glyphs are recognized and assembled into a text stream by a contextual postprocessor. The contextual processor may detect poorly recognized glyphs and request a re-segmentation from the segmentation module. Re-segmented glyphs are recognized and considered for inclusion in the output text stream. The contextual processor may choose glyphs other than the top choice of the recognition module based on bi-gram data for the language. The contextual processor can also use an error modeling file, in which the user can specify hints on dealing with common difficulties in the language.

2. Modules in LITRE

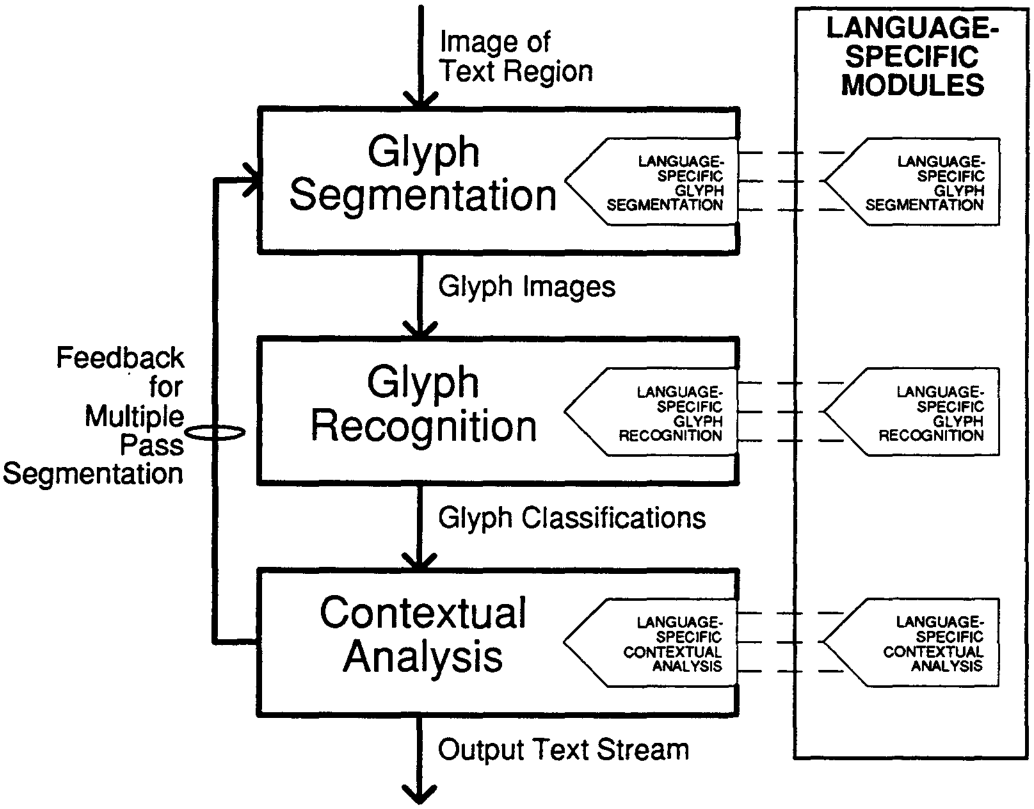


Image of Text Region
LANGUAGE-SPECIFIC MODULES
Glyph Segmentation
LANGUAGE-SPECIFIC GLYPH SEGMENTATION
LANGUAGE-SPECIFIC GLYPH SEGMENTATION
Glyph Images
Feedback for Multiple Pass Segmentation
Glyph Recognition
LANGUAGE-SPECIFIC GLYPH RECOGNITION
LANGUAGE-SPECIFIC GLYPH RECOGNITION
Glyph Classifications
Contextual Analysis
LANGUAGE-SPECIFIC CONTEXTUAL ANALYSIS
LANGUAGE-SPECIFIC CONTEXTUAL ANALYSIS
Output Text Stream

Figure 1. LITRE Character Recognition System

LITRE consists of three main modules: Segmentation, Recognition and Contextual postprocessing. Each module is independent of the language or script to be recognized and can be configured for a specific language through a configuration file. Segmenter's configuration file specifies which algorithms should be used to segment words into glyphs and the parameters for these algorithms as well as parameters for line and word segmentation algorithms. Classifier's configuration file specifies which neural network configuration should be used. Context's configuration file specifies the parameters for its algorithms, an optional bi-gram file for the language and an optional error-modeling file. The error modeling file can be used to specify errors frequently made by the system and to suggest corrections for these errors.

3. Segmentation

The segmentation module used in LITRE performs segmentation of the input image - first into lines, then words and last glyphs. It also allows iterative segmentation - it can provide alternative segmentations for specific areas in the word if requested to do so. The segmentation module assumes that the input image contains only text, and that it has been de-skewed.

The line segmentation algorithm segments the image into horizontal lines of text. As this process is the same for nearly all languages, only the algorithm's parameters can be adjusted in the configuration file. Similarly for word segmentation, it is broadly the same for all languages[1], and its parameters can be adjusted in the configuration file.

The process of character segmentation is far more complex, as it must separate characters that may be touching or broken and can appear above or below each other as well as besides each other. The segmentation module offers several character segmentation algorithms, from which a subset may be selected in the configuration file. Therefore, if a language contains characters that may appear above the main character, the algorithm to segment characters vertically must be selected. The configuration file allows selecting these algorithms as well as specifying the parameters for them.

Segmentation of touching characters presents difficult challenges most of the time, as it is very hard to tell without recognition, whether an image contains two touching narrow characters, or a single wide character. Moreover, segmentation errors are the hardest errors to recover from - recognition of isolated machine printed characters is very reliable, and even when a character is misrecognized, the correct answer can frequently be retrieved by examining the second choice of the recognition engine. Therefore, we found it necessary to provide a mechanism for detecting and recovering from segmentation errors by allowing requests for re-segmentation of characters.

The re-segmentation process is driven by requests from the contextual module. Characters with low recognition confidence are specified for re-segmentation. Then, the segmentation module re-analyzes the area of the image from which the original characters were constructed and provides an alternative segmentation of this area. The re-segmented characters are recognized and considered for replacing the original characters. This process can be carried out several times. The re-segmentation process is intrinsically independent of any language, and the system allows for setting its parameters.

The segmentation module can be configured to under-segment, if under-segmented glyphs will result in a set of glyphs that can be reliably recognized. For example, under-segmentation might lead to construction of glyphs containing two characters. If the set of possible two-character glyphs is limited, the neural network in the recognition module can be trained to recognize such glyphs and the contextual module can then place the characters corresponding to the glyph in the output text stream. Thus when configuring LITRE for a specific language it is not necessary to insist on perfect character segmentation, if under-segmentation can be corrected by recognizing multiple-character glyphs.

[1] Some languages, such as Thai, do not have spaces between words. The segmenter breaks up the lines into "chunks" at the visible spaces. The visible spaces delimit the actual words in most languages, but in some languages the chunks may have to be broken up into words during contextual analysis.

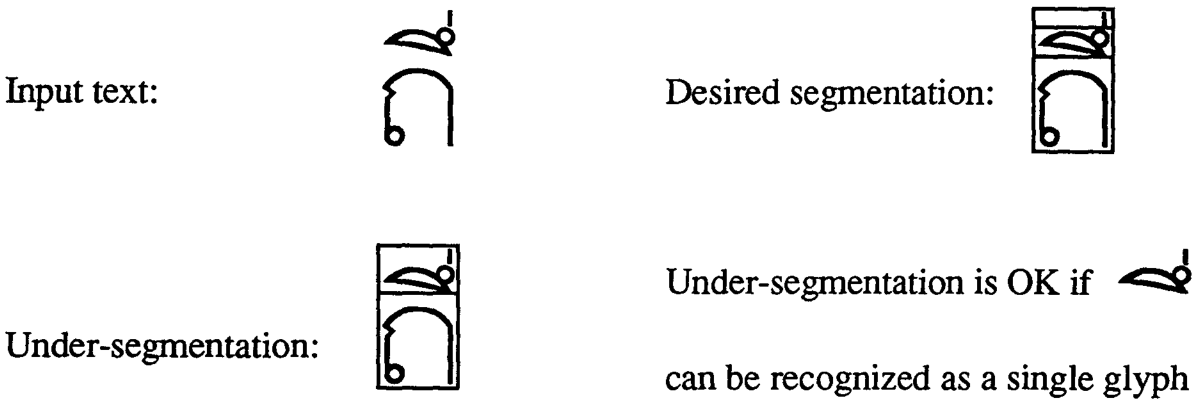

Input text:
Desired segmentation:
Under-segmentation:
Under-segmentation is OK if
can be recognized as a single glyph

Figure 2. Under-segmentation can be OK in some cases

4. Glyph recognition

The glyph recognition module recognizes each glyph found by the segmenter module using a neural network, and it returns a list of its recognition candidates. This module can be configured for any language by training a neural network to recognize the set of glyphs for this language. As mentioned in the segmenter module description, the set of glyphs used to train the network need not be limited to just the individual characters in the language - combinations of two or more characters in a glyph can also be included as glyphs for training the network.

The recognition module supports a hierarchy of neural networks to perform recognition. This can be useful if the recognition module must recognize a large set of glyphs. In this case, the set can be partitioned into several classes of glyphs, according to their similarity, and a separate network can be trained to recognize each class. In addition a top-level network must be trained to determine which partition a glyph belongs to. Then the recognition process first runs the top-level network, which determines what partition the input glyph falls into. Next, the network for that partition can be run, determining the class of the glyph. In this process the individual networks can be smaller and thus run faster. Also, the division of the entire set of glyphs need not be a perfect partition - the classes can overlap, especially for the glyphs that are frequently misrecognized by the top-level network. If the top-level network has trouble placing a certain glyph in its partition, and sometimes places it in another partition, that glyph can be added to the second partition, so that it can be still recognized despite the failure of the top-level network. Typically there might be about 5 such glyphs in a set of 100 glyphs.

For the languages that we have tested, we were able to train neural networks on synthetic data. That is, glyph images were constructed from TrueType font descriptions in various fonts and point sizes, and then artificially degraded using Baird's degradation model.

5. Contextual postprocessing

The contextual postprocessor collects recognized glyphs and constructs an output text stream from them. In this process it applies contextual correction rules and considers glyphs for re-segmentation. If any glyphs are re-segmented it examines whether the re-segmented alternatives are better than original alternatives for inclusion in the output stream.

The first step is construction of the output text stream from recognized glyphs. The glyphs are first combined into stacks, if they overlap vertically, or into words (chunks) if they don't overlap vertically. The stacks are sorted according to the rules for the given language - each language that allows characters to be above or below one another, specifies in which order the characters must appear in a text stream. For example, in Thai the baseline character comes first, then the vowel above, then tone mark above and last, the vowel below baseline. Also at this step, the glyphs representing multiple characters are marked with the sequence of character codes that they contain.

After the sequence of characters for a word (chunk) is constructed, the contextual module looks for glyphs with low recognition confidence and requests a re-segmentation for them. After re-segmented glyphs come in, a DAG (directed acyclic graph) structure is constructed to represent the alternative segmentations. The edges in the DAG are weighted by their combined confidence (segmentation, recognition and bi-gram) and the path with the smallest average weight is found. Because the paths in the DAG may have different lengths, we must look for one with lowest average weight as opposed to lowest absolute weight.

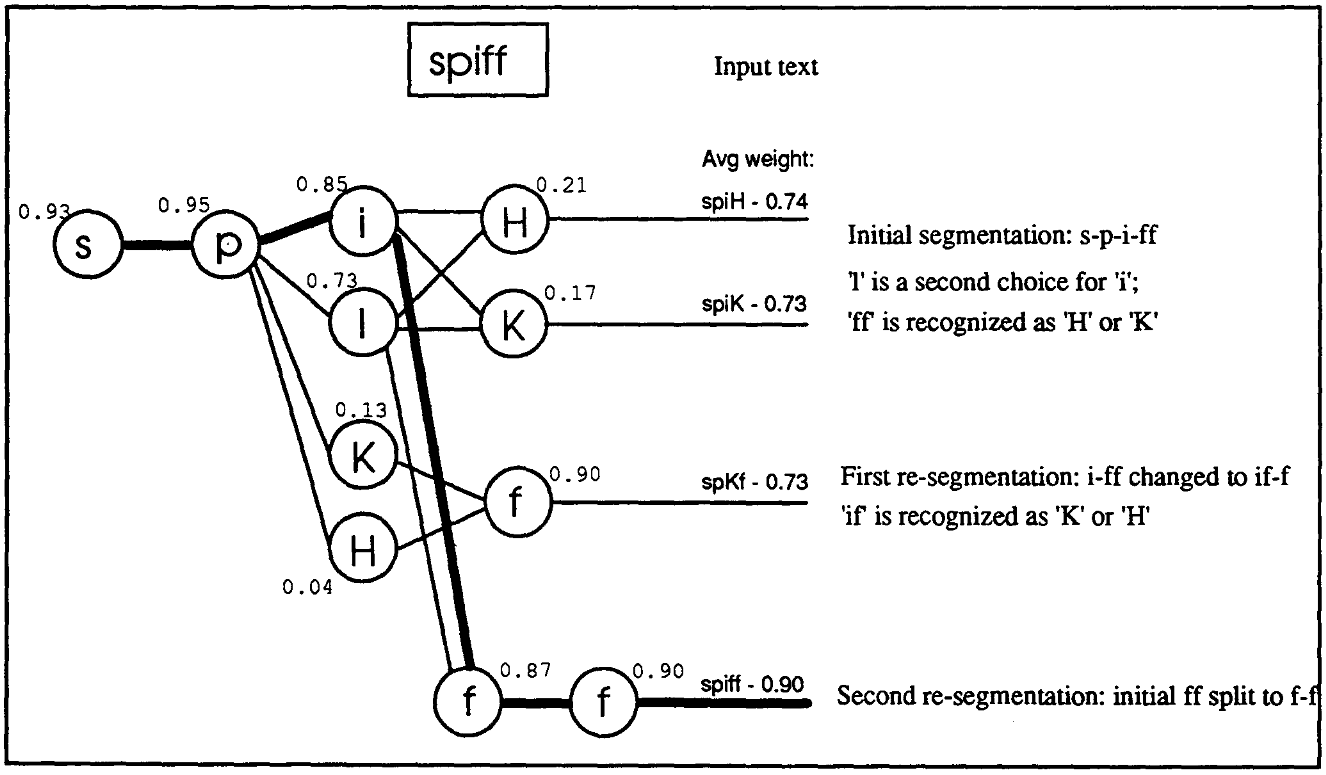

spiff
Input text
Avg weight:
spiH - 0.74
spiK - 0.73
spKf - 0.73
spiff - 0.90
0.93
0.95
0.85
0.73
0.13
0.04
0.21
0.17
0.90
0.87
0.90
s
p
i
l
K
H
H
K
f
f
f
Initial segmentation: s-p-i-ff
'l' is a second choice for 'i';
'ff' is recognized as 'H' or 'K'
First re-segmentation: i-ff changed to if-f
'if' is recognized as 'K' or 'H'
Second re-segmentation: initial ff split to f-f

Figure 3. Best segmentation for a word is the best path in the DAG structure

Lastly, user-specified hints are processed along the best path in the DAG. These hints may take into the account errors that can be frequently made by the system and suggest correction for these errors. For example, the English character 'W' might be cut down the middle into two 'V's. In this case the user can specify that whenever two 'V's are found, they should be combined and recognized as a single character, and if that character's confidence as a 'W' is better than two 'V's, then choose 'W' for the final output. These user specified hints can also be placed in the context of neighboring characters. For example, in order to help avoid the confusion between 'r' 'n' and 'm', a rule could be placed to search for occurrences of the substring 'b' 'o' 'm' 'e' and when it is found, change the 'm' into 'r' 'n'. As the substring 'b' 'o' 'm' 'e' does not occur in any English words, it is reasonable to change it to 'b' 'o' 'r' 'n' 'e', which occurs in several words.

6. Languages tested

LITRE has been configured to recognize four languages, each using a different script - English, Burmese, Thai and Lao. Currently Vietnamese is being configured - it is based on the Latin script, with several additional diacritical marks that can be placed above or below vowels. Because of a very limited set of test data we were unable to thoroughly test the recognition accuracy for these languages. From the data that we had we can anticipate recognition in the range of 98-99% for reasonably clean English documents, approaching 100% for perfectly clean documents (without touching or broken characters). For Thai, Lao and Burmese, the recognition ranges from approximately 93% on fairly clean documents to 99% on perfectly clean documents. By fairly clean documents we mean documents with up to about 25% characters touching. We have not tested LITRE on images with a large number of broken characters. Currently we are preparing to test the recognition of English language on the UW-II database.

Example Burmese text:	မြန်မာတို့တွင် ဝေါယရကြွယ်ဝသော
Example Lao text:	ເຫັນຫນ້າຜູ້ເປັນບິດາທີໄດ້ສນເສັຽຈິວິຕໄປ
Example Thai text:	ผิดสมัยที่ฝรั่งผู้นั้น

Figure 4. Examples of complex script texts

7. Conclusions and future work

We have demonstrated with LITRE that it is possible to recognize different scripts and languages with the same system, by adjusting a set of external parameters. We have also demonstrated the ability to recognize complex scripts such as Thai, Lao and Burmese. LITRE incorporates techniques such as multiple character segmentation algorithms, hierarchical neural networks, iterative segmentation, bi-gram processing and user-defined error correction hints.

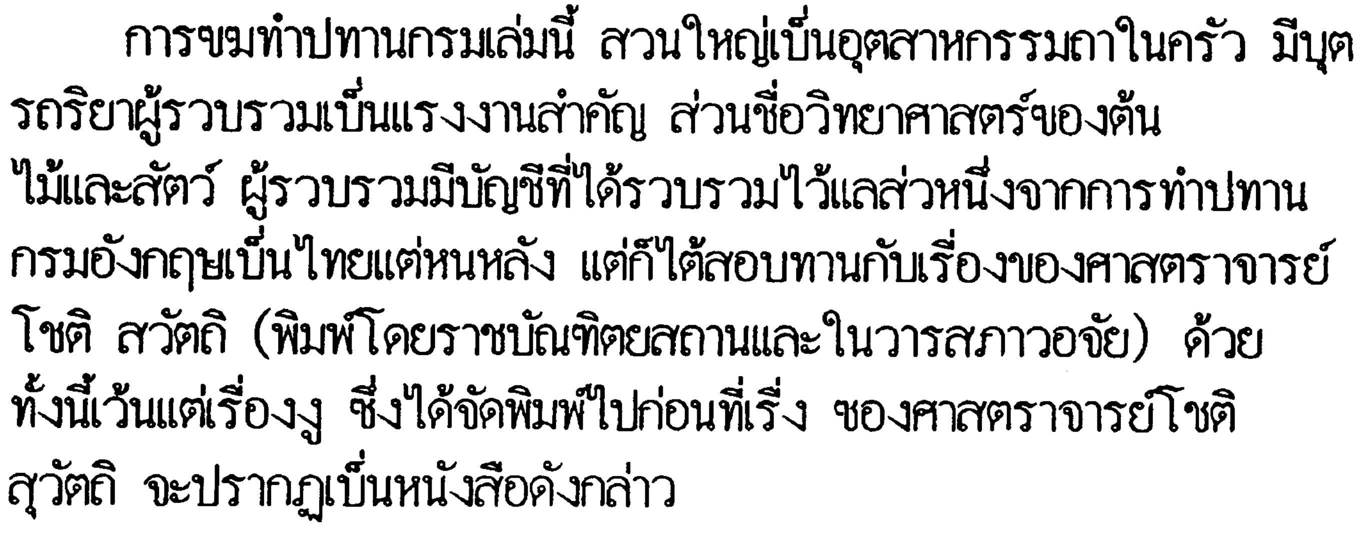
However, LITRE still needs some work to make it a state-of-the-art recognition system. The biggest needs lie in the segmentation system, whose performance is currently the biggest limiting factor on the overall performance. The character segmentation algorithms do not compete for segmentation, but instead are tried in a fixed order until one of them succeeds. Making the selected algorithms compete for segmentation should bring an improvement in character segmentation. Moreover, character segmentation is currently based on connected components, with very limited means of joining two components to form a character. Therefore, broken characters frequently cannot be segmented correctly. Lastly, re-segmentation of characters currently favors splitting characters first rather than weighing the advantage of splitting versus joining. It should be modified so that it can select joining of characters when it determines that this would more likely be a correct segmentation.

