LAMP-TR-010
CFAR-TR-856
CS-TR-3786

April 1997

Document Image Compression and Analysis

O. Kia

Language and Media Processing Labratory
Instititue for Advanced Computer Studies
College Park, MD 20742

Abstract

Image compression usually considers the minimization of storage space as its main objective. It is desirable, however, to code images so that we have the ability to process the resulting representation directly. In this thesis we explore an approach to document image compression that is efficient in both space (storage requirement) and time (processing flexibility). A representation is presented in which component-level redundancy is removed by forming a prototype library and component location table. This representation forms a basis for compression and provides direct access to image components. To generate the prototype library, a new clustering approach is developed which is suitable for document image components. The distance metric is based on a character degradation model so that degraded versions of the same character will be grouped together. To achieve a lossless representation when required, the residuals are encoded efficiently using a structural distance ordering. OCR is then used as a measure of readability to evaluate the rate distortion tradeoff for lossy compression. A set of algorithms is presented for typical document processing applications which operate effectively on the compressed representation. Applications demonstrated include subdocument retrieval, skew estimation

***The support of the LAMP Technical Report Series and the partial support of this research by the National Science Foundation under grant EIA0130422 and the Department of Defense under contract MDA9049-C6-1250 is gratefully acknowledged.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **APR 1997**	2. REPORT TYPE	3. DATES COVERED **00-04-1997 to 00-04-1997**

4. TITLE AND SUBTITLE
Document Image Compression and Analysis

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Language and Media Processing Laboratory,Institute for Advanced Computer Studies,University of Maryland,College Park,MD,20742-3275

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**141**	

Abstract

Title of Dissertation: *Your Dissertation Title*

Your Full Name, Doctor of Philosophy, 1997

Dissertation directed by: *Academic title and name of advisor*
Department of Mathematics

Image compression usually considers the minimization of storage space as its main objective. It is desirable, however, to code images so that we have the ability to process the resulting representation directly. In this thesis we explore an approach to document image compression that is efficient in both space (storage requirement) and time (processing flexibility).

A representation is presented in which component-level redundancy is removed by forming a prototype library and component location table. This representation forms a basis for compression and provides direct access to image components.

To generate the prototype library, a new clustering approach is developed which is suitable for document image components. The distance metric is based on a character degradation model so that degraded versions of the same character will be grouped together. To achieve a lossless representation when required, the residuals are encoded efficiently using a structural distance ordering. OCR is then used as a measure of readability to evaluate the rate distortion tradeoff for lossy compression.

A set of algorithms is presented for typical document processing applications which operate effectively on the compressed representation. Applications demonstrated include subdocument retrieval, skew estimation, keyword search and document image matching.

Extensions of the paradigm to grayscale and graphic document images, networking and multimedia objects are discussed.

Keywords: compression, coding, representation, compressed-domain processing

Your Dissertation Title

by

Your Full Name

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland at College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
1997

Advisory Committee:

Academic title and name of advisor, Chairman/Advisor
Academic title and name of members

Dedication

Farangis Sarraf Kia
and loved ones who were and will be ...

Acknowledgements

I would like to thank all the committee members for their time, input and diligence. I especially would like to thank Professor Chellappa for his time and advice. I would like to thank Professor Rosenfeld for an untiring appetite for perfection. I truly value his critiques and thoughtful suggestions over a large number of meetings. I would especially like to thank Doctor Doermann for an unending supply of ideas, patience, and encouragement. I greatly value his friendship and have great respect for his contributions to the scientific community.

I would like to take this opportunity to thank Professor Jafar Vossoughi. Without his help, advice, and encouragement during my undergraduate years, none of this would have ever been possible. I truly owe all I have become to him.

Most importantly, I would like to thank my parents, Ebrahim and Farideh, for bringing me to this earth, and my wife, Farangis, who makes living on it bearable through her patience, hard work, and love. I cannot imagine a world without Farangis and always pray to God to keep us together. I dedicate this dissertation to Farangis Sarraf Kia and all those who have touched me emotionally and spiritually.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Technological advances in processing, storage, and visualization have made it possible to maintain large numbers of documents in digital image form and make them accessible over networks. In order to do this effectively, three primary concerns must be addressed. The first is document size. An ASCII representation of a document page can easily be stored in 2-3 KB, whereas a typical scanned image of a page may require between 500 KB and 2 MB. If we are to maintain documents in image form, an efficient compressed representation is essential for both storage and transmission.

The second concern is providing efficient access to the compressed image. Traditional compression techniques used for document images have been successful in reducing storage requirements but do not provide efficient access to the compressed data. It is desirable to use a compression method that makes use of a structured representation of the data, so that it not only allows for rapid transmission but also allows access to various document components and facilitates processing of documents without the need for expensive decompression.

The third concern is that of readability. Many lossy compression and progressive transmission techniques use resolution reduction or texture-preserving methods that can render a document image unreadable. It is desirable that a document be readable even at the highest levels of lossy compression and at the start of a progressive transmission. The highly lossy representation can then be augmented by subsequent transmissions for better rendition. This is preferred to a scenario in which the highest resolution is the only readable resolution. In this thesis, we will address these concerns for scanned images of machine-printed documents.

The task of compression, organization, and transmission of information is not new to the image and video domains. A number of techniques have been developed to address the problems of image quality and compressed-domain access but the solutions have had very little general application because of the domain-specific requirements. Likewise, document images have specific characteristics and requirements which require special attention when considering approaches to compression. Two general questions which arise immediately are:

1. What are the characteristics of a document image which can be exploited to benefit compression and compressed-domain processing?

2. What has been done before that can be extended to the document image domain?

We will address these questions in the next subsections by reviewing standard compression techniques, examining document image characteristics, and reviewing work done on document image compression.

1.1 Image compression

The initial breakthroughs in the compression of one-dimensional signals [109] were easily extended to the image domain by concatenating image rows or columns into a single stream. Techniques such as Shannon-Fano coding [95] and Huffman coding [26, 29, 55, 106, 118, 119] use redundancy-reduction mechanisms which result in shorter codes for more frequently appearing samples. It is necessary to scan the data samples in order to determine their probabilities of occurrence and create an appropriate code. Adaptive variations of these techniques initially assume equal probability for all samples and calculate subsequent probability measures based on a fixed window length prior to the sample of interest. This allows local changes in probability measurements and achieves higher global compression. Run-length coding [38] is another redundancy-reduction coding method where in a scan-line each run of symbols is coded as a pair that specifies the symbol and the length of the run.

While most redundancy-reduction methods are lossless, other arbitrarily lossy coding methods have achieved higher levels of compression. Transform coding [44], subband coding [61, 77, 105, 114], vector quantization [61, 62, 66], and predictive coding [1, 3, 22, 31, 46, 57, 59, 76, 83, 97] are among the ones that have achieved high levels of lossy compression.

The major transform coding techniques include cosine/sine [2, 44], Fourier [5, 44], Hadamard [27, 44], Haar [27, 44], slant [44], and principal-component (Karhunen-Loeve) transforms [6, 44]. All transform coding based compression algorithms are pixel-based approaches which decompose a signal unto an orthonormal basis to achieve energy compaction. Lossy compression is then achieved by coding the high energy components and leaving out the low energy ones. While some transformations such as the Hadamard and Haar transforms can be performed relatively quickly, other transforms are computationally intensive and either require dedicated hardware or restrictions such as limiting the size of the transform to a power of two.

In subband coding, as the name implies, different frequency bands are treated differently to achieve higher fidelity or higher compression. As with transform coding, the signal is transformed to an alternate domain, but instead of leaving out a component, it is sampled at a lower resolution while more important components are quantized at higher resolution. In effect, at each subband the application of a different quantization strategy is used to preserve fidelity and achieve compression.

Vector quantization is a hybrid of statistical analysis and pattern recognition with a large number of well-established techniques. This method first divides an image into blocks which are then serialized into large-dimensional vectors. Each vector is treated as a sample in a high-dimensional space and this space is partitioned into subspaces, called classes. For each class of vectors, a prototype is created that closely resembles the vectors which belong to the class. Vector quantization shifts the partition boundaries by calculating a distance measure. Errors between in-class vectors and the corresponding prototypes over all classes are minimized to provide optimum prototype placement. High levels of compression are achieved by coding every vector that belongs to a given class by a reference to the prototype vector. The level of lossiness is then dependent on the distance measure and the number of prototype vectors.

Predictive coding, such as the differential pulse code modulation (DPCM) patented by Cutler [22], predicts the next sample based on the previously coded samples and codes the error between the prediction and the actual sample. The main objective of DPCM is to narrow down the range of coded samples. However, images are not stationary in texture, and some predictors may work in certain areas, but not in others. Hence, the concept of adaptive predictors [3, 31, 83] was introduced so that the coder and decoder could choose from a set of predictors at the cost of transmitting the parameters needed for the prediction. This method of coding needs to limit prediction error for efficient coding and achieves lossless compression in coding the entire error; lossy coding is achieved by limiting the range of the error value.