The character recognition module is fully developed and only minor improvements could be made to it. One possibility would be to add the ability to use a set of neural networks for a language, or to implement re-recognition. This might be helpful in recognizing specific classes of characters (such as digits or uppercase letters) or specific fonts (such as italic or chancery). However, these improvements would result in small gains in accuracy compared with improvements in the segmentation module.

The contextual postprocessing module has some room for improvement. The capability of postprocessing with a lexicon should be added and the user-specified error modeling should be expanded to allow correcting errors in finding word boundaries. The current implementation of error modeling allows for correcting only recognition and character segmentation errors. In addition, contextual postprocessing currently cannot handle glyphs that make up only parts of characters. Modifying context to handle this would allow the segmenter to separate, for example, a dot from the 'i', the recognition module could recognize these glyphs separately and contextual postprocessor would reconstruct a character 'i' from these two glyphs.

Overall, LITRE is a usable character recognition system capable of recognizing complex scripts and fully tunable with external parameters. Several small improvements mentioned above along with an automatic zoning and preprocessing modules can make LITRE a state of the art document recognition system.

Appendix A. Recognition Examples

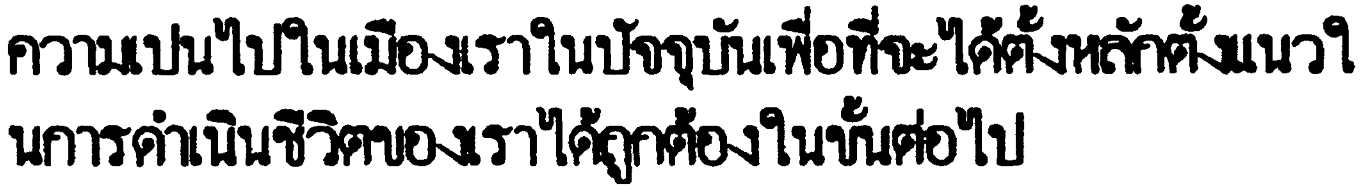


การขมทำปทานกรมเล่มนี้ สวนใหญ่เบิ่นอุตสาหกรรมถาในครัว มีบุต
รถริยาผู้รวบรวมเบิ่นแรงงานสำคัญ ส่วนชื่อวิทยาศาสตร์ของต้น
ไม้และสัตว์ ผู้รวบรวมมีบัญชีที่ได้รวบรวมไว้แลส่วหนึ่งจากการทำปทาน
กรมอังกฤษเบิ่นไทยแต่หนหลัง แต่ก็ไต้สอบทานกับเรื่องของศาสตราจารย์
โชติ สวัตถิ (พิมพ์โดยราชบัณฑิตยสถานและในวารสภาวอจัย) ด้วย
ทั้งนี้เว้นแต่เรื่องงู ซึ่งได้จัดพิมพ์ไปก่อนที่เรื่ง ซองศาสตราจารย์โชติ
สุวัตถิ จะปรากฏเบิ่นหนังสือดังกล่าว

Figure 5. Input Thai image

```
การขมทำปทานกรมเล่มนี้สวนใหฤเบิ่นุตสาหกรรมถาในครัวมีนต
รถริยาผู้รวบรวมเบิ่นแร งงานสำคัฤเส่วนชื่อวิทยาศาล่ตร์ขงต้น
ไม้และสัตว์ผู้รวบรวมมีบัฤเชีที่ได้รวบรวมไว้แลส่วหนีงจากการ ทำปทาน
กรมอังกฤษเบิ่น ไทยแต่หนหลังแต่ก๊ไต้สอบทานกับเรื่องของศาล่ตราจารย์
โชติสวัตถิ(พิมพแดยราชบักเฑิตยสถานและ ในวารสภาวอจัย) ด้วย
็ทงนี้เว้นแต่เริ่ งงูซึงได้จัดพิมพ?1.ปก่อนที่เริ่ งซองศาสตราจารยีเชติ
ฤวัตถิจะปรากจุ1เบิ่นหนังสือดังกล่าว
```

Figure 6. Recognition results for Figure 5

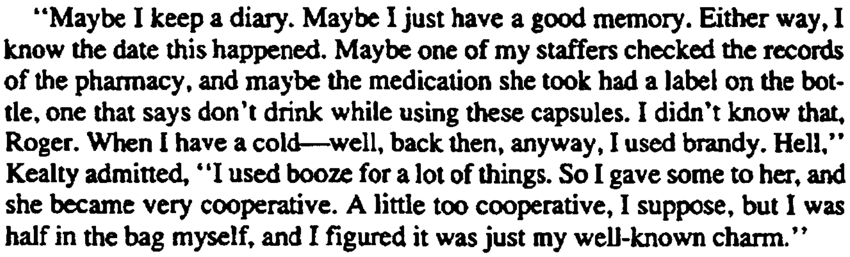


ความเปนไปในเมืองเราในปัจจุบันเพื่อที่จะได้ตั้งหลักตั้งแนวใ
นการดำเนินชีวิตของเราได้ถูกต้องในขั้นต่อไป

Figure 7. Thai input image

```
ความเปนโปในเมืองเราในปจ๊ไขันเพื่อที่ฯโด้ต็งหลัตต็งแนวโ
นการดำเนินชีวิตของเราโด้ลูกค3งในขั็นฝพก้เป
```

Figure 8. Recognition results for Figure 7



"Maybe I keep a diary. Maybe I just have a good memory. Either way, I know the date this happened. Maybe one of my staffers checked the records of the pharmacy, and maybe the medication she took had a label on the bottle, one that says don't drink while using these capsules. I didn't know that, Roger. When I have a cold—well, back then, anyway, I used brandy. Hell," Kealty admitted, "I used booze for a lot of things. So I gave some to her, and she became very cooperative. A little too cooperative, I suppose, but I was half in the bag myself, and I figured it was just my well-known charm."

Figure 9. English input image

```
"Maybe I keep a diary. Maybe I just have a good memory. Either way, l
know the date thi s happened. Maybe one of my staffers checked the records
of the pharmacy, and maybe the medication she took had a label on the bot-
tle, one that says don't drink while using these capsules. I didn't know that,
Roger. When I have a cold--well, back then, anyway, I used brandy. Hell,"
Kealty admitted, "I used booze for a lot of things. So I gave some to her,and
she became very cooperative. A little too cooperative, I suppose, but I was
half in the bag myself, and I figured it was just my well-known charm."
```

Figure 10. Recognition results for Figure 9

Miscellaneous Document Understanding

Document Understanding Interests of the FBI

John D. Hoyt
FBI on detail to ARPA
ARPA DoD/DoJ Joint Program Group
3701 N. Fairfax Drive
Arlington, VA 22203

Document Conversion Architecture

Robert H. Thibadeau
Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

Abstract

This talk will overview the three major initiatives of the Imaging Systems Laboratory in document conversion: The Feature Center Non-Text Conversion Architecture for Technical Publication Analysis, Chinese Document Conversion, and Korean Document Conversion.

Unconstrained Handwritten Text Recognition

Sargur N. Srihari **Venu Govindaraju** **John Favata**
Center of Excellence for Document Analysis and Recognition
State University of New York at Buffalo
Amherst, NY 14228

Abstract

Machine recognition of handwritten text is a three-stage process. The three stages in a text recognition system are: (i) segmentation, *concerns extracting words from a block of text, (ii)* word recognition, *concerns handwritten word recognition algorithms with and without a dictionary and (iii)* post-processing, *concerns using linguistic constraints to improve recognition performance.*

This project is concerned with implementing techniques for handwritten text recognition that will enable automate processing of handwritten text in a variety of documents spanning several commercial applications. The emphasis will be on segmentation and developing methodologies for automated recognition of entire sentences.

1 Introduction

The thrust of the research is twofold. First, to improve upon existing methods of word separation which use spatial cues alone, and secondly, to develop methodologies for recognition of entire sentences. The methodology involves the following subtasks (Figure 1): (i) implementation of a domain independent technique for word separation based on spatial distance metrics alone, (ii) use of word recognition to evaluate various word separation hypotheses, (iii) use of language models and other domain specific knowledge to improve performance. The efficacy of the proposed system will be demonstrated on diverse applications.

Following is a brief summary of the issues involved in the areas research being pursued in the project.

1.1 Segmentation

Most previous work in recognizing text assumes that words are already isolated or that isolation is trivial or that words are written in boxes whose location is known. Few published reports describe methods of separating words in unconstrained handwriting.

Deriving a computational methodology for word separation is quite complex as can be demonstrated by the examples in Figure 2. Humans use several cues to perform word separation: (i) spatial separation between components, (ii) presence of punctuation marks, (iii) presence of upper case characters following a string of lower case characters, (iv) transition between numerals and alpha characters, and (v) actual recognition of words prior to word separation. Among these, the cue of spatial separation is most commonly used.

The commonly adopted computational approach to word separation using spatial distance cues consists of the following steps: (i) determine the connected components in the given line, (ii) compute the distance (or gap) between pairs of adjacent components, (iii) sort the gaps in descending order of magnitude, and (iv) classify the gaps into inter-word gaps and inter-character gaps by choosing a threshold. Gaps greater than the threshold are deemed to be word separation points.

Computing the distance between adjacent components ((ii) above) is an open issue which warrants further research. The objective is to obtain an estimate of the inter-component gaps as perceived by humans. The simplest estimation method computes the distance between bounding boxes of connected components, where the bounding box of a component is defined as the smallest horizontal rectangle enclosing the component. The second method uses *run-lengths* and *euclidean distances* between connected components. A variety of other metrics have been experimented with (section 3.1.1). Each method is prone to fail on certain categories of input lines.

Methods based on spatial cues alone have limited success for two reasons. First, handwriting does not necessarily adhere to the rule of placing larger gaps between words than between characters. Second, the perceived space between components cannot be easily estimated by a 1-dimensional scalar metric. Figure 2 illustrates the necessity for incorporating cues other than spatial ones in order to separate words correctly in different applications.

1.2 Word Recognition

Offline machine recognition of isolated handwritten words, especially cursive script, is a complex problem. The problem is made tractable only when con-

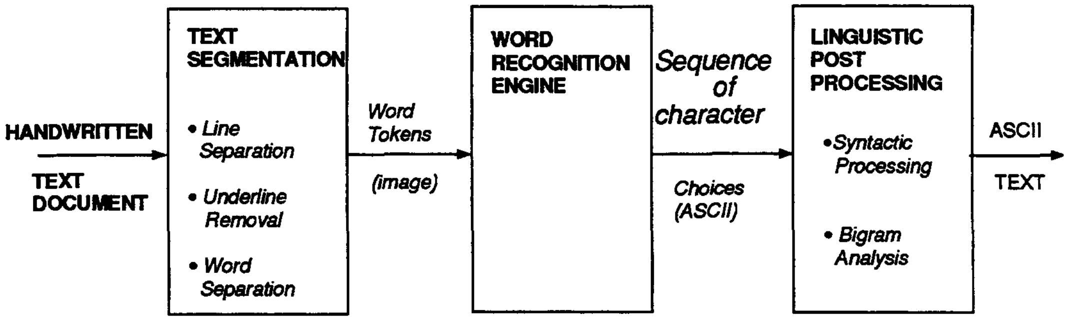


Figure 1: Three-stage text recognition architecture. The output of text segmentation is referred to as a list of 'tokens'. The tokens may or may not correspond to actual words depending on the accuracy of the segmentation procedure. The output of the word recognition engine is a string of character choices corresponding to the 'token'. Postprocessing module uses language models, typically in terms of an appropriate lexicon, to match the character string with a lexicon entry. Final output of a text recognition system is a stream of ASCII text correponding to the document image.

strained by a *lexicon* which is a list of possible words that includes the *truth* or identity of the image as one of its entries. Research in offline handwritten word recognition (HWR) has traditionally focused on relatively small lexicons of the order of 10-100 entries. Recognition rates of 95% and above have been achieved for offline HWR with such small lexicons. For larger lexicons of 1000-2000 entries, both recognition performance and efficiency are significantly lower.

Implementations of algorithms for word recognition are often called *word classifiers* in the literature. An offline word classifier typically accepts as input a raster image of the word and a lexicon of possible words and returns a ranked list of words, sometimes with corresponding confidence values. Classifiers for handwritten text may be classified broadly as being *holistic* or *analytical* in paradigm. In the former, recognition is performed on the word as a whole and there is no attempt to segment or identify individual characters. Analytical classifiers regard words as sequences of smaller, simpler units. The word is segmented implicitly or explicitly into "segments" (also referred to as "pseudo-characters" or "graphemes") and recognition is performed at this intermediate level. The labels assigned to the individual segments (or groups of segments) are then matched against entries of the lexicon. In general, holistic approaches for offline HWR are highly susceptible to changes in word shape, and lack the discriminatory power of analytical approaches.

When multiple classifiers or experts are available, recognition performance on larger lexicons can be improved by combining classifiers at the decision level. The *parallel* combination of classifiers has received considerable attention in recent times. Voting schemes, rank-based schemes such as Borda Count, and theories of evidence combination such as Bayes, Logistic Regression, Fuzzy Integrals and Dempster-Shafer have been applied to combine classifiers at the abstract, rank and measurement levels.

In contrast, the *serial* combination of classifiers has received relatively less attention from the Pattern Recognition(PR) community. In a typical serial combination of two classifiers, a fast classifier – the *reducer* – is used to reduce or filter the lexicon as a preprocessing step for a better and generally slower classifier – the *recognizer*. Serial combinations have been used in the past primarily for reasons of efficiency – either to make a recognition task tractable when the original lexicon is very large, or to improve the throughput of a HWR system [5]

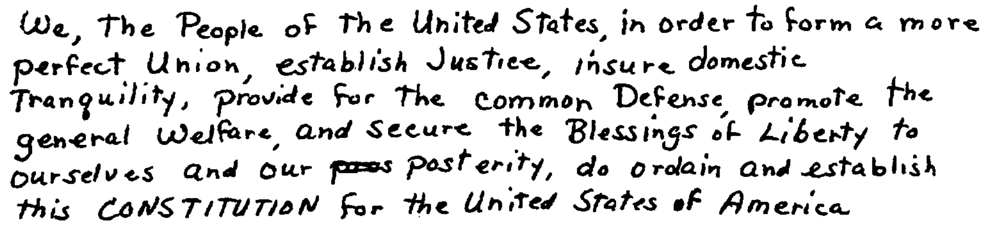
1.3 Report Organization

Section 2, *current status* of the project describes the accomplishments since its commencement. Section 3, *future goals*, discusses the methods we propose for handwritten word separation and the overall methodology for text recognition. Section 4 gives some conclusing remarks.

2 Current Status

Following accomplishments have been made thus far on the project.

1. Initiated an investigation into the problems involved in machine recognition of unconstrained offline handwritten sentences.
2. Developed an overall methodology for the general problem of text recognition which initiated work in several subtasks.
   - line separation and word separation
   - word recognition
   - linguistic analysis
   - scoring metrics
3. Implemented a software system which deals with some of the issues of the overall text recognition task making two assumptions: i) writing is predominantly cursive (as opposed to discrete) and ii) the vocabulary is limited to 2,000 words.



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 ~~pos~~ posterity, do ordain and establish
this CONSTITUTION for the United States of America

Figure 2: Handwritten text sample from the NIST database

4. Initiated the construction of a database of handwritten sentences for training/testing. A database of 100 images have been selected based on the following properties:
   - Images contain a single line of text, or two lines that are well separated.
   - Lines do not have stray markings from the lines above and below it. It is to be noted that the images in the database were extracted from the image of a full page of text. Large handwriting resulted in the individual text lines touching each other.
   - Writing style is predominantly cursive.
   - Words in the image are well separated and are not pathological cases for the word separator.
   - Sentences do not contain digits and punctuation which are currently not handled adequately by the word recognizer

2.1 Word recognition

We performed an experiment involving three word classifiers, two analytical in paradigm and one based on holistic features, and a lexicon of approximately 1700 words (see 3.3). The aims of the experiment were threefold: (i) to study the potential of serial combinations of classifiers to improve recognition performance on large lexicons, (ii) to study the influence of the reduction performed in the first stage on the recognition performance of the combination, and (iii) to study the influence of relative orthogonality of the features used by the reducer and the recognizer on the recognition performance of the combination.

We found that the combination of the two analytical classifiers gave more than a 10% increase in top choice recognition performance over the top choice recognition performances of either of the two classifiers (the classifiers performed at approx. 70% on the lexicon of 1700 words, their serial combination performed at 83%). The serial combination performed better than two popular parallel combination methods - Borda Count and Logistic Regression - and was computationally less expensive. The holistic-analytical serial combination showed promise, but its recognition performance was limited by the lower performance of the holistic classifier.

In conclusion, serial combination of classifiers is a new and effective method for dealing with degraded recognition performance on larger lexicons in real-world word recognition scenarios.

The performance of a serial combination is a function of the reduction and recognition performances of the two classifiers in combination, the extent of first-stage reduction, as well as the relative orthogonality of their features. Given a set of word classifiers, the serial combination of the two individually best classifiers may *not* necessarily be the best possible serial combination. It is a reasonable conjecture that a serial combination of two classifiers with less correlated features, such as a high-performance holistic classifier with an analytical classifier, will perform better than a serial combination of two analytical classifiers. Even with two classifiers having correlated features, substantial gains in recognition performance may be obtained for larger lexicons.

2.2 PENSYS: End-to-End System

Input to the end-to-end system (PENSYS) are "token" images corresponding to different hypotheses generated by the word separation module for a given sentence image. When the sentence is on more than one physical line, word separation hypotheses are generated for each line separately. All token images corresponding to a single page (or scan) are stored in the same subdirectory. Corresponding to each page, there are S sentences, each sentence s may have one or more lines. Each line may have multiple word separation hypotheses, and each hypothesis is comprised of multiple token images. A token image in the most general sense is the part of the line image between two gaps that were adjudged to be inter-word gaps, and as such may contain the image of a word or parts of words.

Given a sentence from a particular scan to process, PENSYS generates all possible word separation hypotheses for the sentence by combining hypotheses for each line in the sentence. Each of these sentence hypotheses is simply a linear ordering of token images corresponding to one possible segmentation of the sentence image. Word recognition is called to recognize each token image and top K choices for each token are compiled into a single file. This file is input to syntactic postprocessing, which reranks the K choices for each token and also computes a new confidence for each word choice based on the initial recognition confidence, word frequency and syntactic category transition probabilities. The new top word choices for each token are taken together to form the sentence result for the hypothesis in question. An overall sentence confidence value is computed and assigned to the sentence result. The sentence results corresponding to different sentence hypotheses are ranked in decreasing order of confidence, and the best result is taken to be the output of the system for the given sentence.

PENSYS can be called as a subroutine from a master script that actually performs the word separation. In its latest form, it provides options for turning syntactic post-processing "on" or "off".

Alternate word separation hypotheses for a given sentence image differ only in the presence or absence of a few (≤ 3) gaps. Therefore hypotheses have several tokens in common. PENSYS assigns a signature (comprised of the start row, start column, number of rows and number of columns) to every token image encountered, and the signature of a new token is compared with the signatures of tokens in previous hypotheses. A table of unique tokens and the corresponding results of word recognition are stored and added to the table. When a token is encountered that has been processed as part of a prior hypotheses, word recognition results are looked up in the table rather than recomputed.

2.2.1 PENSYS subtasks

Following subtasks are performed:

1. *Image cleanup:* Removes small components and stray ascenders and descenders.

2. *Line separation:* Input is a "clean" image that contains a single line or two lines of text. Output is images of the individual lines.

3. *Line merging:* Input is 2 line images generated by line separation, that are part of the same logical sentence. Output is a single image with the second line appended at the end of the first line. This module is not called for images that originally contain a single line of text.

4. *Word separation:* Input is an image of a line. Output is a list of word separation hypotheses (up to 5), where each hypotheses is a list of possible words in the line. The key modules in the program include gap estimation and classification of gaps into word and non-word gaps. For the former module, the Convex Hull metric (see section 3.1.1) is used, and for the latter module, the differences between consecutive gaps in the sorted list of gaps is used.

2.2.2 System performance

An image is said to be correctly recognized if at least one of the ASCII interpretations is fully correct. This implies that every word in the sentence should be correctly recognized. The word separation is said to be correct if all the words are correctly isolated.

Number of images:	117
Corrects:	11
Word separation errors:	15
Word recognition errors:	91

2.2.3 Performance analysis

Word separation did not perform well on noisy images and on images with very few words. The latter case occurred when the tail of a sentence was written on the next line by the user and identified as a separate line by the line separation module. To circumvent such problems, image "cleaning" and line-merging steps were undertaken before word separation module is called. The separation module did not detect punctuation (commas, periods and apostrophes), which were either included as part of adjacent words or marked as separate words. Both these cases resulted in errors in word recognition.

2.2.4 Scoring mechanisms

The evaluation of the system performance raises several issues. Given the truth of a sentence and the system output, how can one measure the performance of the system such that the score computed reflects the accuracy of transcription? A sentence may be thought of as a string of words, and hence standard string distance metrics are applicable. Hamming distance (match/mismatch at each word position) is simple to compute but yields scores that are not intuitive. One intuitive metric is based on LCS, or the longest common subsequence of two sentences. This metric extracts the longest subsequence of words common to the two sentences. Words in the subsequence are ordered left to right in both sentences, but do not have to be contiguous. The score is $\frac{length(LCS)}{length(truth)}$.

Example:

Truth:	he keeps zipping in and out of the house all the time
Output:	he e keeps apparently visit and out of me hours all the himself
LCS:	he keeps and out of all the
Score:	7/12 = 0.58

One drawback of the LCS-based metric is that it ignores the length of the output. Edit distance metric with editing operations of insert, delete and substitute, (with fixed or variable costs assigned to these operations) is another choice.

When all costs are set to the same constant, the edit distance is equivalent to LCS. Complex edit distance functions may be designed by making the costs a function of the words. For example, the cost of inserting or deleting a word is proportional to the length of the word and the cost of substituting one word by another.

PENSCORE script implements the general edit distance, and is hence capable of computing the LCS score as well as more complicated scores. Other string distance metrics such as the proximity metric capture similarities and differences between two strings that generalize edit distance misses. It may be useful, for example, to capture a sense for the number and length of phrases or contiguous chunks of words that are common to the two sentences. To the extent that humans construct the meaning of sentences from the meanings of significant chunks of words (and not from meanings of isolated words), the recognition of chunks of words is an important factor.

To summarize, the scoring system can measure correctness at a purely lexical or syntactic level. Alternately, measures of semantic correctness can also be incorporated into the score. The design of such a scoring system is a topic of future research, and will depend on the definition of correctness that is appropriate to this project and the application scenarios.

2.3 Linguistic Processing

We have tested a syntactic analysis algorithm which ranks sentences based on part-of-speech (POS) tagging and POS bigram frequencies. Early tests of this algorithm have indicated that it has many limitations. It is currently not included in the end-to-end system.

Currently, a Viterbi algorithm re-ranks the ambiguity sets produced by one of the word recognizers (CMWR, see section 3.3). The Viterbi algorithm finds the best path through the ambiguity sets, where each ambiguity set is the result of running a word recognizer on a single word-position in the input sentence.

However, what is required is a likelihood score for each member of the set. A Baum-Welch estimation method is being proposed since it considers all ambiguity sets from the beginning of the sentence to the current position (forward probability) and all ambiguity sets from the current position to the end of the sentence (backward probability).

1.75 million word corpus from Usenet newsgroups, typifying the informal language which occurs in the recognition input, has been collected. An efficient method has been developed that compiles word-bigram and word-trigram statistics based on the corpus.

3 Future Goals

There are tasks proposed in three specific areas: (i) word separation, (ii) sentence recognition, (iii) word recognition, and (iv) evaluation.

3.1 Word Separation

We propose a framework for word separation that (i) uses primarily spatial cues (and domain-specific non-spatial cues when available) to generate a ranked list of *word separation hypotheses* or alternate segmentations of the line and (ii) incorporates word recognition for verification of hypotheses and in rare cases for generating alternate new hypotheses.

Every word separation hypothesis or alternate segmentation of a given line divides the line into line segments or *tokens.* A token t flanked by gaps p and q has one of the eight possible syntactic structures shown in Figure 3. Only one of the eight syntactic structures (shaded in Figure 3) corresponds to a correctly isolated word.

Spatial cues are used in the *Hypothesis Generation* module in order to generate a ranked list of alternate word separation hypotheses with associated confidences.

If the best hypothesis has a confidence exceeding an empirically determined threshold τ_s, the hypothesis is "finalized", and no further processing is deemed necessary. For reasons of efficient processing, it is desirable that the majority of word separation problem instances be solved at this stage. When spatial cues are insufficient to provide a clear "winner", word recognition and linguistic constraints are used to evaluate each hypothesis in the ranked list and suitably modify the associated confidences. The best hypothesis is finalized if its new confidence exceeds a threshold τ_w.

In the rare event that no hypothesis is found satisfactory, a new word separation hypothesis is generated by using word recognition with a sliding window technique.

The input to this module is the binary image of a line of text. The output is a list of hypotheses, where each hypothesis is a possible segmentation of the line into words. Two sub-problems are (i) the estimation of gaps between pairs of adjacent components in the line, and (ii) the generation of word separation hypotheses through the use of the gaps obtained in (i).

3.1.1 Metrics for Gap Estimation

In this section, we address the following question: given a pair of adjacent connected components, what is a good estimate of the gap between them? We begin by briefly describing three methods for gap estimation that have been developed in CEDAR.

The first and most straightforward estimation method computes the horizontal distance between the *bounding boxes* of adjacent components, where

Case	True identity of p	True identity of q	Syntactic structure of token - example
1	True inter-word	True inter-word	<word> PERFECT
			<word><word>+ UNION, ESTABLISH JUSTICE,
2	True inter-word	True inter-char	<prefix> JusTi
			<word>+<prefix> insure domes
3	True inter-char	True inter-word	<suffix> blish
			<suffix><word>+ N ORDER To
4	True inter-char	True inter-char	<middle> quili
			<suffix><word>+<prefix> rovide for t

Figure 3: Syntactic types of token samples contained in word separation hypotheses. There are at least 8 different ways in which a token can be syntactically defined of which only one is correct. The incorrect tokens correspond to over-segmentation as well as under-segmentation.

the bounding box of a component is defined as the smallest rectangle enclosing the component.

The second method, denoted RLE_H2 [2], uses *run-lengths* and *Euclidean distances* between connected components, and two heuristics H1 and H2. The heuristics handle cases where adjacent components do not have sufficient overlap in the x and y direction.

The third technique approximates the gaps between components by the distance between their convex hulls.

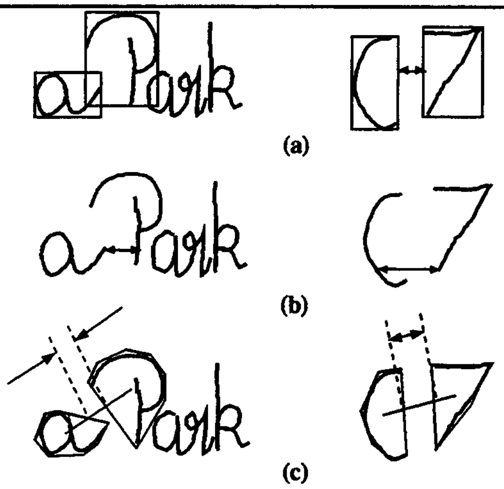


Figure 4: Inter-component gaps generated by (a) Bounding box method (b) Run length based method and (b) Convex hull based method

Figure 4 illustrates the gaps estimated by the above techniques on two pairs of connected components.

Proposed Tasks: Recently, we performed a comparison of the three gap metrics and identified some problems with the convex hull based metric [3]. Since this method was found to be superior to the other two in terms of robustness and performance, our focus will be on improving the performance of this method. The following information about connected components will be used in identifying the particular pairs of components between which gaps are incorrectly estimated: (i) structural details of components, such as height, width, number of horizontal runs in the component, (ii) a ligature-based discrimination of discrete and cursive text, and (iii) shape of convex hulls, and an estimate of how they fit within bounding boxes of components.

After identifying problematic component pairs, we have the choice of simply modifying the convex hull gap or obtaining a different gap estimate using a second gap metric, such as a run-length based metric. The broader goal is to design an intelligent combination of different gap metrics that will result in a good speed/performance tradeoff.

3.1.2 Hypothesizers

The second part of our research focuses on the use of inter-component gaps in isolating the words in a line. The module which performs the above task is referred to as a word separation hypothesizer. We use the term "hypothesizer" to highlight the fact that the output of this module is a list of hypotheses, where each hypothesis consists of a possible segmentation of the line into words.

In the following discussion, we define a *word gap* as a gap that separates adjacent words. Words are then defined as the set of connected components that lie between adjacent word gaps.

We begin by describing two techniques for word separation that have been developed over the past year. Both the techniques operate by sorting the gaps in *decreasing* order of magnitude and by finding a *gap threshold* such that gaps with magnitude above the threshold are classified as word gaps and gaps below the threshold are classified as non-word gaps. A key assumption made by the two methods is that all word gaps are higher than all non-word gaps. The difference between them lies in the manner in which the gap threshold is computed.

The first method, referred to as the MAX method, obtains the difference between adjacent gaps in the sorted list of gaps. The largest difference between successive gaps is assumed to occur between a word gap and non-word gap. Based on this assumption, the pair of adjacent gaps that yields the largest difference is identified, and the gap threshold is set as the larger of the gaps in the above pair. The second method, referred to as the AVG method, sets the gap threshold to be equal to the average inter-component gap. The assumption here is that gaps strictly less than the average gap are non-word gaps.

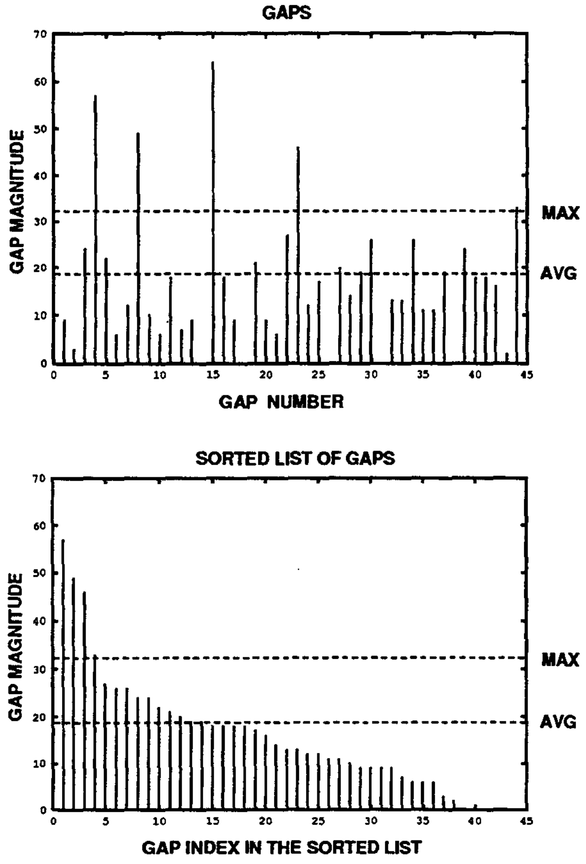


Figure 6: Gaps in a text line and the sorted list of gaps.

Figure 6 shows the gaps between adjacent components in the line shown in Figure 5 and the list of gaps sorted in decreasing order of magnitude. In the top graph, the k-th gap represents the gap between components k and $k+1$ in the line. For example, gap 4 represents the gap between the fourth component, which is 'A' and the fifth component, which is 'M'. Observe that most of the peaks in the graph correspond to the gaps between adjacent words. The lower graph shows the sorted list that is used by the methods to determine the gap threshold. The thresholds obtained by MAX method and AVG method are shown in both the graphs. We note that for the given line, the MAX threshold is much higher than the AVG threshold. As a result, a smaller number of gaps are declared as word gaps, by the MAX method.

In Figure 7, the word separation hypotheses generated by the two methods are shown. The "words" proposed by the methods are each enclosed by a rectangular bounding box. We see that the MAX method considered many word gaps as non-word gaps and hence grouped across true words, while the AVG method performed the reverse operation. Although the above example of a discrete line represents a failure case, the next figure shows a line of cursive text that is perfectly segmented by the above methods.

Proposed Tasks: The first task is the evaluation of the above methods on a large test set consisting of discrete, cursive and mixed text. The purpose of the evaluation is twofold: (i) to obtain an estimate of the current performance, so that it can be used as a reference in future comparisons and (ii) to identify patterns of lines that lead to errors in word separation. In order to perform automatic evaluation, we need to specify a truthing scheme and a corresponding evaluation strategy. One possible truthing scheme is to record the bounding boxes of the words in a line and compare it with the output of the hypothesizers. Word separators can be evaluated based on the expected number of correctly isolated words, and the expected number of lines in which all words are correctly separated. The latter is a very stringent evaluation measure, since a line can be counted as an error even if one of the words is incorrectly isolated.

The second task is to develop new and better methods to classify gaps into word and non-word gaps. To this end, we propose the following new technique, as the first alternative to the existing methods. The technique, which will be referred to as the MODE method, is based on the assumption that the variance between non-word gaps is much smaller than the variance between the word gaps. The assumption is valid for many text lines, including the example shown previously. Based on this assumption, the MODE method will cluster the gaps in the lower end of the sorted list, so as to minimize the variance. The exact measures used to quantify variance, and the parameters involved have to be ascertained, prior to implementation.

The third task is to define a "confidence measure" for the word separation hypotheses. Since confidence measures of individual modules play a significant role in the overall system, it is advantageous that such a measure be computed by the word separation module as well. The task of assigning confidences to word break hypotheses is decidedly non-trivial, and

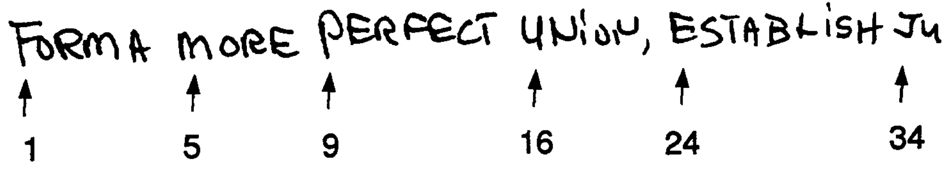


Figure 5: An example of a line containing many discrete characters.

merits careful analysis and research.

Statistics collected from training data are used extensively in word recognition and character recognition, but not in word separation. The main reason for the omission of training statistics is that we have not yet identified the exact parameters that are relevant to the problem. In the course of our research, we hope to identify these parameters and use them in word separation.

3.1.3 Recognition driven separation

Although the notion of using recognition confidences to drive segmentation is quite common in segmenting a word into its constituent characters, its natural extension to the segmentation of a line into words has not been explored, as yet, by the research community. It is our thesis that such a control structure can greatly enhance the performance of text segmentation.

The starting point for this approach is a set of gaps that is assumed to include all the true word gaps in the line. Such a set may be obtained from a hypothesizer such as AVG that is inclined towards over-segmentation (Figure 7b). Alternately, such a set may be obtained as the union of all the inter-token gaps in the generated hypotheses.

The set of gaps may be partitioned into two subsets: *anchor* and *non-anchor* (Figure 8). Anchor gaps are those gaps which are large and the probability of the gap being a word break approaches 1.0. For example, the gaps common to the different hypotheses may be considered anchor gaps, and the rest, non-anchor.

Figure 8a shows the general case where there is a mixture of anchor and non-anchor gaps. An example of the worst case - no anchor gaps - is shown in Figure 8b.

The proposed solution is to consider a sliding window approach to determine word boundaries. If we sequentially group components of the line, from left to right, and perform a recognition cycle on each grouping, we can build a graph of possible word boundaries for the line. This sliding window algorithm and the resulting word recognition results will produce a graph of possible segmentations of the line. Linguistic analysis can be used to determine the best path in the graph which would correspond to the most likely segmentation of the line. Rather than consider all paths which could be computationally expensive, we can take advantage of any anchor gaps in the line to reduce the number of paths that must be searched in the graph.

3.1.4 Role of linguistic processing

The role of linguistic processing is significant when dealing with specific applications, since richer language models can be constructed when the context is known. For example, legal amounts on bank checks are composed of words from a fixed lexicon of some forty words.

At the character level, such a small lexicon is devoid of many letter to letter transitions. It is also likely that character statistics (learned from corpora) can give useful information about word-endings and word-beginnings (*e.g.*, the character sequence "-i-n-g" signals a word-ending with high probability).

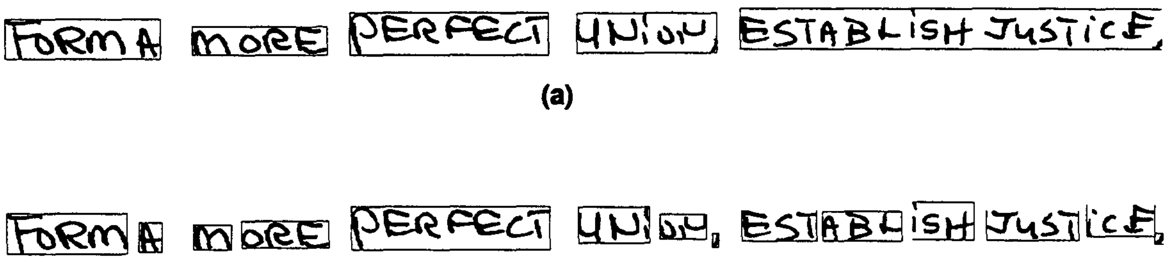
At the word level, similar transition statistics can be used. One such notion referred to in literature is that of *collocations*. Collocations are word patterns that occur frequently in language; intuitively, if word A is present, then with a high probability, word B is also present in its immediate neighborhood. Collocations are categorized based on (i) the strength of their association (mutual information score, *mis*[1]) and (ii) the mean and variance of the separation between them.

At this point we are considering only fixed collocations such as rigid noun phrases (*e.g.*, "computer scientist", "letter of intent"), and lexico-semantic collocations such as verb–particle pairs (*e.g.*, "give up"). The concept of collocations can be used to advantage in application domains where the handwritten text is predominantly phrases of a few content words, as in the case of data fields in census forms. Segmentation hypothesis which result in high collocation score are favored.

On the other hand, in the restricted domain of bank checks, segmentation hypothesis which result in minimal collocation score can be rejected. For instance, 'twenty twenty', 'one five', etc., are not plausible and hence alternate word separation hypothesis should be chosen. It is to be noted, that such pairs would pass the word recognition test but would fail a test based on such linguistic constraints.

Further, the syntactic structure of the input is also severely constrained in legal amounts on checks. For instance, it is known that the words 'dollars' and 'cents' occur only once (if at all) in the text, and if both appear, 'dollar' always precedes 'cents'.

[1] $mis(A,B) = log_2 \frac{P(A,B)}{P(A)P(B)}$


FORM A MORE PERFECT UNION ESTABLISH JUSTICE,
FORM A MORE PERFECT UNION, ESTABLISH JUSTICE,

(a)

(b)

Figure 7: The hypotheses generated by (a) MAX method and (b) AVG method.

3.1.5 Robustness

Non-spatial cues such as punctuation and average letter width will be used. Robustness of the method will be improved to handle the following kinds of input:

- discrete style of writing, as opposed to predominantly cursive
- sentences with noisy components (eg. underlines, stray ascenders, streaks etc).
- over-fragmentation of components, which result in a large number of inter- component gaps.
- sentences where the patron did indeed allocate more space for one or more characters than words.

3.2 Sentence recognition methodology

Word recognition and linguistic analysis verify word segmentation in a hierarchical fashion. Top N word segmentation hypotheses are considered where each hypothesis is a sequence of connected components. The hypotheses are ranked according to word gaps generated by a distance metric. The word gap information, along with word recognition and linguistic processing, is used to verify the word segmentation. About 50% of lines encountered in text are reasonably well behaved, and choosing top 5 hypotheses allows for swift and accurate segmentation/verification of the word breaks.

3.3 Word recognition

We propose to develop serial combinations of classifiers to improve both speed as well as recognition performance [5]. Hybrid combinations involve both serial and parallel combination, and promise to improve recognition performance considerably.

Recognizers [6,7] will be enhanced to handle embedded punctuation such as hyphens and percentage symbols, typographical errors, words not in the lexicon, numerals and special symbols Word recognition algorithms need to improve to yet higher levels of performance.

3.4 Evaluation

Develop semantic measures to reflect the understandability rather than correctness in the strict sense of transcription. Edit distance based metrics for evaluation of recognition at the sentence level need to be developed.

Confidence values must be returned along with recognition results of a sentence. The confidence value should take into account the reliability of line separation, word separation, word recognition and syntactic postprocessing. Performance of the system may then be evaluated at different reject rates and a case may be made for a human-assisted (or semi- automated) system for handwritten text.

4 Conclusion

We have implemented an end-to-end system (PENSYS) which takes as input a complete sentence. The system performs word segmentation and word recognition. A live demonstration of the system is underway.

References

[1] V. Govindaraju and R.K. Srihari and S.N. Srihari, Handwritten text recognition, *Document Analysis and Systems* A. Lawrence Spitz & Andreas Dengel (editors), *Series in Machine Perception Artificial Intelligence*, Vol. 14, 1995.