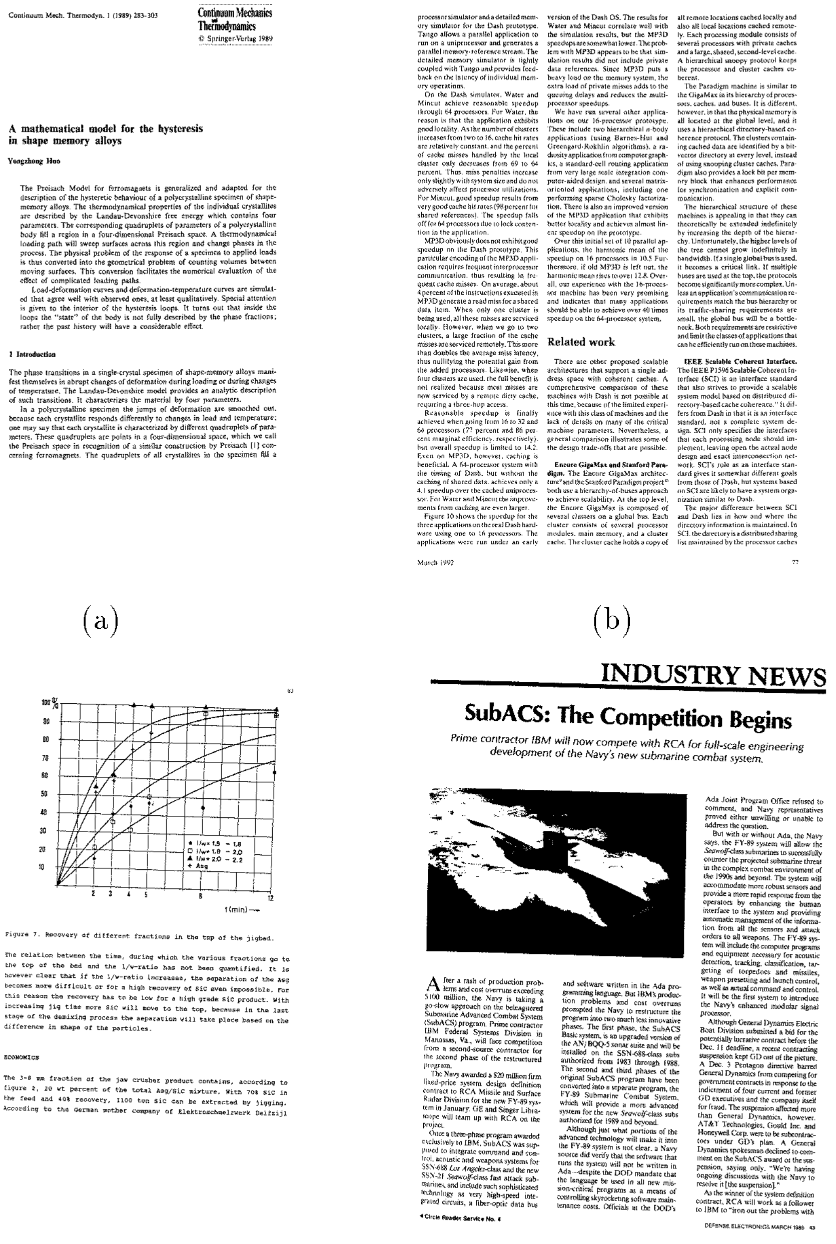
More recently, second-generation compression algorithms based on human visual behavior [68] have been proposed which have the potential for much higher compression ratios. Also, developments in user-specified wavelet transforms [18, 68] can be used to address compression requirements for specific image types. Fractal coding [117], which is also called chaos coding, is an example of progress in the area of two-dimensional redundancy reduction. All of these compression techniques are pixel-based and they perform best for textured images.

1.2 Document characteristics

Document images are scans of documents which are in most cases pseudo-binary and rich in textual content. Informally, we can define document images as images that contain components that resemble the symbols of a language. Many documents look like those shown in Figure 1.1 and are represented adequately by binary images produced by scanning at a resolution of 300 dots per inch (dpi). They tend to be highly structured in terms of layout, and have significant redundancy in the symbols which occur in them.

Since a document can be used in a large number of ways, an important consideration is how the image is affected by compression. For example, if we intend that the document ultimately be read by humans, it is necessary for compression schemes to preserve the shapes of the components so that they are recognizable by the reader after retrieval. If the document must be reproduced in near-original form, techniques which reduce resolution may not be acceptable. Thus the required resolution is dependent on the task; some resolutions may leave the document readable, but may destroy fine detail.

As is well known, the performance of conventional compression algorithms (such as those based on transform coding [44], vector quantization [30], fractal compression



Continuum Mech. Thermodyn. 1 (1989) 283-303

Continuum Mechanics and Thermodynamics
© Springer-Verlag 1989

A mathematical model for the hysteresis in shape memory alloys

Yongzhong Huo

The Preisach Model for ferromagnets is generalized and adapted for the description of the hysteretic behaviour of a polycrystalline specimen of shape-memory alloys. The thermodynamical properties of the individual crystallites are described by the Landau-Devonshire free energy which contains four parameters. The corresponding quadruplets of parameters of a polycrystalline body fill a region in a four-dimensional Preisach space. A thermodynamical loading path will sweep surfaces across this region and change phases in the process. The physical problem of the response of a specimen to applied loads is thus converted into the geometrical problem of counting volumes between moving surfaces. This conversion facilitates the numerical evaluation of the effect of complicated loading paths.

Load-deformation curves and deformation-temperature curves are simulated that agree well with observed ones, at least qualitatively. Special attention is given to the interior of the hysteresis loops. It turns out that inside the loops the "state" of the body is not fully described by the phase fractions; rather the past history will have a considerable effect.

1 Introduction

The phase transitions in a single-crystal specimen of shape-memory alloys manifest themselves in abrupt changes of deformation during loading or during changes of temperature. The Landau-Devonshire model provides an analytic description of such transitions. It characterizes the material by four parameters.

In a polycrystalline specimen the jumps of deformation are smoothed out, because each crystallite responds differently to changes in load and temperature; one may say that each crystallite is characterized by different quadruplets of parameters. These quadruplets are points in a four-dimensional space, which we call the Preisach space in recognition of a similar construction by Preisach [1] concerning ferromagnets. The quadruplets of all crystallites in the specimen fill a

(a)

processor simulator and a detailed memory simulator for the Dash prototype. Tango allows a parallel application to run on a uniprocessor and generates a parallel memory-reference stream. The detailed memory simulator is tightly coupled with Tango and provides feedback on the latency of individual memory operations.

On the Dash simulator, Water and Mincut achieve reasonable speedup through 64 processors. For Water, the reason is that the application exhibits good locality. As the number of clusters increases from two to 16, cache hit rates are relatively constant, and the percent of cache misses handled by the local cluster only decreases from 69 to 64 percent. Thus, miss penalties increase only slightly with system size and do not adversely affect processor utilizations. For Mincut, good speedup results from very good cache hit rates (98 percent for shared references). The speedup falls off for 64 processors due to lock contention in the application.

MP3D obviously does not exhibit good speedup on the Dash prototype. This particular encoding of the MP3D application requires frequent interprocessor communication, thus resulting in frequent cache misses. On average, about 4 percent of the instructions executed in MP3D generate a read miss for a shared data item. When only one cluster is being used, all these misses are serviced locally. However, when we go to two clusters, a large fraction of the cache misses are serviced remotely. This more than doubles the average miss latency, thus nullifying the potential gain from the added processors. Likewise, when four clusters are used, the full benefit is not realized because most misses are now serviced by a remote dirty cache, requiring a three-hop access.

Reasonable speedup is finally achieved when going from 16 to 32 and 64 processors (77 percent and 86 percent marginal efficiency, respectively), but overall speedup is limited to 14.2. Even on MP3D, however, caching is beneficial. A 64-processor system with the timing of Dash, but without the caching of shared data, achieves only a 4.1 speedup over the cached uniprocessor. For Water and Mincut the improvements from caching are even larger.

Figure 10 shows the speedup for the three applications on the real Dash hardware using one to 16 processors. The applications were run under an early version of the Dash OS. The results for Water and Mincut correlate well with the simulation results, but the MP3D speedups are somewhat lower. The problem with MP3D appears to be that simulation results did not include private data references. Since MP3D puts a heavy load on the memory system, the extra load of private misses adds to the queuing delays and reduces the multiprocessor speedups.

We have run several other applications on our 16-processor prototype. These include two hierarchical n-body applications (using Barnes-Hut and Greengard-Rokhlin algorithms), a radiosity application from computer graphics, a standard-cell routing application from very large scale integration computer-aided design, and several matrix-oriented applications, including one performing sparse Cholesky factorization. There is also an improved version of the MP3D application that exhibits better locality and achieves almost linear speedup on the prototype.

Over this initial set of 10 parallel applications, the harmonic mean of the speedup on 16 processors in 10.5 Furthermore, if old MP3D is left out, the harmonic mean rises to over 12.8. Overall, our experience with the 16-processor machine has been very promising and indicates that many applications should be able to achieve over 40 times speedup on the 64-processor system.

Related work

There are other proposed scalable architectures that support a single address space with coherent caches. A comprehensive comparison of these machines with Dash is not possible at this time, because of the limited experience with this class of machines and the lack of details on many of the critical machine parameters. Nevertheless, a general comparison illustrates some of the design trade-offs that are possible.

Encore GigaMax and Stanford Paradigm. The Encore GigaMax architecture[9] and the Stanford Paradigm project[10] both use a hierarchy-of-buses approach to achieve scalability. At the top level, the Encore GigaMax is composed of several clusters on a global bus. Each cluster consists of several processor modules, main memory, and a cluster cache. The cluster cache holds a copy of all remote locations cached locally and also all local locations cached remotely. Each processing module consists of several processors with private caches and a large, shared, second-level cache. A hierarchical snoopy protocol keeps the processor and cluster caches coherent.