[2] G. Seni and E. Cohen, External word segmentation of off-line handwritten text lines, *PR*, Vol. 27, No. 1, (1994) 41-52.

[3] U. Mahadevan and D.B. Benson, Gap metrics for word separation in handwritten lines, *ICDAR-95*, Montreal, Canada, (1995) 124–127.

[4] R.K. Srihari, Use of lexical and syntactic techniques in recognizing handwritten text, *ARPA Workshop on Human Language Technologies*, Princeton, NJ, (1994) 403–407.

[5] S. Madhvanath and V. Govindaraju, Serial classifier combination for handwritten word recognition, *ICDAR-95*, Montreal, Canada, (1995), 911–915.

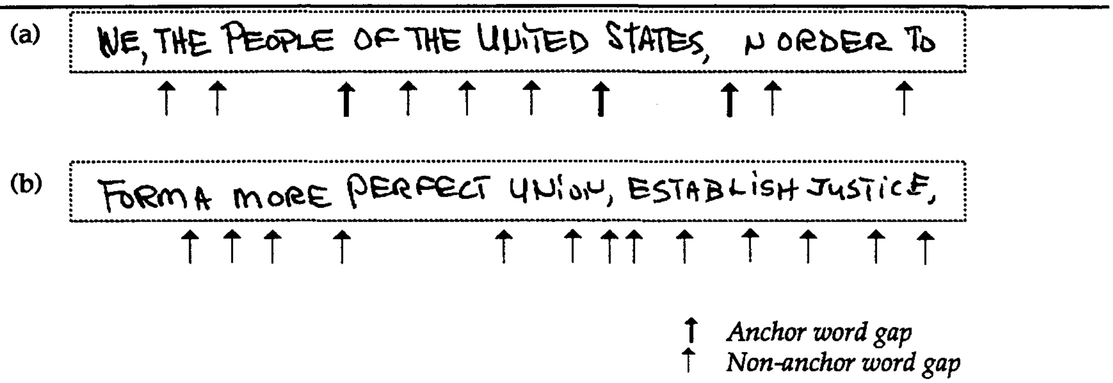

(a)
WE, THE PEOPLE OF THE UNITED STATES, N ORDER TO
(b)
FORM A MORE PERFECT UNION, ESTABLISH JUSTICE,
Anchor word gap
Non-anchor word gap

Figure 8: The set of potential word gaps may be classified as anchor and non-anchor gaps. The general case (a) presents a mixture of anchor and non-anchor gaps. In the worst case (b), all gaps are non-anchor gaps.

[6] J.T. Favata and S.N. Srihari, Off-line recognition of handwritten cursive words, *SPIE symposium on electronic imaging science and technology*, San Jose, CA, (1992).

[7] G. Kim and V. Govindaraju, Handwritten Word Recognition for Real-Time Applications, *ICDAR-95*, Montreal, Canada, (1995) 24–27.

Japanese Optical Character Recognition

Sargur N. Srihari Geetha Srikantan
Center of Excellence for Document Analysis and Recognition
State University of New York at Buffalo
202 UB Commons, Amherst NY 14228-2567

Abstract

Many Japanese Optical Character Readers (OCRs) have been developed to meet specific needs such as reading bank checks, forms and office documents. Research at CEDAR is oriented towards developing a general-purpose Japanese document recognition system. A fully automatic Japanese character recognition engine with omnifont capability has been developed. The input to the system consists of scanned Japanese document images (either 400 *or* 200 *ppi). Automatic preprocessing deskews the image contents, and lifts text regions which are decomposed into text lines and characters, and recognized. The final output of the system is a ranked list of classifier decisions for each character.*

We also describe the new Japanese character image database developed for system training, at CEDAR. This database consists of approximately 180,000 *labeled character images from over* 3000 *categories, extracted from diverse document images. Results of our system on this machine-print Japanese document and character database are also presented.*

1 Introduction

The goal of Japanese optical character recognition research at our research center is to develop a highly accurate preprocessing, segmentation and recognition methods for hand-print characters, as well as for machine-print characters in diverse quality Japanese documents. Although many character readers are commercially available, intensive research is still being conducted to improve accuracy and data throughput.

A brief survey of research in Japanese OCR is presented in Section 2. The remainder of this report gives an overview of the research topics addressed in the Japanese OCR project at CEDAR. Section 3 discusses a new Japanese document and optical character image database developed at CEDAR. Section 4 describes a Japanese OCR system for omnifont machine-printed text. Techniques for document image preprocessing, page segmentation, text segmentation, and character recognition are presented. We also present results of these system modules based on the new CEDAR database. Finally, future directions of this project are outlined.

2 Background

Rigorous effort in Japanese optical character recognition research is propelled by the need to automate the recognition of documents. This need is pre-eminent in banking, insurance, government and postal services. As in systems for English document recognition, a Japanese document recognition system has three major components: layout analysis, character segmentation/recognition, and post-processing. Layout analysis techniques for Japanese documents [1, 2] and multi-lingual documents [3, 4] have been developed. Given an image of a document page, layout analysis will locate text blocks and further segment them into text lines. In Japanese documents, text lines can be either vertically or horizontally oriented, hence layout analysis must determine text orientation also. Further, as word boundaries are not easily distinguishable in Japanese text images, typically characters are directly segmented from the text. Information from connected-component analysis and projection profile analysis can be used to assist character segmentation[2, 5–7]. Feedback from a character recognizer can also be used to resolve ambiguities and correct errors in segmentation.

The Japanese character set is a combination of several scripts: Kanji (Chinese characters), Kana (Japanese consonant and syllabary characters), Roman and Arabic numerals. Kanji are Chinese ideographs which were introduced into Japanese, refer [8, 9]. In the electronic standard, Japanese Information System (JIS) X 0208-1990, refer [10, 11], there are 6,879 characters. Among them, 6,355 are Kanji, divided into two distinct sections – JIS Level I and Level II; where Level I spans 2,965 of the most frequently used Kanji characters. Recognition methods can be broadly classified into two paradigms: *structural analysis* and *pattern matching*.

In the structural analysis methods, feature extraction is carried out from the viewpoint of character pattern primitives. Many character features have been devised to satisfy design criteria of the engineer. It is now commonly believed that local-geometrical-contour features are most useful, and that they are naturally implemented in decision trees, finite state automaton, relaxation matching, and other classification methodologies [12]. These methods have been applied to the recognition of hand-print characters, other than Chinese ideographs, or Japanese Kanji.

In the pattern matching methods, global and uniform features are used to simplify the recognition scheme[12]. The feature extraction methods studied in this field can be categorized as based on 1) orthogonal expansions, 2) stroke distribution, 3) stroke analysis, and 4) background feature distribution. Karhunen-Loeve expansion has been used very successfully for both machine-print and hand-print Kanji recognition[13]. Stroke distribution methods, such as the local direction contribution and the stroke density, are very useful[2]. Others such as, cellular features, surrounding area features, complexity index, histogram, and gradient features, are also useful, refer [13, 14].

Similarity or distance measures have been proposed to achieve more reliable and accurate performance in the discrimination process. The combination of stroke direction features and the multiple similarity method[15] or the modified quadratic discriminant functions [16] have been applied to the recognition of hand-print Kanji, and omni-font machine-print characters.

Neural network models have been widely applied to character recognition. However, most of these networks have been used to recognize a small number of classes, such as alphabets, numerals and Kana. It is difficult to extend these network models to Kanji recognition, as there are over 3000 Kanji classes. For Kanji recognition, it is desirable to have neural network training rules that result in fast convergence. Further, the architecture of such networks would have to be designed such that the computational requirements are feasible. A large scale neural network model, known as "multiple modified learning vector quantizer neural network", has been developed to achieve high performance on machine-print Kanji recognition[17].

Kanji characters are complex in their stroke structure and several characters share structural similarities. Large dimensional feature descriptors are typically needed to discriminate between these character classes. Correspondingly computation time requirements can be high, when these features are used. Efficient multi-stage recognition methods have been developed [2], where coarse or approximate classification stages are followed by finer and more accurate classification stage. The key to the success of the multi-stage recognition method is the reliability of the coarse classification stage, that is, the correct class must be present in the choices output from the coarse classifiers. Classification is error-prone when the input pattern is degraded.

Postprocessing is typically intended to improve accuracy by detection and correction of OCR errors. Linguistic knowledge sources such as dictionary and transition probability between characters, and domain knowledge are exploited in postprocessing [18]. Contextual information, such as character n-gram, dictionary, semantic knowledge and domain knowledge have been applied [19, 20, 18], to detect and correct errors. Linguistic error correction disambiguates recognition results, particularly in identifying a correct Kanji character among candidates which are structurally similar.

Hence, the major challenges in developing a Japanese OCR system are as follows [13]:

- Large variety of print layouts.
- Vertical and horizontal alignment of text.
- Large number of character categories.
- Structural complexity of each character pattern.
- Existence of character patterns that have similar shape and structure.
- Wide variety of character shapes due to different typeface and image quality in machine-print documents.

The research goals in Japanese OCR are:

- Realization of ultra high accuracy recognition for machine-printed characters.
- Considerable recognition rate improvement for low quality handwritten characters.
- Development of a unified recognition method suitable for multi-lingual character patterns, such as Roman, Greek, and Chinese language symbols.

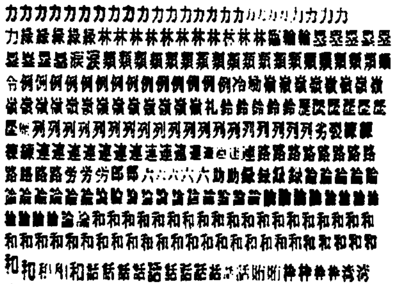


Figure 1: Samples of character images from CEDAR dataset

3 CEDAR JOCR Database

One of the main issues dealt with in this project is the creation of a large database of Japanese document images to overcome a lack of representative data for system training. At CEDAR, we have created a Japanese character image database, which will be available on CD-ROM shortly. This database has two components - (a) a character image database for recognizer development and (b) document images for development of document analyzer and segmenter modules. Japanese documents were digitized at 400 ppi on a flatbed scanner. These documents are from varied sources - facsimiles, photocopies, newspapers, books, journals and magazines - and span diverse document layouts and print qualities. A total of 264 pages have been scanned into the document database. Characters extracted from these images have been tagged with the truth value in the JIS code. Approximately 180,000 truthed character images are available for recognizer development in this database (see Figure 1 for samples of character images from the database). The data has been partitioned into train and test sets. Both the document and character databases along with accessing software and documentation are available on CD-ROM. In addition to this CEDAR CD-ROM, ETL and the University of Washington have made available databases, for OCR system development.

4 The Design of a Japanese OCR System

We describe now a Japanese OCR system for omnifont machine-printed text. This system is designed to lift text automatically from scanned documents and segment text into character units. The main challenges are the wide variations in data quality (facsimile, photocopy, newspaper etc.), low scan resolution (200ppi), large character set (JIS Level 1 & 0) and the variety of font and point sizes. Further, the system is expected to perform without knowledge of context. The system comprises of a document analyzer, page and text segmenters as well as character recognizers and postprocessors. A user interface was also developed for this system. The preprocessing, segmentation and recognition modules are described in detail in the following subsections, along with results based on the CEDAR Japanese document and character image database.

4.1 Document Analyzer

This module is designed to detect and correct skew in document images. A fast directional profile analysis is used in the detection of skew angles between ±30. A simple affine transformation is used for skew correction. The document analyzer is required to perform fast skew detection on images containing graphics, tables and figures as well as text. An outline of the skew detection algorithm follows, refer [21] for a complete description.

4.1.1 Skew Detection

This algorithm is an extension of a technique by Ishitani[22]. The document image is divided into local regions. In each region a directional profile analysis is performed to detect skew. Overall document skew is estimated as a consensus among the local region estimates. Local regions of the entire document can consist of text, graphics and figures, see Figure 2. This division into multiple regions increases the text hit rate, a factor that influences further processing.

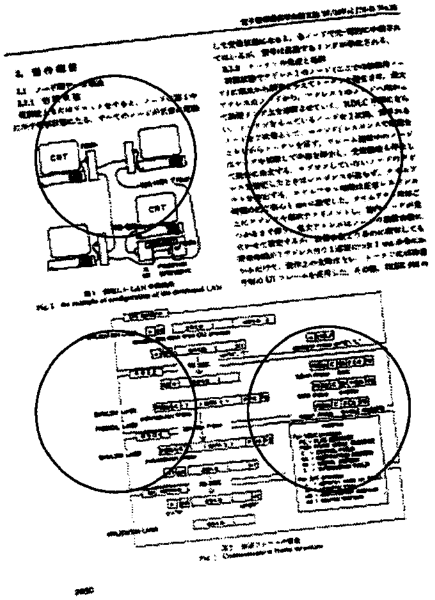


Figure 2: Local Regions are extracted for directional profile analysis from the document image

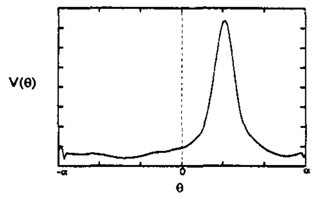


Figure 3: Directional Profile obtained by taking complexity variation measurements in several directions.

Local region complexity variance is profiled over a range of orientations. Complexity is measured by computing black pixels on scan lines oriented at some angle in the range. For a particular orientation θ, the variance in complexity $V(\theta)$ is:

$$V(\theta) = \frac{1}{n}\sum (N_i - M)^2,$$

where, n is the number of scan lines, N_i is the complexity of the scan line i, and $M = \frac{1}{n}\sum N_i$. Notice that the variance increases as θ approaches the actual skew angle.

Complexity variance measurements from a range of orientations define a directional profile, see Figure 3. The angle which maximizes the profile is reported as the skew angle of this local region. Observe that the mix of graphics, figures and text in local regions can cause directional profile analysis to yield misleading estimates of the skew angle. However, it is likely that skew estimates on several text regions cluster near a particular value. Hence the document skew angle is estimated as the weighted average of clustered skew estimates.

4.1.2 Skew Detector Results

The skew detector was tested with 467 artificially skewed images, with varying amounts of noise, fragmentation, black pixel density and ratio of text to graphics. Of these, skew was detected within 0.5% accuracy of actual skew in 71% of the images, within 1.0% of actual skew in 92% of the images and within 2.0% of the actual skew in 98% of the images. Results are also improved with high quality images when compared with poor quality document images. The document analyzer achieves 97% accuracy within ±1% of the actual skew on good quality document images, and 86% on noisy or complex documents. The average time required for skew detection is 2 cpu seconds on a SPARC-10.

4.2 Document Segmenter

This module incorporates a local-to-global approach for discrimination between text/non-text regions as well as segmentation of non-rectangular blocks, and is robust in the presence of noise.

The objective here is to decompose a scanned document image into regions, which contain homogeneous entities, such as text, graphics and half-tones. Further, this decomposition has to be performed across a wide variety of documents with no prior knowledge of document types nor constraints on the document layout. Other significant aspects of this module include robustness with respect to low quality data, capability of segmenting non-rectangular as well as rectangular regions, and applicability to multi-lingual document images. An outline of this algorithm is presented next, for details refer [23].

4.2.1 Local-to-Global Segmenter

This algorithm is based on two steps - (a) hypothesis generation and (b) hypothesis testing.

A segmentation hypothesis is generated by recursively partitioning the document into smaller regions and locating plausible partition points from local regions. Refer Figure 4 for initial regions generated by the recursive partitioning and Figure 5 for examination of plausible partition regions. A quad-tree of the image regions is formed while allowing for partial segmentation on low-quality/non-rectangular regions. Partitioning stops when either a high confidence partial segmentation is attained or the region is too small for further subdivision. Partial segmentation is assisted by profile analysis on local regions and statistical analysis on connected components.

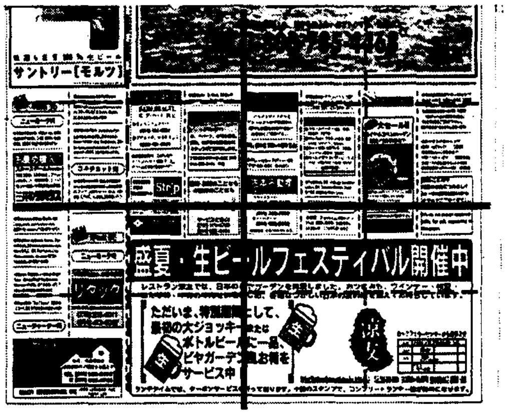


Figure 4: Recursive Partitioning of Document Image into smaller regions.

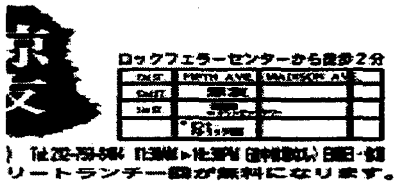


Figure 5: Locate Plausible Partitions from Local Regions.

Segmentation hypothesis testing involves the creation of region candidates based on a neighbor constraint satisfaction approach, refer Figure 6. Regions are then classified based on profile and run-length analysis, refer [24]. Creation of region candidates is accomplished by combining partial segmentations of 4 neighboring regions. Profile and component information is used to terminate propagation and block classification.

4.2.2 Results in Page Segmentation

The document segmentation module was tested with 200 Japanese and English documents. The performance of the document segmenter ranges between 92% completely correct segmentations and 94% partially correct segmentations. On an average this module requires 27 cpu seconds on a SPARC-10. The processing time of this module is directly proportional to the size and quality of the image as well as the layout complexity. Hence, on book images it takes approximately 17 seconds.

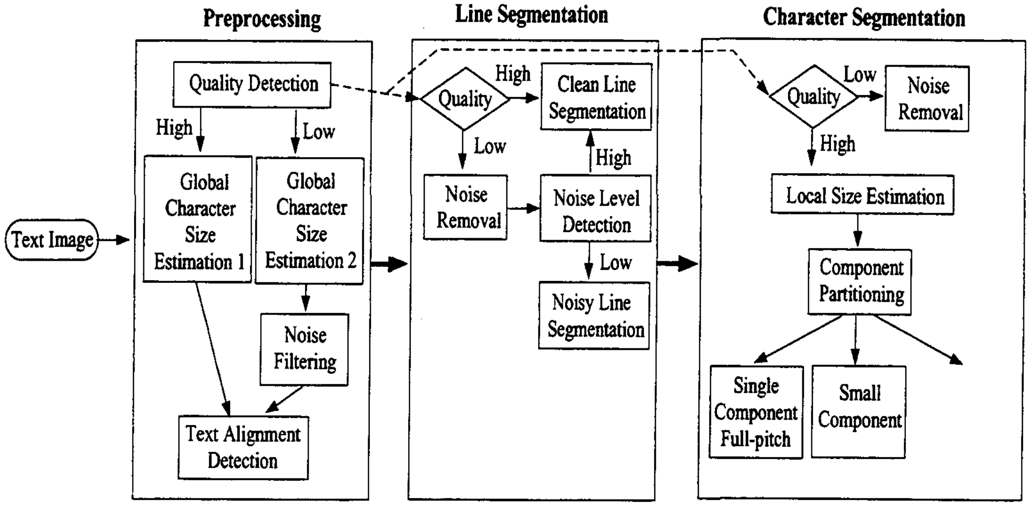


Figure 7: Overview of Text Segmentation.

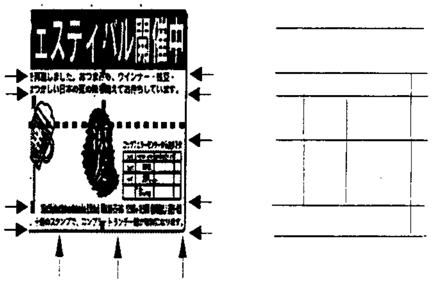


Figure 6: Test Hypothesized Region Candidates via Constraint Satisfaction.

4.3 Text Segmenter

This text segmenter module incorporates a divide-and-conquer approach. The segmentation task is divided into smaller ones and each task is then handled individually.

The text segmenter module was designed with the objective of segmenting multi-line text blocks into character units, and performing very reliably on high quality images and reasonably well with noisy images. Among the challenges in developing such a system is the diversity in document image quality. Print styles vary from clean and uniformly separated text units, to noisy and degraded. Font styles vary and inter-symbol spacing varies between closely spaced and widely spaced. When Japanese and English characters appear together, there is a large difference in size of each character type. It is often hard to distinguish noise from characters and segment characters of mixed pitch.

This module segments a text block into lines and characters via six subprocesses - image quality estimation, character size estimation, text alignment detection, text-alignment detection, line and character segmentation. An overview of this module is shown in Figure 7. Each of the subprocesses is described briefly here, for details refer [25].

4.3.1 Image Quality Estimation

This stage determines these aspects of image quality, particularly relevant to text segmentation. Several features are used in characterizing image quality, including:

- periodicity of profile pulse size (uniform/other)
- abnormal size of a pulse (noisy/other)
- symmetricity of pulse (font-varying/noisy)
- ratio of pulse-width/height (noisy/other)
- number of connected components in image (small/large)

4.3.2 Character Size Estimation

Character size is estimated globally based on histogram plots of connected components. In clean images, the highest peak in the component histogram corresponds to the width of full pitch characters. In noisy images, this component histogram is smoothed, mean width of the histogram is evaluated. The highest peak in the histogram which is also greater than the mean is chosen as the width of full pitch characters; refer Figure 8.

4.3.3 Text Alignment Detection

Two approaches to determine whether text is vertically or horizontally aligned are described here.

The first approach is based on a minimum spanning tree of components, while the second is based on direct projection profile analysis.

A minimum spanning tree is built with the connected components forming nodes in this tree. Each connected component is represented by its center and the minimum spanning tree is constructed with these points (this is similar to the technique of Ittner and Baird [3]). A histogram of edge orientations is built using these, the peak in such a histogram identifies text alignment direction. In another approach, the variance in horizontal and vertical projection profiles is analyzed. The direction of largest variance is the direction of the alignment, refer Figure 9.

4.3.4 Line Segmentation

In clean and high quality documents, separation of text blocks into lines is relatively simple. It is accomplished by estimating line breaks from valleys in the projection profiles. For noisy and degraded images, line segmentation is not as simple, as the projection profiles are not as sharply defined. Our approach is to filter out noise components during the estimation of line breaks. The projection profiles of text components are then smoothed and a recursive threshold is applied until there is only a small deviation in the main pulse widths of the projection profile. At this stage, the approach used for estimating line breaks of clean images can be applied, refer Figure 10.

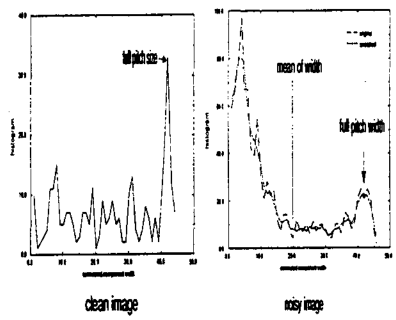


Figure 8: Global Character Size Estimation.

4.3.5 Character Segmentation

Initially character components are located by performing a projection profile analysis of black pixels and identifying white spaces as component separators. Refer Figure 11 for an illustration of this method. Local character size estimation is then performed for accuracy, based on a mean size estimate and line width. A rule-based approach is then used to partition components into three groups. These groups identify single component characters, partial character components and touching characters (or multi-character components). The sub-character components are merged based on a few simple rules to form complete characters. Touching characters are split based on further analysis and heuristics. At this stage a character recognizer has also been used to validate proposed splitting points. Figure 12 indicates how components may be merged or split under the rule-based scheme.

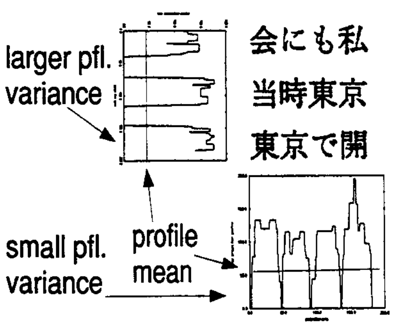


Figure 9: Projection Profile Variance based estimation of Text Alignment.

4.3.6 Text Segmenter Results

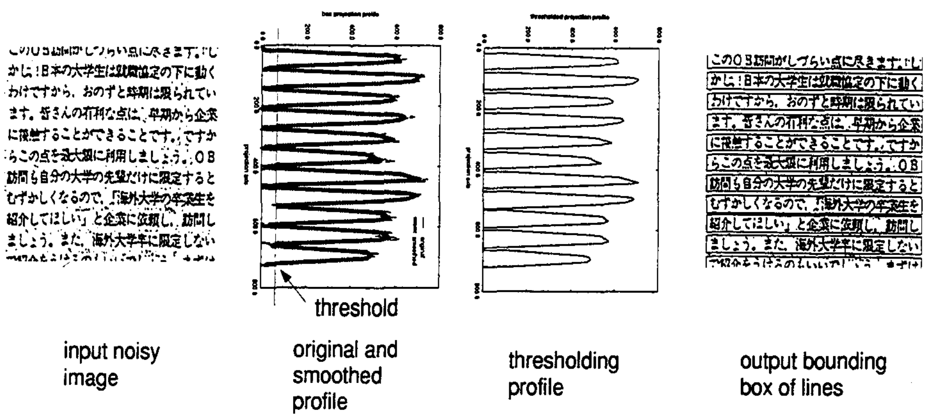
The text and character segmenter was evaluated with respect to several criteria, as shown in Table 1. Text segmenter performance varies between 98.5% for good quality images and 94% for degraded images. Simple connected component analysis is used for high quality printed text, while more complex analysis is applied for low quality document images.

4.4 Character Recognizers

Two independent character recognition modules have been developed to handle the 3300 classes of Hiragana, Katakana and JIS level 1 Kanji, alphanumeric & symbol characters. Hence the main challenges are to design recognizers that are fast as well as accurate in handling complex character shapes. The recognizers are based on (a) subspace and and (b) fast nearest-neighbor classifiers. After preliminary experiments using several kinds of feature sets, two of the best performing feature sets were chosen. These are the Local Stroke Direction (LSD) and Gradient, Structural & Concavity (GSC) feature sets. These feature sets are described next, followed by the discussion of the minimal error subspace classifier and the nearest neighbor classifier (with results).

4.4.1 Local Stroke Direction Features

When given a character image, its LSD feature vector can be computed as follows [12] (refer Fig-



input noisy image

original and smoothed profile

thresholding profile

output bounding box of lines

Figure 10: Line Segmentation after Profile Processing.

ure 13 for a detail example): first, for each black pixel, compute its directional run-length for each of four directions and normalize as a ratio to the total run-length in all directions; second, partition the image into $n \times n$ areas and compute the directional run-length of each area as an average of the pixels in the area. Here, we choose that n equals 8. Therefore, the size of LSD feature vector is $4 \times 8 \times 8 = 256$. In Figure 13 (c), the values in the LSD feature vector are scaled to integers at the range of 0 and 255 so that they can be visualized in a grayscale image.

4.4.2 Gradient, Structural and Concavity Features

The gradient and structural features encode local structure, while the concavity features are global descriptors extracted from binarized images.

A gradient map is constructed from the normalized digit image, by estimating gradient value and direction at each pixel. Figure 14 contains the gradient map of the character image. Histograms of directional features are recorded in each region - indicating presence (or absence) of a small number of oriented ranges of gradients in the region. Directional histograms from each region are concatenated into a fixed-length *gradient feature* vector. *Structural features* are computed from the gradient map by examining similarities in gradient direction in a local neighborhood of each pixel. These features record curvature in an approximate fashion. Curvature histograms indicating presence of changing orientation of character contours are estimated in each region. These histograms are concatenated from each region in a fixed-length structural feature vector. Gradient and structural features have been described in more detail in [26]. *Concavity features* are coarse global descriptors and are of three kinds: pixel density, large stroke, and true concavity. The large stroke features encode horizontal and vertical strokes in the image. Concave regions enclosed with a character are also recorded Concavity features have been described in detail in [27].

4.4.3 Minimum Error Subspace Classifier

Subspace methods have been a major field of study in pattern recognition [28, 29], particularly for classification and feature subset selection. The minimum error subspace classifier is a discriminant function derived from the Karhunen-Loeve expansion. It is given by:

$$g_j(X) = |X - M_j|^2 - \sum_{i=1,k} \{\phi_{ij}^T(X - M_j)\}^2.$$

Here, $g_j(X)$ defines the discriminant classifier for the j^{th} class, given an input measurement/feature vector X; M_j is the mean vector and ϕ_{ij}^T is the transpose of the i^{th} eigenvector of the covariance matrix for the j^{th} class; k is chosen as the dimension of the subspace defined by the dominant k eigenvectors. An unknown test feature vector is evaluated with these discriminant functions for each class, that is, $g_j()$, for $j = 1, 2, ..., C$, where C is the number of classes. The class of X is determined as that of the discriminant function for which the residual error is least, that is: $g_J(X) \leq g_j(X)$, $\forall j \in \{1, 2, .., C\}, j \neq J$.

Projections of an unknown test vector on the subspaces defined by the dominant eigenvectors of each class are subtracted from the projections on the entire eigen-space. The subspace of the class which best represents the unknown feature vector, results in the least residual error (and results in the least difference in projections between the whole space and the subspace of the principal eigenvectors). The class corresponding to the subspace with least residual error is chosen as the class-identity of the unknown test vector.

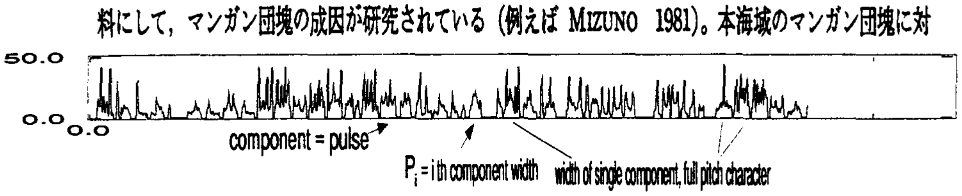


Figure 11: Character Segmentation from Projection Profiles.

This classifier can also be derived from the modified quadratic discriminant function [30], and can be interpreted as a quadratic discriminant classifier.

The training phase of this classifier requires the computation of the covariance matrices and their eigen-decomposition. For each class, the eigenvectors are then sorted in decreasing order of the corresponding eigenvalues. The mean vector M_j is evaluated for each class. Only the dominant eigenvectors (k, in this case) are used in the subsequent processing. During testing, the unknown or test feature vector is projected onto the subspace defined by these dominant vectors, for each class. The term $|X - M_j|^2$ defines the projection of the test vector onto the whole eigen-space. Hence, the class of an unknown vector is determined by the best subspace projection, given by the least residual error.

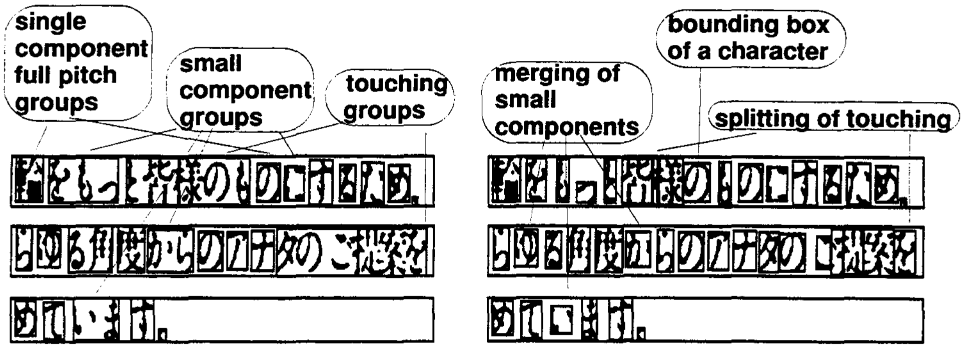
The classifier had been trained and tested on the CEDAR dataset, consisting of 175,988 character images from 3,354 categories. The training set contains 118,417 samples picked at random from each category and the test set contains the remaining 57,571 samples. Although the samples were extracted from document pages which were scanned in 400ppi, they are downsampled to 200ppi by applying a multi-rate method [31] to simulate degradation effect and used in our experiments. Using the LSD feature, the performance of the ME (Minimum Error) Subspace Classifier is listed in Figure 2.

4.4.4 A Nearest-Neighbor Classifier

A NN (nearest neighbor) classifier is designed for Japanese character recognition [32]. New algorithms for prototype reduction, hierarchical prototype organization and fast NN search are implemented in the classifier. In these algorithms, k-nearest/farthest neighbor lists are used to estimate the distribution of samples in the feature space. Given a set of samples, $P = \{p_0, p_1, ..., p_{n-1}\}$, for each sample p_i, its k-nearest neighbor list, $N_i = \{n_{io}, n_{i1}, ..., n_{i,k-1}\}$, and its k-farthest neighbor list, $F_i = \{f_{io}, f_{i1}, ..., f_{i,k-1}\}$, can be computed by comparing this sample with the rest of samples.

Given a training set, prototype reduction leads to the selection of a subset of samples which can represent the whole training set. Figure 15 illustrates the process of prototype selection. The subset can be generated by deleting redundant samples from the training set. By checking the pre-computed nearest neighbor list of each training sample, the prototype reduction algorithm decides whether a training sample can be deleted without negative effect on correct classification of itself and other samples if the sample is removed. Previous methods, such as Hart's *condensed nearest neighbor rule* (CNN) and Gates' *reduced nearest neighbor rule* (RNN), have to call the procedure of classification iteratively to test whether the deletion(or adding) of a prototype affects the correct classification of the other prototypes[33, 34]. The advantage of the method here is to avoid such an iterative process. A brute-force NN algorithm suffers from its computational complexity because every prototype has to be matched with the input pattern to see whether it is close to the input pattern. To speed up NN classification, we proposed a fast NN search algorithm, FNN. The basic idea is to avoid unnecessary comparisons. Suppose there is a test sample X to be classified (see the triangle in Figure 16). It has to compare with prototypes from the prototype set. After the comparison between the input pattern X and a prototype p_i, we know that the distance between them is very small. For those prototypes in p_i's k-farthest neighbor list, such as f_{i1},f_{i2} and f_{i3} shown in Figure 16, without going further to compare each of them with the input sample X, we know the distance will be very large and therefore they can not be in X's k-NN list. Similarly, after the comparison between X and a prototype p_j, we know that the distance between them is very large. For those prototypes in p_j's k-nearest neighbor list, such as n_{j1},n_{j2},n_{j3} and n_{j4} shown in Figure 16, without going further to compare each of them with the input sample X, we know the distance will be very large and therefore they can not be in X's k-NN list.

In order to further speed up the search process, The prototype set can be organized into a hierarchical representation. The prototype set is divided into two levels. The first level is the so-called "*centroid set*." For a centroid, there may be a set of prototypes which are stored in the second level. The set for a centroid is called as the centroid's "*package*." A centroid can approximately represent those prototypes in its package. The process to create


single component full pitch groups
small component groups
touching groups
bounding box of a character
merging of small components
splitting of touching

grouping process

merging and splitting process

Figure 12: Character Segmentation - Splitting and Merging of Components.

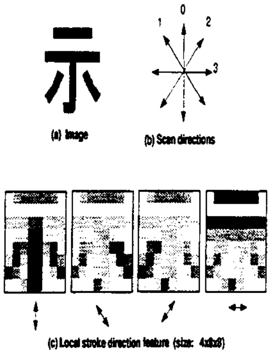


Figure 13: A Kanji character image and its *local stroke direction* (LSD) feature vector

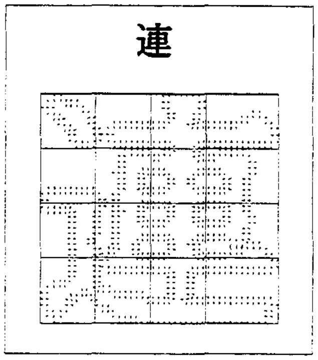


Figure 14: Gradient and Structural Features extracted from Gradient Map of Character Images.

such a hierarchy is to pack prototypes for selected centroids. Two versions of the algorithm are designed: one is *farthest-like-neighbor-based*; another is *nearest-unlike-neighbor-based.*

After packing prototypes, the k-nearest neighbor list and k-farthest neighbor list of each prototype in the centroid set C will be computed. The NN classifier can be described as a two-step procedure. Given a test sample X, its approximate k-nearest neighbors $\overline{N}_X$ in the centroid set C can be calculated using the search algorithm described above. Then the prototypes in the packages of those centroids in $\overline{N}_X$ will be compared with X to generate the final k-nearest neighbors N_X for the sample.

The NN classifier using LSD feature and city-block distance measure was trained and tested on the machine-print ETL dataset. This dataset, contains 26,002 training samples, which were reduced to 5,117 prototypes. These reduced prototypes resulted in 97.9% accuracy on the test set (containing 26,002 independent samples). On an average, each test image was compared to 56.8% of the prototypes.

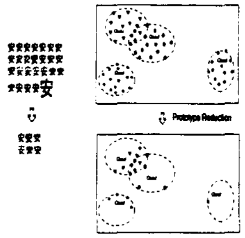


Figure 15: Prototype reduction

5 Future Work

An outline of the primary system modules of the Japanese OCR system developed at CEDAR, has been presented. Current efforts are directed towards integration of these components into a complete system. This system is expected to operate in

Table 1: Correct rate and Timing of character and line segmentation subsystems: CharsegC and LinesegC correspond to performance with high quality images; charsegN and LinesegN correspond to performance with noisy images.

Text Segmenter Performance					
Quality	Alignment	CharsegC	LinesegC	CharsegN	LinesegN
98.2%	100%	98.5%	100%	94.2%	98.6%
> 100 img/s	> 1000 img/s	41.5 c/s	> 1000 img/s	14.3 c/s	6.2 img/s

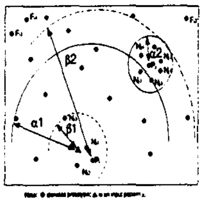


Figure 16: Speed up nearest neighbor search by avoiding redundant comparisons

Table 2: Results on 3300 character classes, at 200 *ppi*

Min. Error Subspace Classifier		
Top 1	Top 5	Top 10
93.48%	97.86%	98.45%

batch-mode to process document images and generate recognition output for each image. Robustness and reliability issues are currently being addressed. System performance will be evaluated via a *benchmarking* scheme. This benchmark will examine performance at several levels - individual modules and combination of certain modules as well as the entire system. To assist in this benchmark, we are in the process of generating a set of *ground truthed* Japanese document images - where the ground truth includes, skew angle, and block, line and character segmentation information. Future efforts will also be directed towards *postprocessing* of recognizer output to improve performance. Improvements in the algorithms used for page and text segmentation will be evaluated. The use of recognizer feedback in text segmentation has been promising. A thorough study of *integrated recognition and segmentation* is planned.

Team Members

Present and continuing team members include Tao Hong, Brian Grom, Chong O. Kim, Shu-Fang Wu and Weh-Chih Wang. Previous team members include Stephen W. Lam, Jonathan J. Hull, Victor C. Zandy, Qunfeng Liao, Chi Fang, Dar-Shyang Lee, Zhou Hong, Seiichiro Kashimori, Xiaoming Yang, Minsio Kariya, Xiaomin Ren, Yi L. Chiang and Daniel R. Mechanic.

References

[1] Q. Luo, T. Watanabe, and N. Sugie. A structure recognition method for Japanese newspapers. In *SDAIR*, pages 217 - 234, 1992.

[2] T. Akiyama and N. Hagita. Automated entry system for printed documents. *Pattern Recognition*, 23:1141–1154, 1990.

[3] D.J. Ittner and H.S. Baird. Language-free layout analysis. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 336–340, 1993.

[4] S. Tsujimoto and H. Asada. Major components of a complete text reading system. *Proceedings of the IEEE*, 80:1133–1149, 1992.

[5] S. Ariyoshi. A character segmentation method for Japanese printed documents coping with touching character problems. In *Proceedings of the International Conference on Pattern Recognition*, pages 313–316, 1992.

[6] Y. Kobayashi, K. Yamada, and J. Tsukumo. A segmentation method for hand-written Japanese character lines based on transitional information. In *Proceedings of the International Conference on Pattern Recognition*, pages 487–491, 1992.

[7] Y. Maeda, F. Yoda, K. Matsuura, and H. Nambu. Character segmentation in Japanese hand-written document images. In *Proceedings of the International Conference on Pattern Recognition*, pages 769–772, 1986.

[8] J. D. Becker. Typing Chinese, Japanese, and Korean. *IEEE Computer*, 18(1):27–34, 1985.

[9] J. K. Huang. The input and output of Chinese and Japanese characters. *IEEE Computer*, 18(1):18–24, 1985.

[10] K. Lunde. *Understanding Japanese Information Processing*. O'Reilly and Associates, Inc., 1993.

[11] R. Matsuda. Processing information in Japanese. *IEEE Computer*, 18(1):37–45, 1985.

[12] S. Mori, K. Yamamoto, and M. Yasuda. Research on machine recognition of handprinted characters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:386–405, 1984.