The Paradigm machine is similar to the GigaMax in its hierarchy of processors, caches, and buses. It is different, however, in that the physical memory is all located at the global level, and it uses a hierarchical directory-based coherence protocol. The clusters containing cached data are identified by a bit-vector directory at every level, instead of using snooping cluster caches. Paradigm also provides a lock bit per memory block that enhances performance for synchronization and explicit communication.

The hierarchical structure of these machines is appealing in that they can theoretically be extended indefinitely by increasing the depth of the hierarchy. Unfortunately, the higher levels of the tree cannot grow indefinitely in bandwidth. If a single global bus is used, it becomes a critical link. If multiple buses are used at the top, the protocols become significantly more complex. Unless an application's communication requirements match the bus hierarchy or its traffic-sharing requirements are small, the global bus will be a bottleneck. Both requirements are restrictive and limit the classes of applications that can be efficiently run on these machines.

IEEE Scalable Coherent Interface. The IEEE P1596 Scalable Coherent Interface (SCI) is an interface standard that also strives to provide a scalable system model based on distributed directory-based cache coherence.[11] It differs from Dash in that it is an interface standard, not a complete system design. SCI only specifies the interfaces that each processing node should implement, leaving open the actual node design and exact interconnection network. SCI's role as an interface standard gives it somewhat different goals from those of Dash, but systems based on SCI are likely to have a system organization similar to Dash.

The major difference between SCI and Dash lies in how and where the directory information is maintained. In SCI, the directory is a distributed sharing list maintained by the processor caches

March 1992

77

(b)

Figure 7. Recovery of different fractions in the top of the jigbed.

The relation between the time, during which the various fractions go to the top of the bed and the l/w-ratio has not been quantified. It is however clear that if the l/w-ratio increases, the separation of the Asg becomes more difficult or for a high recovery of SiC even impossible. For this reason the recovery has to be low for a high grade SiC product. With increasing jig time more SiC will move to the top, because in the last stage of the demixing process the separation will take place based on the difference in shape of the particles.

ECONOMICS

The 3-8 mm fraction of the jaw crusher product contains, according to figure 2, 20 wt percent of the total Asg/SiC mixture. With 70% SiC in the feed and 40% recovery, 1100 ton SiC can be extracted by jigging. According to the German mother company of Elektroschmelzwerk Delfzijl

(c)

INDUSTRY NEWS

SubACS: The Competition Begins

Prime contractor IBM will now compete with RCA for full-scale engineering development of the Navy's new submarine combat system.

After a rash of production problems and cost overruns exceeding $100 million, the Navy is taking a go-slow approach on the beleaguered Submarine Advanced Combat System (SubACS) program. Prime contractor IBM Federal Systems Division in Manassas, Va., will face competition from a second-source contractor for the second phase of the restructured program.

The Navy awarded a $20 million firm fixed-price system design definition contract to RCA Missile and Surface Radar Division for the new FY-89 system in January. GE and Singer Librascope will team up with RCA on the project.

Once a three-phase program awarded exclusively to IBM, SubACS was supposed to integrate command and control, acoustic and weapons systems for SSN-688 *Los Angeles*-class and the new SSN-21 *Seawolf*-class fast attack submarines, and include such sophisticated technology as very high-speed integrated circuits, a fiber-optic data bus and software written in the Ada programming language. But IBM's production problems and cost overruns prompted the Navy to restructure the program into two much less innovative phases. The first phase, the SubACS Basic system, is an upgraded version of the AN/BQQ-5 sonar suite and will be installed on the SSN-688-class subs authorized from 1983 through 1988. The second and third phases of the original SubACS program have been converted into a separate program, the FY-89 Submarine Combat System, which will provide a more advanced system for the new *Seawolf*-class subs authorized for 1989 and beyond.

Although just what portions of the advanced technology will make it into the FY-89 system is not clear, a Navy source did verify that the software that runs the system will not be written in Ada—despite the DOD mandate that the language be used in all new mission-critical programs as a means of controlling skyrocketing software maintenance costs. Officials at the DOD's Ada Joint Program Office refused to comment, and Navy representatives proved either unwilling or unable to address the question.

But with or without Ada, the Navy says, the FY-89 system will allow the *Seawolf*-class submarines to successfully counter the projected submarine threat in the complex combat environment of the 1990s and beyond. The system will accommodate more robust sensors and provide a more rapid response from the operators by enhancing the human interface to the system and providing automatic management of the information from all the sensors and attack orders to all weapons. The FY-89 system will include the computer programs and equipment necessary for acoustic detection, tracking, classification, targeting of torpedoes and missiles, weapon presetting and launch control, as well as actual command and control. It will be the first system to introduce the Navy's enhanced modular signal processor.