[13] K. Sakai, S. Hirai, T. Kawada, S. Amano, and K. Mori. An optical Chinese character reader. In *Proc. 3rd Proceedings of the International Conference on Pattern Recognition*, pages 122–126, 1976.

[14] R. Oka. Handwritten Chinese character recognition by using cellular feature. In *Proc. 6th Int. Joint Conf. Pattern Recognition*, pages 783–785, 1982.

[15] T. Ijima, H. Genchi, and K. Mori. A theory of character recognition by pattern matching method. In *Proc. First IJCPR*, pages 587-594, 1973.

[16] F. Kimura, K. Takashina, S. Tsuruoka, and Y. Miyake. Modified quadratic discriminant functions and the application to Chinese character recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:149–153, 1987.

[17] K. Miyahara and F. Yoda. Printed Japanese character recognition based on multiple modified lvq neural network. In *Proceedings of the Second International Conference on Document Analysis and Recognition*, pages 250–253, 1993.

[18] A. Konno and Y. Hongo. Postprocessing algorithm based on the probabilistic and semantic method for Japanese OCR. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 646–649, 1993.

[19] H. Fujisawa and K. Marukawa. Full-text search and document recognition of Japanese text. In *Symposium on Document Analysis and Information Retrieval*, pages 55–80, 1995.

[20] K. Kigo. Imporoving speed of Japanese OCR through linguistic preprocessing. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 214–217, 1993.

[21] S. W. Lam and V. C. Zandy. Skew detection using directional profile analysis. In *Proceedings of Machine Vision Applications*, 1994.

[22] Y. Ishitani. Document skew detection based on local region complexity. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 49–52, 1993.

[23] S. W. Lam. A local-to-global approach to complex document layout analysis. In *Proceedings of Machine Vision Applications*, 1994.

[24] L. O'Gorman. The document spectrum for page layout analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15, 1993.

[25] S. W. Lam, Q. F. Liao, and S. N. Srihari. A divide-and-conquer approach to Japanese text segmentation. In *Proceedings of the Conference on Document Recognition, 1995 SPIE Symposium(SPIE95)*, February 1995.

[26] G. Srikantan, S.W. Lam, and S. N. Srihari. Gradient-based contour encoding for character recognition. *Pattern Recognition Journal.* to appear.

[27] J.T. Favata, G. Srikantan, and S. N. Srihari. Handprinted character/digit recognition using a multiple feature/resolution philosophy. In *International Workshop on the Frontiers of Handwriting Recognition, IWFHR - 4*, 1994.

[28] E. Oja. *Subspace Methods in Pattern Recognition.* John Wiley & Sons, 1983.

[29] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, 1990.

[30] F. Kimura, M. Shridhar, and Y. Miyake. Relationship among quadratic discriminant functions for pattern recognition. In *International Workshop on the Frontiers of Handwriting Recognition, IWFHR - 4*, 1994.

[31] G. Srikantan, D.S. Lee, and J. T. Favata. Comparison of normalization methods for character recognition. In *Proceedings of the Third International Conference on Document Analysis and Recognition*, pages 719–722, 1995.

[32] T. Hong, S.W. Lam, J.J. Hull, and S.N. Srihari. The design of a nearest-neighbor classifier and its use for Japanese character recognition. In *Proceedings of the Third International Conference on Document Analysis and Recognition ICDAR'95*, 1995.

[33] P.E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, IT-14(3):515–516, May 1967.

[34] G.W. Gates. The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, IT-18(3):431–433, May 1972.

Document Understanding at SRI International

Prasanna G. Mulgaonkar, Jeff L. DeCurtins, Chien-Huei Chen, Gregory K. Myers
SRI International
Information, Telecommunications, and Automation Division
333 Ravenswood Avenue
Menlo Park, California 94025

Abstract

The ability to search digitized document images for relevant information is a growing need in many business and government applications. Conventional approaches to this problem involve the use of page, line, and character segmentation followed by optical character recognition to convert the pixel information into symbol strings that can be manipulated. Document degradations, however, cause the loss of important information at the pixel level, which in turn affects the quality of the characters extracted from the documents, and therefore, degrades the quality of the search results. SRI International has several related research efforts underway that are exploring the use of collateral, or "contextual," information that can compensate for the degradation-induced information loss. In general, these methods use shape information from entire words to complement character recognition; lexicons organized in domain-specific ways to enhance recognition; language models that can focus attention on parts of a document; and information combined from graphical and textual modalities within a single document. Several individual efforts are described in this paper.

The goal of our research is to produce systems that can synergistically extract information from different parts of a printed document to produce a consistent and searchable representation of the total information content.

1 Introduction

Businesses and government agencies need to collect and organize massive amounts of information that is currently available only in hard-copy form. These functions require the ability to extract information about the content and the logical structure of the documents from raster-scanned images. This information extraction process has traditionally relied on optical character recognition (OCR).

The major limitation of current OCR technology has been its inability to perform well on poor-quality documents. Most OCR systems assume that documents will have sufficient contrast and resolution, relatively undistorted character shapes, and low levels of image noise. Many faxes, multiple-generation photocopies, carbon copies, and documents printed on low-quality paper stock present images that violate these assumptions. Degradations cause information to be lost, and interfere with the segmentation and shape-based recognition of individual characters performed by conventional OCR systems.

Contextual information has tremendous potential to compensate for this loss of information. Most OCR systems rely on lexicons of valid words and n-gram probabilities to correct errors in the identification of individual characters. SRI International (SRI) has explored the use of additional sources of contextual information, such as the shapes of whole words, relationships between words, and cues from associated graphics, as well as novel ways to use word lexicons in the recognition process. In this paper we present several approaches that exploit different types of contextual information to solve specific information-extraction problems.

2 Mail-Piece Address Reading

SRI has been developing systems to locate and read addresses on machine-printed mail pieces for the U.S. Postal Service. Machine-printed mail pieces carry on their covers not only the destination address we wish to find, but also the return address, a postmark, and often one or more advertisements.

Our approach consists of two processing steps: address location and address recognition. Our approach to address location relies on the conventions of address structure to distinguish the address from other printing on the mail piece: these conventions include left justification as well as the length, height, and spacing of the text lines.

Our approach to address recognition uses as a subsystem as an off-the-shelf OCR software package modified for the environment of scanned letter mail, which, as we have noted, includes a significant amount of poorly printed text and interfering background patterns. To correctly interpret the character recognition results, SRI took advantage of the logical relationships among the words' addresses, to use a method based on hypothesis generation and verification. In an address, the street should be part of the city, the city should belong to the state, and the street-city-state combination should correspond to the ZIP code.

SRI uses the character recognition results from parts of an address to access address directory databases in order to generate hypotheses about the corresponding elements in the remainder of the address. Our system then verifies these hypotheses by examining the character recognition results from the remainder of the address. Figure 1 shows an example of a poorly printed address line, the character hypotheses produced by the OCR subsystem, and the word hypotheses generated and verified by the system.

3 Text Recognition without Word Segmentation

When an OCR process is run on scanned newspapers, interference from the background between words sometimes causes the OCR process to produce phantom characters where an empty space ordinarily would be detected. In these cases, a postprocessing step that attempts word recognition would not be able to rely on correct segmentation of words. To solve this problem in processing low-quality newspaper images, SRI developed a text recognition method that processes entire lines of text at a time [1]. In this approach, the output of a character recognition process is treated as a continuous string of characters instead of first being divided it into words before word-level contextual knowledge is applied. Hypotheses of word identities and their positions are based on "seed features" (high-confidence character substrings) extracted from the output of the character recognizer. The verification of these hypotheses consists of selecting the set of consecutive word hypotheses that best fit the character recognition data and are the most mutually consistent across the entire line.

This approach is relatively impervious to errors in word segmentation. For example, if two words are joined, correct word hypotheses can still be generated from partial substrings. The lack of a delimiter between the words is treated as just a character deletion, which usually does not significantly alter the overall match between the sequence of characters in the hypothesized words and the character identities generated by the OCR process.

4 Lexicon-Driven Word Recognition

We have developed an original approach to the usage of lexicons for word recognition; we call this approach *lexicon-driven* word recognition [2, 3]. Instead of using a lexicon in a postprocessing step, our approach tightly integrates the lexicon into the recognition process. The rationale for this approach is rather simple: Given a word image, it is much easier to answer the question "Is this word X?", where X is some lexicon word, than it is to answer the question "What word is this?" This statement is also true for characters. Based on this rationale, SRI's approach performs recognition in a hypothesize-and-test framework. Given a word image, the recognition system extracts a set of likely candidate words from the lexicon, and tests which word best interprets the word image. To test a lexicon word against the word image, the system sequentially tests the characters of the lexicon word against the corresponding character images.

This approach has two important characteristics:

- It is segmentation-free. In this approach, both the position of a character and its identity are determined together; therefore, there is no need to presegment word images into isolated characters. Thus, we have avoided the very difficult and error-prone process of segmentation.
- It is effective in character recognition. Most conventional OCR systems perform character recognition by using classification techniques. Since the number of possible character classes is fairly large, those systems must use a large set of features to be able to discriminate among all the classes. In our approach, however, character recognition is simply the verification of a character hypothesis (from a candidate lexicon word). Therefore, we can selectively apply a set of features that is smaller and yet more effective in identifying the predicted character. As a result, each individual character can be more accurately identified.

There is, however, one critical issue we must address for this approach to be practical in general-purpose document understanding—that is, its ability to handle very large lexicons (e.g., the UNIX dictionary, which contains more than 40,000 words). We address this issue in two steps:

1. First, we organize the lexicon in such a way that, given a word image, we can quickly retrieve a small subset of lexicon words that is very likely to contain the correct word. We call this step *lexicon indexing*: the lexicon is indexed by word image features that are relatively invariant to different fonts, sizes, and image degradation.

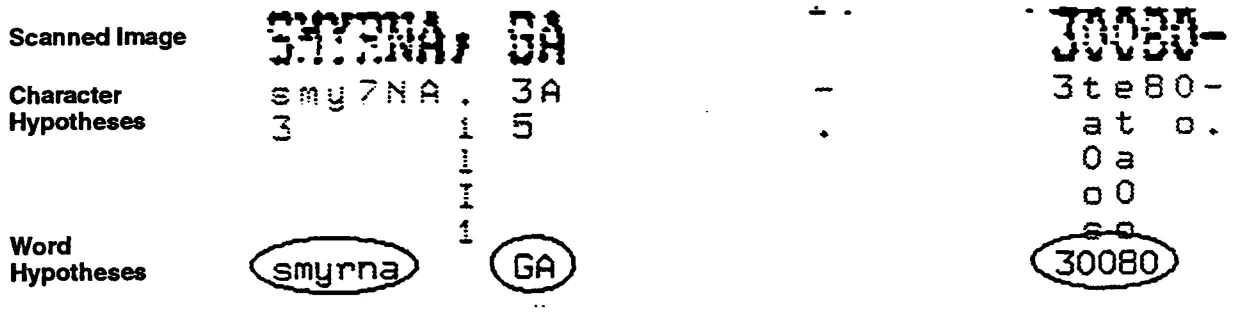


Figure 1: Verification-based recognition.

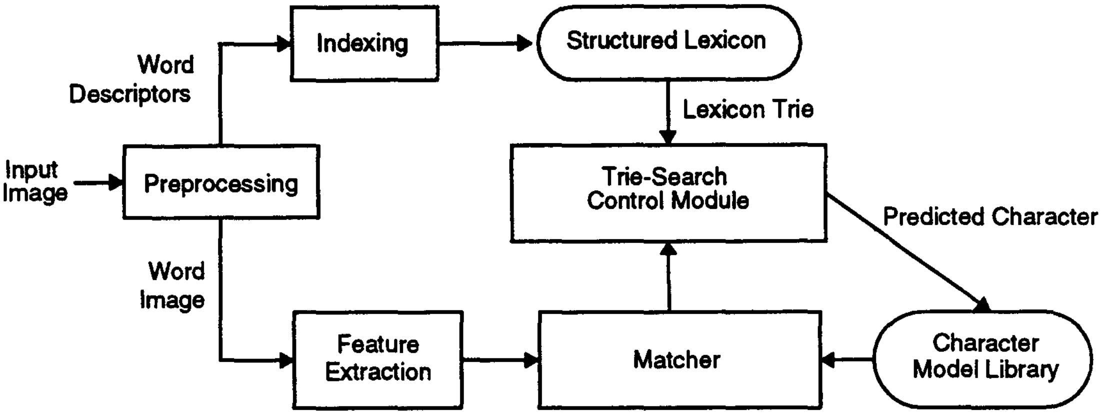


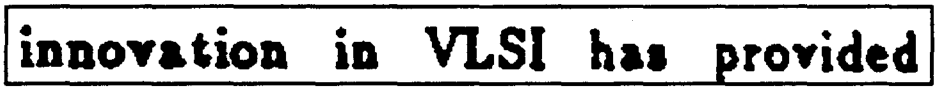
Figure 2: Lexicon-driven word recognition.

2. To facilitate the pruning of erroneous words from the lexicon, we use a trie to encode the lexicon words. Each node in the trie data structure represents a character, and each lexicon word is represented by a unique path from the root to a unique terminal node. In this data structure, a substring of characters from the root to a node is the common prefix shared by all the encoded words of the branches below the node. The task of finding the lexicon word that best matches a word image can be formulated as a search for a complete path, in which the characters along the path yield the best overall match to the word image.

A schematic diagram of our approach is shown in Figure 2. The input image is first segmented into blocks of isolated words. Each isolated word image is normalized to a standard height, and then the word image features are computed and indexed into a lexicon trie. The trie-search control module searches the lexicon trie and sequentially generates a character hypothesis to be tested on the word image. The matcher retrieves the respective character model and matches it against the features of the word image. The match result is fed back to the control module to steer the search. The word image is recognized if a lexicon word's characters all match well to the image. If none of the lexicon words yields a good match to the image, the control module switches to a bigram search to find the best sequence of bigram that match the word image.

The results of recognition processing on two severely degraded images are shown in Figure 3. For comparison, we show results from SRI's approach and two leading commercial OCR systems. We can see that our approach recognized several words better than the two commercial OCR systems did, especially those words where much

Image 1:


innovation in VLSI has provided

SFOCR:

innovation in VLSI has provided

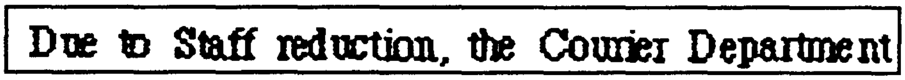
Commercial OCR 1:

inao~ltion Jo VLSI MU pro~idt~

Commercial OCR 2:

innovation in VLSI has provided

Image 2:


Due to Staff reduction, the Courier Department

SFOCR:

Due as Staff reduction he Courier Department

Commercial OCR 1:

Do to Staff redul:tion~ the Gouger Depar~ent

Commercial OCR 2:

Due ID Sum ndwtion, & Couner Deparmni

Figure 3: Results of recognition processing on two degraded images by SRI's system and two commercial OCR systems.

character merging occurs. (Note: the word "VLSI," which was not in the lexicon, was confirmed by bigram search after the lexicon-trie search failed.)

We are currently investigating how to effectively use interword contextual constraints in this approach. Word unigram, bigram, and trigram statistics can act as additional pruning mechanisms to discard unlikely word hypotheses. In addition, they can generate correct word hypotheses that have been mistakenly discarded in the lexicon indexing step.

5 Information Extraction by FASTUS

Information extraction differs from full text understanding: instead of full comprehension of the entire text, the goal of information extraction is to extract only the particular information that relates to a specific task. Therefore, information extraction is appropriate for text-scanning tasks in which only a fraction of the text is relevant, and the desired information can be characterized adequately with syntactic language structures. Techniques for information extraction have been developed independently in information retrieval, message understanding, and other fields in which ASCII text is already available (having been entered manually or generated by humans).

SRI's approach for text information extraction is FASTUS [4, 5], which uses a set of cascaded finite-state automata (FSAs) to search natural language text for specific logical structures of interest. The operation of FASTUS comprises four steps:

1. **Triggering**: Sentences are scanned for key words to determine whether they should be processed further. For example, in the domain of terrorist incidents words such as "terrorist," "killed," and "bomb" are trigger words.
2. **Recognizing phrases**: Sentences are segmented into noun groups, verb groups, and particles.
3. **Recognizing domain patterns**: The sequence of phrases produced in Step 2 is scanned for patterns of interest. When they are found, corresponding "incident structures" are built. An example of a pattern of interest in our application is: <Perpetrator> <Killing> of <HumanTarget>. An incident structure for terrorist incidents would include the date and type of incident, the perpetrator and organization (if known), the instrument used (e.g., gun, bomb), and the number and type of targets (both human and physical).
4. **Merging incidents**: Incident structures from different parts of the text are merged if they provide information about the same incident. Merging is especially important for analyzing news reports, which often mix events from several incidents in a single report, or mention the same details more than once.

Many systems have been built to match patterns in strings of words. The crucial innovation in the FASTUS system is the division of pattern matching into two steps: recognizing phrases, and recognizing domain patterns. Phrases can be reliably recognized with purely syntactic information; they provide precisely the elements that are required for stating the patterns of interest.

For example, consider this report:

Salvadoran President-elect Alfredo Cristiani condemned the terrorist killing of Attorney General Roberto Garcia Alvarado and accused the Farabundo Marti National Liberation Front (FMLN) of the crime.

From this report, FASTUS would produce the following incident structure:

```
Incident:     KILLING
Perpetrator:  FMLN
Confidence:   Suspected or Accused by Authorities
Human Target:"Roberto Garcia Alvarado"
```

SRI has integrated FASTUS into an experimental end-to-end system that automatically extracts relevant information from printed documents [6]. Such a system would be used by an analyst to quickly gather information from diverse sources of printed material. This system scans a document page, segments the page image into multiple text regions, determines their reading order, recognizes the characters and words in the text, searches the recognized text, and automatically extracts the desired information. This system has been used successfully to scan recent newspaper articles on terrorist events in Mexico.

6 Keyword Spotting via Word Shape Recognition

With the advent of on-line access to very large collections of document images, the electronic classification of documents into areas of interest has become possible. A first approach to classification might be the use of OCR on each document, followed by the analysis of the resulting ASCII text. But if the quality of a document is poor, the format unconstrained, or time a critical factor, complete OCR processing of each image is not appropriate.

An alternative approach is the use of word shape recognition (as opposed to individual character recognition) and the subsequent classification of documents by the presence or absence of selected keywords. The use of word shape recognition not only provides a more robust collection of features but also eliminates the need for character segmentation (a leading cause of error in OCR).

SRI has developed a system for the detection of isolated words, word portions, and multiword phrases in images of documents. It is designed to be used with large, changeable keyword sets and very large document sets. The system provides for the automated training of desired keywords and the creation of indexing filters to speed the process of matching.

We seek to develop word-shape recognition because evidence suggests that humans, who can easily decipher degraded text images, do so on a word level rather than a character level. Our approach is based on the concept of occluded-object recognition. We treat words as touching or occluded objects that are subject to special constraints on their positions, i.e., they are juxtaposed with little or no freedom in rotation.

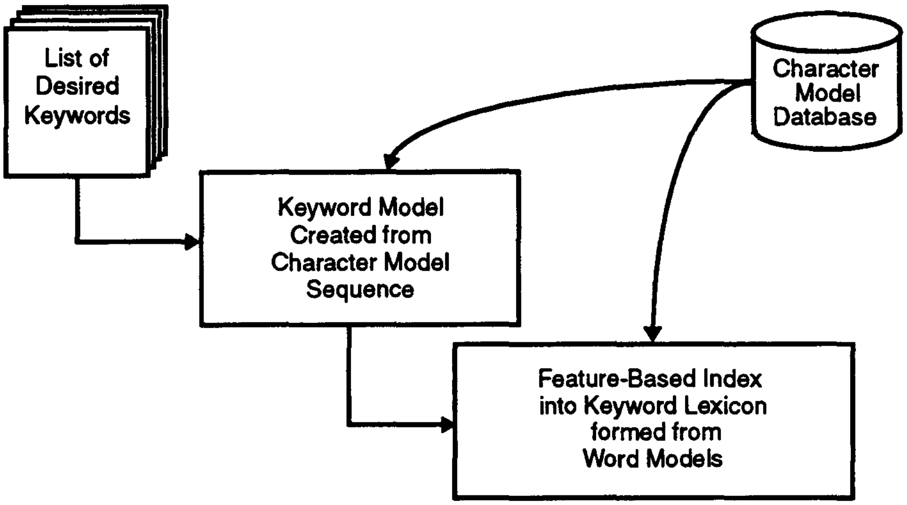

List of Desired Keywords
Character Model Database
Keyword Model Created from Character Model Sequence
Feature-Based Index into Keyword Lexicon formed from Word Models

Figure 4: Model construction phase (off line).