Although General Dynamics Electric Boat Division submitted a bid for the potentially lucrative contract before the Dec. 11 deadline, a recent contracting suspension kept GD out of the picture. A Dec. 3 Pentagon directive barred General Dynamics from competing for government contracts in response to the indictment of four current and former GD executives and the company itself for fraud. The suspension affected more than General Dynamics, however. AT&T Technologies, Gould Inc. and Honeywell Corp. were to be subcontractors under GD's plan. A General Dynamics spokesman declined to comment on the SubACS award or the suspension, saying only, "We're having ongoing discussions with the Navy to resolve it [the suspension]."

As the winner of the system definition contract, RCA will work as a follower to IBM to "iron out the problems with

◂Circle Reader Service No. 4

DEFENSE ELECTRONICS MARCH 1986 43

(d)

Figure 1.1: Examples of binary document images of the types usually found in a document image database. These are images from the University of Washington document database [32], cropped (automatically) to the main body of the text.

[12], pattern matching and substitution [10, 64], and other approaches) depends on the types of images being compressed, and on their texture and content characteristics. Algorithms often do well on some classes of images and not so well on others. Since document images differ significantly from scene images, it is reasonable to assume they will benefit from a specialized compression scheme.

1.3 Previous work on document image compression

Document image compression was first recognized as a special case of compression in the 1970s when increased use of facsimile machines promised potential growth in document scanning and transmission. The international communication standards body then set out to create a standard for use in facsimile transmission. In 1980 the first comprehensive standard was released as the CCITT Group 3 transmission standard [41], followed by CCITT Group 4 [15, 36, 85]. Both standards were originally designed for digital transmission rather than for analog phone lines, but nevertheless have become standards for facsimile transmission. Both schemes use a run-length-based approach where the white and black runs are pre-coded in the compression algorithm. An extension to dual scan-line compression exploits coherence between successive scan lines by inserting references to the previous scan-line's black runs. In Group 4, higher compression is achieved, largely due to the omission of various error-correcting codes which were intentionally included in the Group 3 standard. These methods are clearly sufficient for transmission of facsimile since a facsimile scanner sends and receives in scan-line order.

Joint Binary Image experts Group (JBIG) is a recent CCITT standard. JBIG performs compression on binary images and provides a lossy representation suited for compression and transmission. Context modeling forms the basis of its coding [81, 82]. At the lowest levels, JBIG uses template models and adaptive arithmetic coding to encode predictions. "Layers" of the image are represented at reduced resolutions to provide progressive transmission. Exception pixels are used to preserve overall image layout and structure; this works well for textured dithered images, but loses fidelity when small-scale structure is essential, as in the case of text documents. JBIG, despite its use of context-based coding, is unable to organize information to be used for compressed-domain processing. This is mostly due to its use of arithmetic coding directly on the pixels. The output of the arithmetic coder is highly complex and cannot be easily deciphered for compressed-domain processing. The only favorable characteristic of JBIG is its use of intermediate representations for lossy compression and progressive transmission.

Recent work by Howard [37] performs context-based arithmetic coding on the bit map image based on predictive coding of "clairvoyant" [37, 112] pixels and previously coded pixels. Clairvoyant pixels are pixels belonging to prototype images; in Howard's coding scheme they contribute to knowledge of the current pixel's distribution which can be used for better coding. This is still a pixel-based approach and does not provide for compressed-domain processing. While Howard proposes a methodology for scalable lossy compression, his approach lacks a hierarchical representation suitable for progressive transmission.

An important approach to document compression, suggested by Ascher and Nagy [9] and later formalized by Witten et al. [43, 110, 111, 112], is based on the repetitive nature of text components. In a departure from pixel-level to symbol level coding, marks found within a document are coded. The method is very similar to pattern-matching and substitution algorithms [10, 47, 64, 69, 70, 85, 113]. It is based on the measurement of similarity between symbols within an image. In this way redundancies in symbol shapes are identified, resulting in two-dimensional redundancy reduction versus the traditional one-dimensional reduction.

Ascher and Nagy's approach considered a stream of bitmap images which represented individual characters. They proposed an algorithm which creates a dynamic library; an uncataloged character gets added to the library and a character that has already been seen gets replaced by an index to a library item. The library is a collection of bitmap images. By keeping the library's size limited, an adaptive symbol compression method is achieved; changes in the symbol stream are compensated by creating new library prototypes and throwing out old and unused ones.

Witten et al. expanded on this approach to address extraction of image components, indexing their locations within the page and compressing those indices, and coding the residuals left after inexact replacement of components by library prototypes. This work made contributions to prototype generation, symbol index compression, and residual coding. Connected component analysis is performed on each bitmap image. The components are then clustered by a symbol matching method such as Pratt's combined symbol matching [85], Holt and Xydeas' "weighted and-not" method [36], Holt's combined size-independent strategy [35], and Johnsen's generic pattern matching and substitution [47]. Another method of matching, called compression-based template matching [43] was suggested by Inglis and Witten and was later improved upon by Zhang and Danskin [116]; it was based on mutual information between the component and the prototype. The locations of the components are ordered in natural reading order, indexed with respect to each other (a component is referenced by its upper left corner with respect to the bottom right corner of the previous component), and compressed. The residual map is kept for lossless compression since clustering generalizes the input patterns and there exist differences between the components and prototypes which may affect readability. The residual map is not compressed, but the original image is compressed; the image composed of prototypes is used as context in the context-based arithmetic coding.

This method satisfies our concerns to a large extent but falls short in some areas. Most of the shortfalls lie in the processing domain. The first drawback is that direct access to the components is lost, or at least difficult, when differential indexing is used. The second drawback is the lack of hierarchical coding to allow for a scalable intermediate representation. Only two levels of representation are available (with and without residuals), which does not make this method suitable for lossy compression and progressive transmission. The third drawback is the failure to address readability. They did not utilize a noise model to estimate image quality and derive a scalable representation. They were concerned only with compression, which they achieved rather well.

1.4 Compressed-domain processing of document images

Work on skew estimation like that of Baird [11] has been shown to be efficient and accurate. However, detection of alignment feature locations is computationally demanding since the volume of data is large. It would be extremely desirable if the detection of alignment features could be done in the compressed domain. In Spitz's work [100], the pass codes in CCITT Group 4 compressed images were used for Baird's alignment features (called fiducial points) to estimate a skew angle. Pass codes are part the Group 4 specification which mark the locations where the previous scan line has a run of black pixels and the current scan line does not. The pass codes are heavily populated at the bottoms of characters and provide a basis for skew estimation. The occasional pass codes generated by descenders, dots, etc. do not severely degrade the performance of the skew estimation. However, skew correction is a non-trivial task with a run-length encoding based algorithm.

Scanning and recognizing barcodes using hand-held scanners has been quite successful, but the problem of recognizing barcodes in scanned images has received little attention. In the work done by Maa [65], detection of barcodes is performed on CCITT Group 4 representations. The basis for Maa's work is that since bars and spaces are upright and perfectly aligned, Group 4 compression will use vertical coding so as to reduce vertical redundancy. Thus in all scan lines except the first, every run of bars and spaces will be represented by the same code. This will result in runs of this code at image locations where there exist barcodes with no interfering patterns. With added noise, only one or two other codes were usually found in these runs and they were still usable for detection of barcodes. An 86% performance rate was reported.

For purposes of document image retrieval, it is desirable to index on distinctive image characteristics. One class of such characteristics which appear in document images are logos. Logos are defined by Spitz [101] as spatial arrangements of structures. He used the fiducial points that he had applied to skew estimation, and considered them to be a series of points locally aligned to specific signatures. A signature representative of a logo prototype can be determined and compared with signatures extracted from the image. Localizing the logo was not considered, but this is an easy problem since a prototype logo is of a known size and the signature determined from the image should only depend on that size. Experiments on a small set of images were reported to result in very high recognition rates.

Recently, document image matching has been considered by a number of researchers [89, 96, 115]. Recent work by Hull [40] addresses this problem by use of compressed-domain processing. As in Spitz's work on skew estimation and logo detection, Hull considers the locations of pass codes as feature points and compares two sets of feature points for a possible match. These feature points are contained in a bounding box, and in Hull's work he determines the best choice of a bounding box to yield a high match score. Given two sets of feature points, Hull used the Hausdorff distance [42], which measures the amount of perturbation needed to overlap pairs of points in a pair of images, and reported a high detection rate. Efficient measurement of the Hausdorff distance has been studied by Rucklidge [88].

It would be desirable to develop a document compression method that can address a larger amount of compressed-domain processing. In doing this, we can take advantage of the fact that the features used in such tasks as skew estimation, correction, and matching are dependent on the document image symbols, so that a method allowing access to these symbols should facilitate document image processing.

1.5 Symbolic compression

Symbolic compression is a method of coding similarly-shaped marks in an image with reference to a set of templates. In doing so, individual marks must first be segmented out. For binary document images, segmentation is typically a matter of determining connected components of black pixels in the image. Perfect segmentation is not necessary since the residual bits which remain after replacing a component with a template are also coded. The segmented components are clustered to capture their redundancies. Various methods of clustering, such as simple exclusive-or and compression-based template matching [35, 112], can be used. Once all symbols within the image have been considered for clustering, the cluster centers can serve as representative templates or prototypes for the clusters. Some clusters may need to be removed because of inadequate membership, and some clusters may exhibit large in-class variances which result in abnormally large residuals. The prototypes are stored in a library. The locations of the symbols must be recorded so that a highly compact representation, based on prototypes only, is immediately available. The difference between this symbolic representation and the original image is a residual image which has far fewer foreground pixels than before; this implies a high compression ratio. Context-predictive coding of the residual image further improves the compression ratio.