Our approach consists of two phases. First, the model construction phase takes place off line (see Figure 4). This process establishes the correlation between lexicon words and the attributes of features extracted from images of the isolated characters that make up the words. We then form a feature-based index into the lexicon so that we can handle lexicons consisting of thousands of words.

Second, the word recognition phase uses features and their attributes extracted from the document image at run time to accumulate evidence for the existence of specific words at specific locations (see Figure 5). Only line segmentation is necessary at run time, because the features are extracted from the entire text-line image.

Our system is insensitive to noise, occlusion, and touching words or characters, as long as enough of each word is visible and a robust set of feature detectors is used. Because of its minimal segmentation nature, this method is directly applicable to situations where, even in high-quality images, word or character segmentation is nearly impossible, such as the processing of documents in cursive script or in many foreign languages. In addition, the process is not limited to isolated words. It can detect embedded word portions (e.g., "cut" in "executive") as well as multiword configurations (e.g., "tary info" in the phrase "proprietary information").

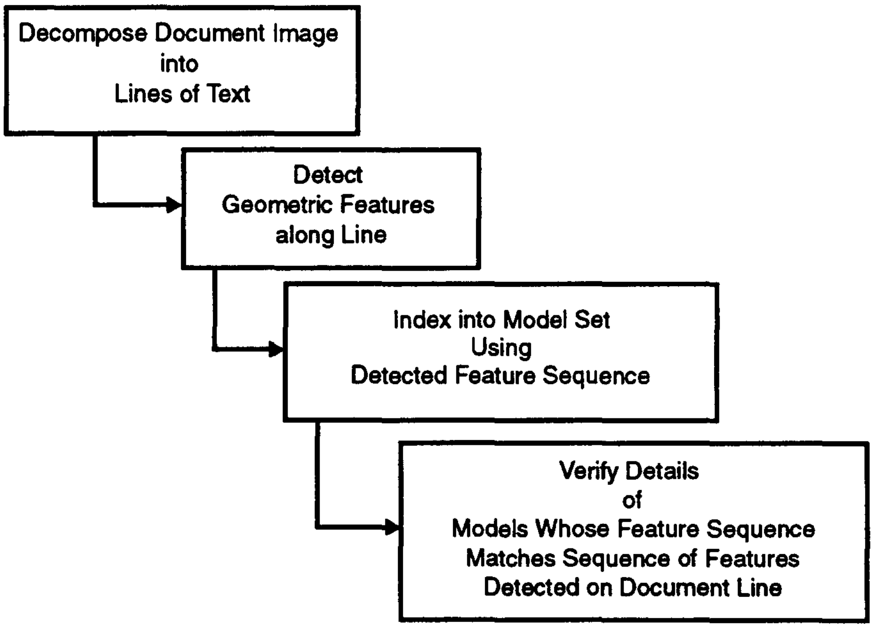

Decompose Document Image into Lines of Text
Detect Geometric Features along Line
Index into Model Set Using Detected Feature Sequence
Verify Details of Models Whose Feature Sequence Matches Sequence of Features Detected on Document Line

Figure 5: Word-shape recognition phase.

The current prototype consists of a Motif[1]-based graphical user interface through which the user specifies keywords and collections of documents to search. The lexicon formation and search processes are initiated by this interface in a client-server paradigm.

The system can search for keywords in English, Cyrillic, and Arabic documents. The entry of Cyrillic and Arabic is facilitated by a "soft keyboard" displayed on-screen. For a lexicon of about one hundred words, the typical execution time on a Sun Microsystems, Inc. (Sun) SPARCstation 10 is between 1 and 2 s per document page. Recognition rates vary between 75% and 100%, depending on the quality of the image.

7 Information Extraction from Maps

Approaches to the automated interpretation of raster-scanned maps have generally assumed that maps are composed of various graphic entities, and that the vast majority of pixel positions on the map each belong to only one type of graphic entity and can therefore be geometrically segmented. As a result, the processing steps in these approaches have generally been to (1) isolate each of the graphic entity types (e.g., lines, text); (2) apply entity-specific recognition; and (3) merge the results into an integrated interpretation by applying some contextual knowledge. However, complex color topographic maps contain several layers of information (e.g., hydrographic layers, terrain, transport, elevation, political divisions, and urban areas) that overlap substantially (often within a single color plane), making it impossible to geometrically segment the map data into distinct regions, each of which contains a single class of graphic object. Because the graphic characteristics of the information in each of these layers differ, an approach that applies the same processing uniformly to features in all layers may not yield optimum results. For example, the parameter values that are appropriate for controlling a vectorization and linking process for one type of linear feature (e.g., roads) may not work well on another type of linear feature (e.g., streams).

SRI has started a preliminary investigation [7] of verification-based recognition approaches that are expected to be much more successful, because they take advantage of the general principle that objects and their relationships can be recognized much more successfully if algorithms know what to look for and do so in a directed search. A verification-based approach uses contextual knowledge and constraints to formulate and then verify interpretation hypotheses. Because they interpret the pixels in terms of the possible hypotheses, verification-based approaches should have several important advantages:

- They can operate successfully amid extraneous graphic information, even where the graphical object of interest is touching or overlapping other information. Recognition techniques that depend on having the object of interest isolated, or the unwanted background "removed" from the image, may not be as successful, because an error in the isolation or removal process will preclude success in the subsequent processing steps.
- They are better able to take advantage of the rich a-priori knowledge associated with most maps. Information from legends, map specifications, gazetteers, and existing digital databases can be a guide to the map data and can drive the interpretation process.
- They are better able to take advantage of the interrelationships between information expressed in different graphical modes. For example, the presence of a bridge symbol can bolster the hypothesis that two linear water segments located on either side of a road should be linked. Similarly, feature hypotheses can drive a search for particular text labels, and the hypothesis of a text label can drive an additional search for the feature with which it should be associated. In other words, both the detection and the association of related map information can produce hypotheses that are confirmed simultaneously only when a consistent interpretation of all of the data is found.
- A verification-based approach is amenable to extracting only the data that are of interest. Therefore, such an approach requires less processing time than approaches that try to interpret everything on the map. This capability is especially valuable when different information is desired from several types of maps covering the same geographic area.

We present below two examples of the power of a verification-based approach for interpreting raster-scanned map data. Each of the examples contains a technique that is focused on extracting a specific type of information from the map, and each of the techniques works amid the clutter of the remaining graphic information on the map.

7.1 Map Symbol Feature Extraction

Point features on a map are represented by two-dimensional symbols, and areal features are represented by regions that are defined by either a fine print screen pattern or a repeated pattern of a discrete symbol. In many cases, these symbols are touched or overlapped by other map entities. Figure 6 shows part of an image, from a United States Geologic Survey (USGS) topographic map, that contains several marsh symbols. Notice that there is appreciable variation in the shapes of the printed symbols.

To this test image we applied object recognition techniques developed by SRI for industrial computer vision domains. In these techniques, training of the system is performed automatically by showing it objects in sample images. Object characteristics are extracted and ranked automatically according to how reliable they are relative to image noise and clutter, and how well they contribute to uniquely distinguishing one object from

[1] All product or company names mentioned in this document are the trademarks of their respective holders.

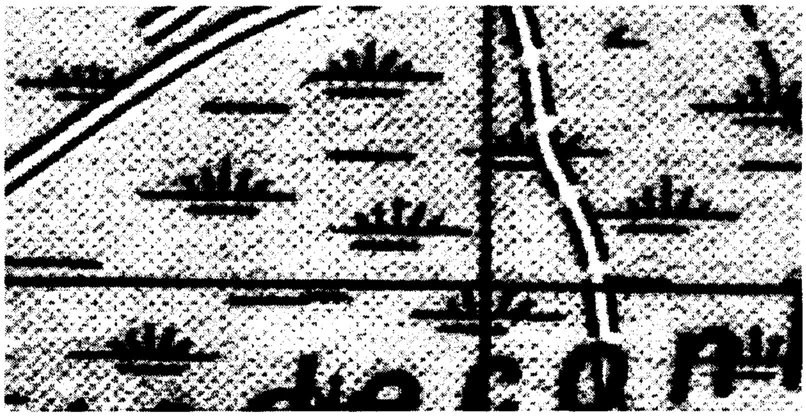


Figure 6: USGS topographic map.

another. During recognition, key characteristics are detected in the image and trigger the generation of hypotheses about the presence and location of particular objects. The hypotheses are verified by checking for the presence of the remaining characteristics in the object model. This method can recognize objects in spite of partial occlusion or noise in the image data.

Figure 7 shows rectangles superimposed over recognized marsh symbols in the test image. Notice that many symbols are correctly recognized, in spite of the variation in their shapes and occlusion by other map entities, and that most of the unrecognized symbols were only partially visible.

7.2 Text Extraction and Association

An example of a verification-based approach for text labels is shown in Figure 8. The black image plane containing both text and graphics information was extracted from a 1:24,000-scale raster-scanned USGS topographic map. A gazetteer was used to automatically generate hypotheses of the characters in the labels and their approximate locations on the map. Text-like shapes that were detected were tested against hypothesized labels, via SRI's approach for keyword recognition (described in the previous section). The set of hypothesized text strings enables us to take advantage of the context for the whole label instead of just independently recognizing isolated characters; and the likelihood of correct recognition of the complete text string on the map can be increased enormously. The three text labels in Figure 8 were successfully detected as text zones and recognized by our software, in spite of the label's coarse resolution and the interference from the black image plane that is intermingled with the text.

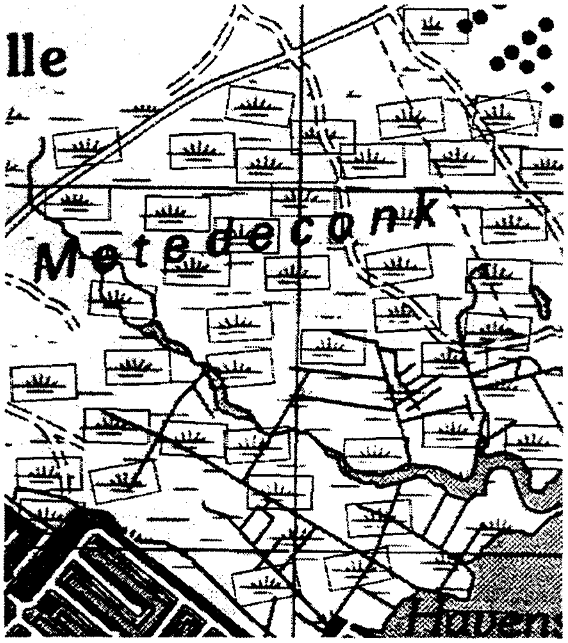


Figure 7: Test images with recognized symbols.

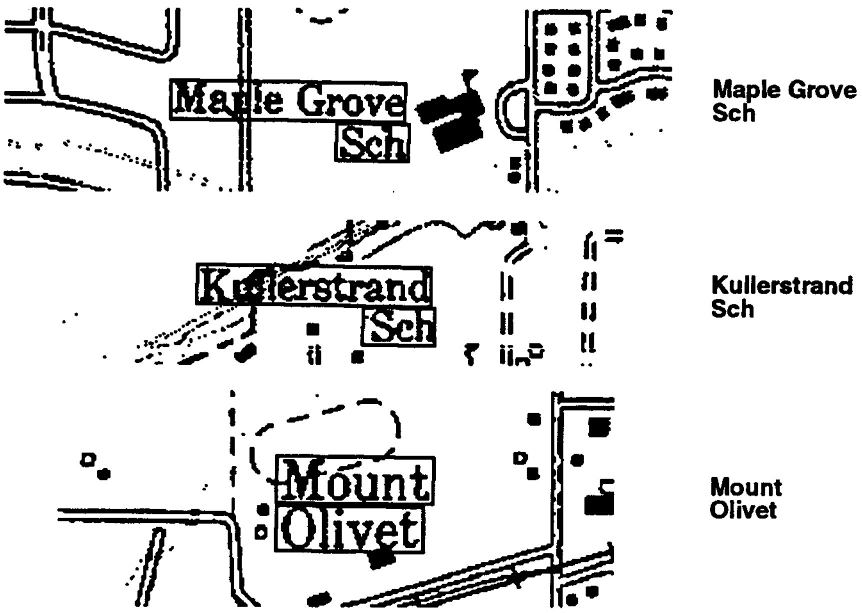


Figure 8: Successfully recognized text labels.

8 Summary

SRI has combined the use of various sources of context information, such as word lexicons, character and whole word shapes, and interword relationships, to develop effective methods for recognizing and extracting information from degraded documents. These context sources allow us to form hypotheses about the true information content of the corrupted areas of the document. We have found that the hypothesize-and-test paradigm is an effective and efficient way to control the application of the contextual information.

SRI is beginning to explore ways of further improving our performance potential by integrating processes on a larger scale. For example, we can exploit the fact that in some cases the same information is represented in multiple modalities, and that each of these modalities is affected differently by document degradation. Integrating the extraction of graphic map symbols and of their associated text labels, which represent the same information, into a single control process rather than separate processes will increase the likelihood of detection (perhaps through setting symbol detection thresholds to be more sensitive when a label is hypothesized, or by generating a text label hypothesis from a detected symbol). In another example, before we applied FASTUS to a printed document, we ran an OCR process on the entire document. By integrating our keyword spotting capability with FASTUS, we can improve the efficiency of the information extraction process, letting the presence of certain proper nouns trigger the application of FASTUS and thus avoiding the time-consuming process of establishing which parts of a document are relevant.

References

[1] G.K. Myers and C.-H. Chen, Lexicon-based word recognition without word segmentation, in *Third Annual Symposium on Document Analysis and Information Retrieval,* April 1993, 117–188.

[2] C.-H. Chen, Structuring a large lexicon for word recognition, presented at the Fourth Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, Nevada, 24–26 April 1995.

[3] C.-H. Chen, Lexicon-driven word recognition, in *Proc. Third Internatlonal Conference on Document Analysis and Recognition,* Montreal, Canada, 14–16 August 1995.

[4] J.R. Hobbs, D.E. Appelt, J. Bear, D. Israel, and M. Tyson, *FASTUS: A System for Extracting Information from Natural-Language Text,* SRI Technical Note 519, SRI International, Menlo Park, California (November 1992).

[5] D.E. Appelt, J.R. Hobbs, J. Bear, D. Israel, M. Kameyama, and M. Tyson, The SRI MUC-5 JV-FASTUS information extraction system, in *Proc. Fifth Message Understanding Conf. (MUC-5),* Baltimore, Maryland, August 1993.

[6] G.K. Myers and P.G. Mulgaonkar, Automatic extraction of information from printed documents, presented at the Fourth Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, Nevada, 24–26 April 1995.

[7] G.K. Myers, P.G. Mulgaonkar, C.-H. Chen, J.L. DeCurtins, and E. Chen, Verification-based approach for automated text and feature extraction from raster-scanned maps, presented at the International Workshop on Graphics Recognition, University Park, Pennsylvania, 10–11 August 1995.

Additional Submissions

Heuristic-based Object Extraction and Recognition

Raymond J. Bennett
HRB Systems, Inc.
State College, PA 16804
E-mail: *rjb@icf.hrb.com*

Abstract

The overall objective of our research has been to develop image processing algorithms which address specific technical problems identified by the Government. We initially developed and demonstrated a capability to perform coarse sorting of logos (and similar graphical objects) into shape-related categories. This research progressed, under government sponsorship, to the development of a multi-variate classification scheme for logo recognition which provided accurate (99%) recognition of (hand segmented) logos from poor-quality images.

Our research was then directed toward the problem of logo segmentation. Most available segmentation approaches experienced some difficulty in extracting graphical objects in a consistent manner, particularly when the image quality was such that even good OCR techniques did not perform well. Because any logo recognition technique relies on the segmentation, we began researching logo segmentation.

Our segmentation approach encompasses deterministic and decision-theoretic techniques. This segmentation has proven to be more robust and inherently more understandable than its neural network counterparts for this particular task.

This paper will discuss the results of our research and implementation efforts in logo recognition and logo segmentation for poor quality document images.

1.0 Introduction

Over the past decade, document traffic has increased dramatically on the Public Switched Telephone Network. Because of the wide variety of visual information that document messages contain, automatic sorting continues to be an elusive goal. Gleaning message content from a document requires the extraction and classification of information-bearing image artifacts from the optically-encoded transmission. Such artifacts include character sets, pictures, graphics, handwriting, and anything else which can be committed to a two-tone representation.

The problem of the information analyst is that of manually reviewing the input image to arrive at an understanding of the document's content, a procedure increasingly overburdened by the volume of documents. Therefore, specialized algorithms which aid by identifying commonly occurring artifacts can provide significant assistance immediately.

A major objective of this research has been to help the analyst effectively deal with overwhelming amounts of poor quality document image data. In the following sections, we will discuss our research and development efforts in Logo Recognition and Logo Segmentation. We also provide a brief discussion of a prototype system which accomplished the integration of these technologies.

2.0 Logo Recognition

Logo recognition was developed as an approach to document sorting based on the ubiquitous organizational logo - a quasi-graphical element which appears in a surprisingly large percentage of relevant documents. As the logo generally identifies the message originator, it provides an effective mechanism for sorting and routing documents of interest.

When considering the problem of logo recognition, we initially proceeded under the assumption that robust segmentation techniques would provide our system with consistent images of logos for processing. The initial portions of our research then, did not address the segmentation of logos, but rather assumed that acceptable logo data would be available.

As a starting point, we attempted to define the subject of our research - *what is a logo?* This question has no easy answer. We eventually adopted a qualitative definition: *a logo is an image element which is (at least partially) graphical in nature, and in general, stands out stylistically from the remaining elements on the page.*

Our logo recognition research occurred in two distinct historical phases. The initial phase performed a first level sorting of input logos into general shape categories. The second phase extended this sorting technique to provide a specific logo recognition capability.

2.1 First Level Sorting

We initially set out to develop algorithms which would produce a first level sort of candidate input logos into top-level sets. This top-level sort provides the

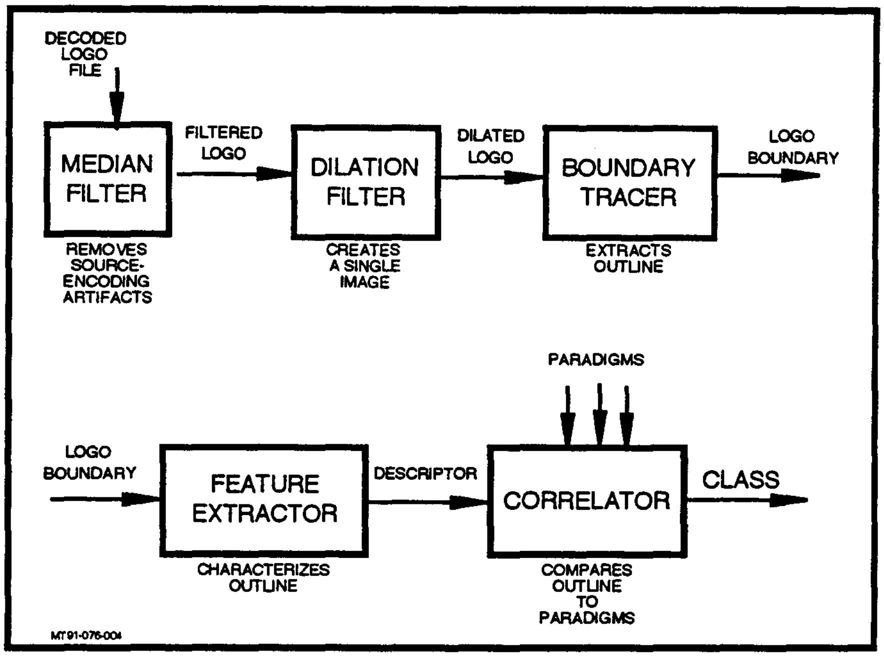


Figure 1: First Level Sorting - Algorithm Flow

framework for a hierarchy of additional, lower-level decision algorithms. We found that geometric figures provided an intuitive and useful top-level classification of logos. Specifically, the archetypes we chose were: *square, 1x2 rectangle, 1x4 rectangle, circle, and triangle*. A sixth category absorbed those logos which did not match any of these five categories particularly well. These six classes serve as the sorting categories as shown in Figure 2.

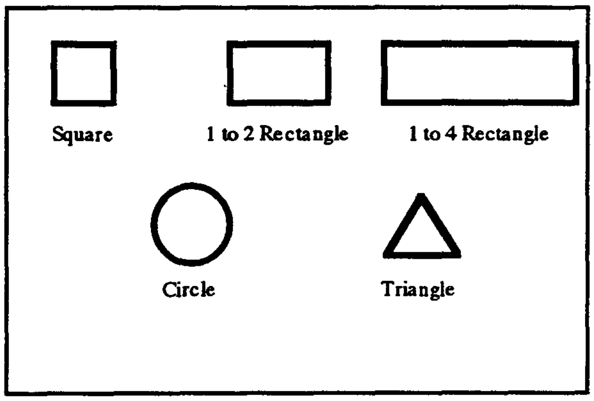


Figure 2: Five Basic Sorting Paradigms

Figure 1 shows algorithm flow for the first level sorts which we perform. The flow may be divided into three basic types of operations: Logo reconstruction and filtering; feature extraction; and correlation.

2.1.1 Logo Reconstruction and Filtering

The logo candidate presented to the system may contain discrepancies which adversely affect the recognition capabilities. These errors appear as dots or voids in the reconstructed image and are handled using a 3-pixel square median filter. This filter assigns pixel values according to the predominant pixel value in its 3x3 neighborhood. This operation is both simple and efficient, and improves the similarity of like logos originating from different source machines. Logo image fields are then presented for feature extraction.

2.1.2 Feature Extraction

The top-level logo sorting technique is based on similarities between a logo's basic outline (its contour) and those of the archetypes. To maximize the effectiveness of our system, we connect the elements of the logo while attempting to retain sufficient boundary articulation to allow meaningful sorting. We employ a morphological dilation filter to generate a single object from the elements which constitute the logo. The dilation radius is selected individually for each logo, so as to avoid the extremes which blurring with a constant radius would yield. This radius is determined using the second-order statistics of the black-pixel distribution.

Next, we extract the boundary of the resulting object using a clockwise-moving, counterclockwise-seeking algorithm. We subsequently re-sample the boundary for normalization and employ a proprietary algorithm to generate a descriptor for the logo which is both scale and rotation invariant.

2.1.3 Correlation

Feature extraction operations reduce each logo to a Fourier angular descriptor. Sorting of the logos is accomplished by comparing the descriptors of the candidate logos with those which have been stored for the archetypes. Since the descriptor may be viewed as an analytic signal, the most readily available method for comparing two logo descriptors is the complex cross-correlation. We then exploit the circular nature of the discrete cross-correlation, thereby allowing us to compute *R(m)* with N-element FFTs.

2.1.4 First Level Sorting Test Results

An important characteristic of our descriptor approach is its relative insensitivity to changes in orientation and scale of the target logo. Specific testing for correlation among logos which ranged in scale from 100% down to 25% was conducted. Additionally, six non-orthogonal rotations of logos were selected for correlation against the original. Top-level sorting was conducted using a 100 logo database of commercial and government logos of varying sizes and rotations. Additionally, 50 logos which were a variant of the first 100 in either size or rotation were added to the data. All variants of the inputs were consistently sorted into the same shape category as the "parent" logo. Further testing with up to 5% random "salt and pepper" noise was conducted. Again, the variants consistently sorted into the same classes as their parents.

2.2 Logo Identification

The ability to sort logos into arbitrarily chosen geometric classes (not selected based on analytical observations) provides a foundation for a logo processing hierarchy which allows algorithms to be selected based on their usefulness to a specific class or classes.

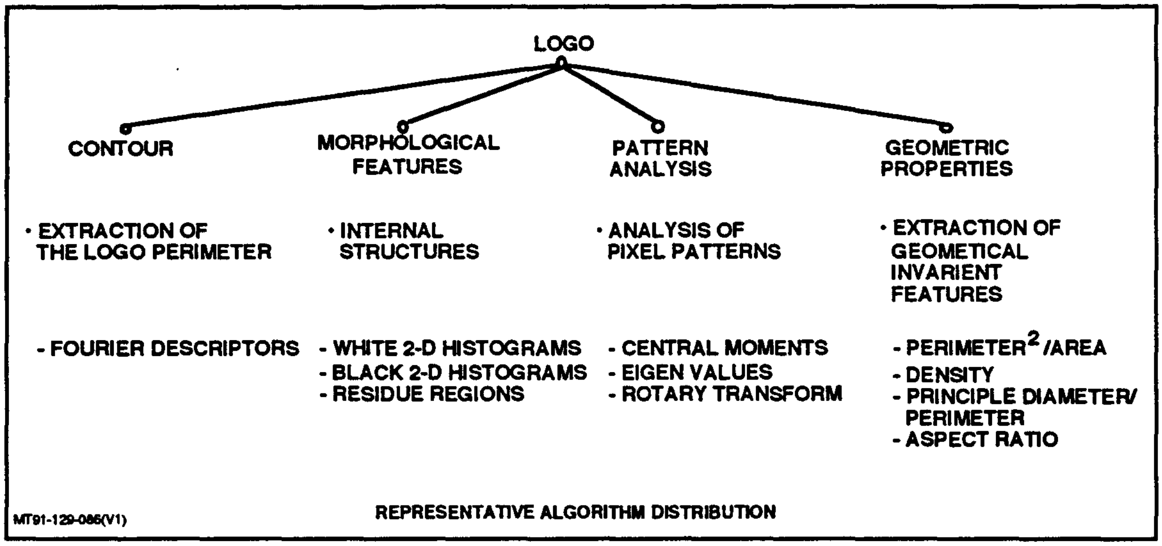


Figure 3: Algorithm Taxonomy

2.2.1 Algorithm Taxonomy

In order to extend the top level sorting concept we identified over 30 algorithms from public domain literature and common image processing techniques. We developed an algorithm taxonomy (Figure 3) so as to select a set of non-redundant algorithms which encompass several types of image analysis. While the taxonomy provided initial guidance on algorithm selection, the final decisions on which algorithm features to keep and which to discard were made empirically.

2.2.2 Feature Identification

A set of analytical features which can be extracted from logo images was selected for evaluation. The primary criterion for selection of candidate features were tolerance for variations in both size and orientation of subject images. We ultimately arrived at a set of 36 features, 34 of which are measured as single values and two sets of vectors used for correlation. A sampling of the features is given below:

density	perimeter squared / area
aspect ratio	principal diagonal
perimeter	residue regions
connected black regions	connected white regions
paradigm correlation	contour sample vector
central moments	rotary sample vector

2.2.3 Feature Space Classifier

Overview

Given a set of characterizing features, the function of the classifier is to determine if any given example of a logo corresponds to any instantiation from a defined set of classes.

Our approach employs a statistical multivariate classification scheme. This requires the existence of a feature-based description of the target logo and a method for classification. Figure 4 shows how the processing is performed. The feature extraction module measures the set of features for a particular logo.

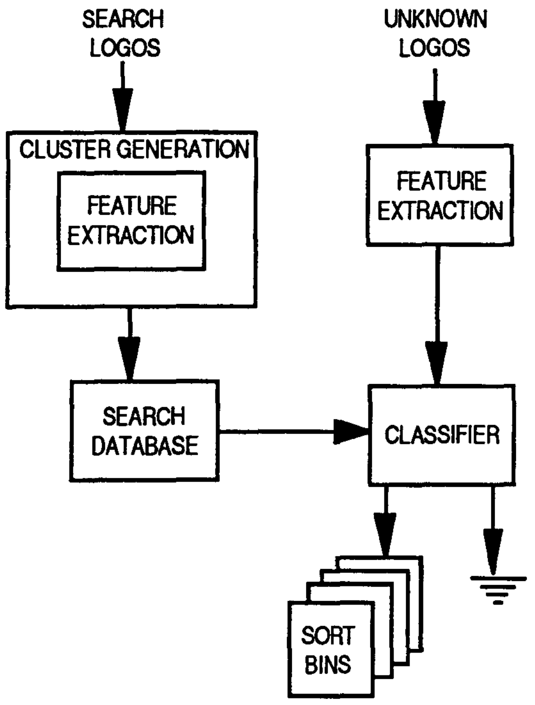


Figure 4: Logo Processing Flow

The cluster generation module generates statistics for a search logo in the n-dimensional feature space. The classifier module calculates the statistical distance between the features of an unknown logo and the statistical model of the archetype and selects the closest matches.

One obvious problem involves generating statistics for a target when only one instance of the logo is available. We addressed this problem by creating multiple instances of the single logo by varying its rotation, varying its scaling, adding noise and adding distortion. Features are extracted from these modified targets, with statistics being calculated from this cluster of measurements.

Feature Selection

A second problem concerns selecting an optimal subset of features which maximize the performance of the classifier. Some quantitative measure of classifier performance is required in order to compare the relative goodness of different combinations of feature and distance metrics.

Two such measures were defined for evaluating the classifier:

- Distance Position Score (DPS)
- Pairwise Fisher Criterion (PFC)

The DPS is a straightforward measure of the ability to correctly identify the Logos of Interest, LOIs, in the database. Calculation of the DPS is as follows. An example of each LOI is given to the classifier as an unknown. For each LOI, the distance ordered list output by the classifier is examined and the position of the matching LOI is recorded. The DPS is the average list position taken across the set of all LOIs.

The PFC indicates the ability of the classifier to reject images which are not represented in its LOI set. PFC is calculated as follows. An example of each LOI is given to the classifier as an unknown and the distance-ordered list obtained. For each LOI, the distance to its correct target and the distance to the nearest incorrect target are recorded. The PFC is thus:

$$\text{PFC} = \frac{\left(\mu_i - \mu_c\right)^2}{\sqrt{\sigma_i^2 + \sigma_c^2}}$$

Because the separation of the correct and incorrect distributions affects classifier false alarm versus missed detection performance, a large value for PFC is desirable.

Classifier performance within our test bed indicates that finding one or more feature sets with very good characteristics is not difficult for a particular set of logos. Furthermore, performance is well behaved around a given point so that most feature selection decisions are are not critical.

2.3.4 Laboratory Test Results

The starting point of this research was the top-level shape sort of logos into a few broad classes. The goal of adding a lower level sorting capability required that we refine these classes into more specific categories.

The ultimate goal of identifying individual logos was met quite successfully. Accuracy rates of 99% with less than 2% false alarms were experimentally achieved using a large set of test data consisting of 1500 images. To illustrate, given an input of 1000 logos, 10 of which are logos of interest, we ran the logo recognition engine against them 10 times. Nine of the ten tests resulted in all 10 logos being found, with the tenth test finding 9. In this tenth test, 20 logos that did not match were incorrectly added to the bin for the logo of interest.

3.0 Logo Segmentation

Segmentation has historically been a major source of difficulty for image processing applications. Repeatability of many downstream algorithms directly depends on the ability of the segmentation process to extract image objects consistently.

3.1 Need for Logo Segmentation

The vast majority of segmentation research and development has been geared toward accommodating a downstream OCR process. As a result, text segmentation has matured to consistently meet the needs of the nominal OCR tasks. However, segmentation for other document image processing tasks, such as engineering drawing conversion (text isolation, dimension set extraction, etc.) is frequently custom-built by or for the specific post-process. Today we see a movement toward some general purpose segmentation techniques for document images with a view to these non-OCR tasks.

Logo segmentation, however is quite a special case and has defied a general algorithmic solution. This is not surprising when one realizes that logos are difficult to parameterize. The logo designer wants to communicate the identification of his or her company or organization at a glance. A good design is understandably unique and subject to whimsical characteristics of style, shape and size. Some logos are quite imaginative, some are rather blasé (logos which are really only BIG **Bold** text). Many logos today contain the names and addresses of their users in close proximity to the logo. Sometimes this is done with standard text, sometimes not.

As a result, standard segmentation techniques tend to have difficulty with logos. This is because decisions are made first as to what the predominant fonts are on the page, then the text is isolated which is readily identifiable. The next step is usually to look for lines and other form-like fields so that they may be removed or extracted for template matching. This is often followed by a search for gray scale candidates - potential photo-like areas which may include a caption. At this point, or some point not much further down the processing path, the unresolvable elements on the page are relegated to simple image zones. These tend to be signatures, logo pieces or artifacts of T-junctions on forms. As logo detection and isolation is not a high priority of most segmentation techniques, the logos presented for post processing tend to be inconsistent from one occurrence to the next. This, of course, makes logo recognition more difficult.

3.2 Approach to Segmentation

Logo segmentation revolves about finding and extracting the logo field from the subject page. Fortunately, users of logos seem to agree to some basic rules on placement and size. In general, the logo occurs in the upper or lower third of the page. Furthermorek most logos are graphical in nature, and if designed well, it will stand out stylistically from the rest of the page elements.

The phrases "graphical in nature" and "stand out from the rest of the page elements" are subjective ideas. They do not lend themselves to mathematical representation. It is for this reason that we chose to investigate heuristic approaches to logo segmentation.

3.3 Segmentation Flow

The Segmentation process consists of these main steps:

- pre-deskew elimination
- detection of skew and orientation
- noise elimination
- threshold desample
- *initial parts elimination*
- connected component generation
- part size calculation
- *secondary parts elimination*
- final connected component generation
- *selective token extraction*

Most of these steps use well-publicized techniques, so we will only elaborate on the steps which we believe make a unique contribution to the success of our segmentation work.

3.3.1 Initial Parts Elimination

Once the image has been deskewed, desampled and cleaned up, we set out to eliminate those items on the page which are most likely NOT logos. This includes text blocks, long horizontal and vertical lines and pictures. We visit each connected component on the page and evaluate against four heuristic criteria:

- measures only 1 pixel in any dimension
- length or width exceeds 4 inches
- total area exceeds 4 square inches
- aspect exceeds MAX_ASPECT (as defined in the setup file)

After completing this elimination step, we proceed to generate connected components from the remaining page elements with a simple horizontal RLS algorithm.

3.3.2 Secondary Parts Elimination

Once components have been generated, we compute the *size* of each component on the page in terms of pixel count, width and length. Elimination of parts is then performed utilizing these measurements and three additional heuristics to the subject token (connected component after RLS):

- length is greater than twice the average part length
- width is greater than 3/5 of the page
- perimeter exceeds it's area

These heuristics tend to eliminate single text lines, text blocks and large graphics (photos) from the document page.

3.3.3 Selective Token Extraction

The final step before token extraction is to perform simple vertical RLS in order to reconnect the remaining pieces of logos.

The token extraction process is selective in that as it arrives at each token on the page, it examines the extents of the token from the original image. If the token is too small (we observed that in spite of the elimination steps taken before, we often had very small artifacts which survived), it is simply not extracted.

Token extraction is simply a function of back referencing the original image to retrieve the token. Tokens are output as a linked list. The page is traversed from upper left to lower right.

3.4 Laboratory Test Results

Our test data consisted of 300 document images scanned at 200 dpi and stored as TIFF files. These images ran the gamut from standard business memoranda to purchase orders and government documents. The content was English in origin, with Roman text being the predominant text form used.

Each of the 300 documents in the set contained 1 or more logos. The segmentor was able to reliably extract the logo from the page 85% of the time. Identical logos which appeared in different locations on different pages, and often in different sizes, were largely extracted in the same form. Subsequent logo recognition led to accurate classification of those logos within the data set.

4.0 Proof-of-Concept System

Based on the success of this research, we integrated the various elements into an end-to-end capability which could be functionally tested on Government furnished data. To this end, we modularized many of the algorithms and standardized the interfaces in a fashion that lent itself to integration with other components.

The functional block diagram of the proof-of-concept system is shown in Figure 5. The system is intended to demonstrate an end-to-end capability for logo recognition. The fundamental input to the system is the document page image. This image is expected to be in TIFF or other standard file formats. The Segmentor, as discussed earlier, is designed to isolate potential logo candidates on a page. The output of this process is a stream of tokens. These token structures are complete with bounding box coordinates from the original image, the token image itself (in bitmap format), the name of the original input image and a pointer to the next token in the structure.

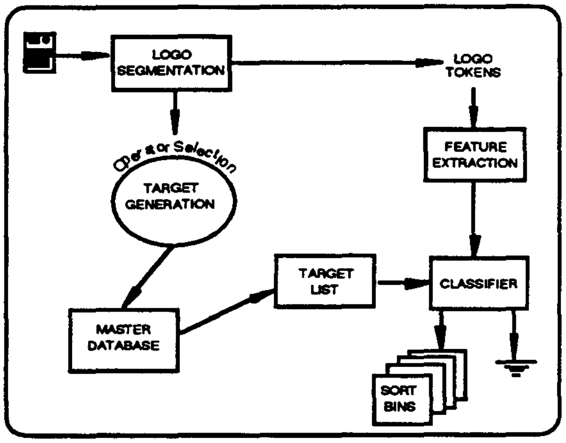


Figure 5: Proof-of-Concept Logo Recognition

The segmentation process feeds the feature extraction process. For each of the tokens, the feature vector is calculated and stored in a record. The format of the record accommodates both the single value features and the vectors described earlier, and is therefore quite large.

Note that target generation is an operator assisted step. The operator must identify those logos which are of interest to him or her. Once identified, the feature records for these logos (including the data from the statistical perturbations) go into the master database. Classification is done against a subset of these records. This subset is described by the Target List. Multiple target lists will allow users to customize their logo recognition requirements as needed.

Once the record for a token has been created, it is ready for classification against a database of target records, as discussed previously. The final output of the system comes from the Classification step. The output is simply a correlation list showing how strongly the features of the unknown correlated to those targets against which it was compared. As the classification process may be thresholded in various ways, those unknowns which fall outside the specified tolerances for correlation will be shown as a no match. This correlation list essentially provides for binning of the unknowns into distinct categories.

4.2 Preliminary Test Results

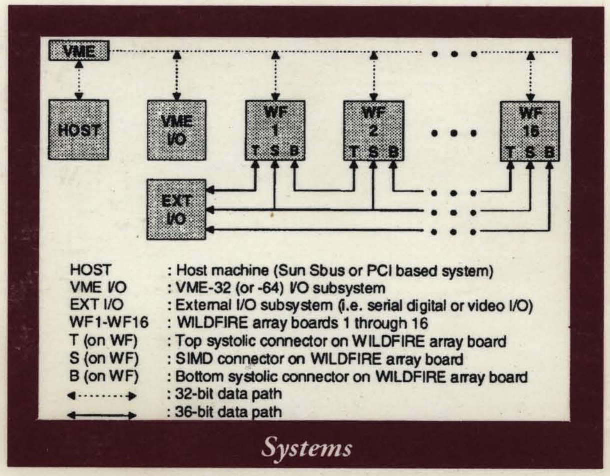
Testing of the Proof-of-Concept system in the laboratory yielded results which were consistent with the previous results of both logo recognition and segmentation. Field testing is being conducted as this paper is being published, and some preliminary results should be available in late October.

5.0 Conclusion and Future Work

We have developed a robust logo extraction and recognition capability which appears to perform well even on poor quality image data. The proof-of-concept system is currently under evaluation in government laboratories. Extensive data sets are being prepared which will provide comprehensive measures of performance for the various system components. This field testing is expected to reveal several recommendations for tuning and customization, as well as pointing out the needs for optimization in algorithm performance.

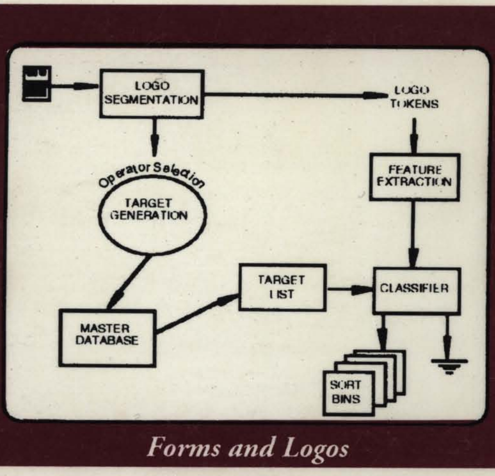
6.0 Acknowledgements

The author wishes to acknowledge the significant contributions of Dr. H. Gary Greene and Mr. Patrick Conroy to this research and development effort.



HOST	: Host machine (Sun Sbus or PCI based system)
VME I/O	: VME-32 (or -64) I/O subsystem
EXT I/O	: External I/O subsystem (i.e. serial digital or video I/O)
WF1-WF16	: WILDFIRE array boards 1 through 16
T (on WF)	: Top systolic connector on WILDFIRE array board
S (on WF)	: SIMD connector on WILDFIRE array board
B (on WF)	: Bottom systolic connector on WILDFIRE array board
◄·······►	: 32-bit data path
◄────►	: 36-bit data path

Systems



Forms and Logos