Residual coding is the portion of the compression algorithm which encodes the residuals. To preserve symbol access for compressed-domain analysis [50, 53], it is necessary to code the error on a per-symbol basis rather than coding it for the entire image. Using a structural model, it is possible to order the error pixels based on their structural importance. The structural order, derived in part from the work of Kanungo et al. [48], is based on the observation that the degradation of symbols in a document image from the effects of copying and scanning occurs mostly around the edges of symbols. Nagy et al. also observed the near-edge degradation of document images by considering the effects of point spread functions on the quality of a scanned image [72]. This type of degradation does not adversely affect to the readability of the document and does not provide structural information. Conversely, it is observed that pixels farther from edges have dramatic effects on the readability of symbols and are structurally more important. By ordering the error pixels in distance-to-edge order, we can exploit their statistical characteristics by putting the most probable error pixels last, and we can provide structural coding, by putting pixels which have the potential to greatly impact the correct recognition of characters first. This paves the way for compression by packing the energy in the error stream, for lossy compression by terminating the error stream at an index calculated by the signal-to-noise ratio, and for progressive transmission by interleaving the error signals for a set of image marks based on their distance order [49].

To perform successful symbolic compression, we must perform a segmentation that decomposes the image into components that repeat. These components then need to be clustered so as to generate a set of representative prototypes. The quality of performance on these tasks, along with the quality of the image, can drastically affect the overall compression performance. In our approach to this problem we concentrate on bi-level document images that have large textual image content. This is not a strict requirement, but allows us to measure the performance of our document image compression scheme effectively. Work done by Sennhauser and Ohnesorge [94] and Sauvola and Pietikäinen [93] proves that block-based compression gives better performance on document images that contain images. It is also useful to consider algorithms such as Etemad et al. [25], Anatoncopoulos et al. [7], Jain et al. [45], Pavlidis et al. [80], and others [28, 79, 108] to provide image segmentation into regions of different types so that different compression schemes may operate on the regions. We assume that there exist enough components in the text regions that can be extracted by connected component analysis and that can be recognized correctly by a human reader. The fact that the components are repetitive in nature makes them a rich source for compression. Also the fact that document images contain a large amount of textual material laid out in a two-dimensional space makes them ideal for a two-dimensional redundancy reduction scheme. Since the useful information in the textual portion is contained in the components, access to the components provides access to that information.

There is a strong case for the use of symbolic compression in document image coding, in addition to space savings. With the use of only about one percent of the information in the original image, it is possible to code the dominant symbol shapes and their locations in the image. Since most of the information in a document is in the symbols, it is possible to perform compressed-domain analysis by using a symbolic compressed representation. Using only this representation, though it is not perfect, it is possible to perform a large number of document analysis tasks [75] such as skew estimation/correction [11, 13, 63, 73, 84], Optical Character Recognition (OCR) [71], layout analysis [74, 80], and so on. Although some work has been done on the processing of compressed document images [65, 100, 101] the outlook for applying such techniques to pixel-based methods does not appear promising.

The organization of this dissertation is as follows. Chapter 2 describes a data representation model which is designed based on requirements related to compression and processing. This representation not only facilitates the derivation of key components of the compression algorithm but also streamlines the design and implementation of a number of compressed-domain tasks. This chapter provides details about the general approach, the compressed representation in terms of streams, compression/decompression of the streams, and modules to integrate various tasks and data structures.

Chapter 3 presents an argument for the use of a special class of clustering algorithms in the compression algorithm. In-depth discussion of clustering algorithms is given along with a new method that facilitates symbolic compression. Motivation and examples are given to support our choices.

Chapter 4 discusses various methods of performing residual coding. The residue is a data structure used in the compression algorithm to allow for lossless and scalable lossy compression. In this chapter a number of methods are pursued which provide desirable lossy representations along with higher compression factors. A method of structural coding is also presented which provides improvements in both lossy and lossless representation. This coding has an immediate application to transmission problems. Analytical and empirical results are given.

Chapter 5 analyzes the rate-distortion trade-off of the residual codes mentioned in Chapter 4. This trade-off is applicable to progressive transmission and lossy compression problems. An OCR-based distortion measure is used to evaluate our code relative to competing methods.

Chapter 6 describes a set of tasks which we were able to perform successfully in the compressed domain. In this chapter we review the algorithms that were used along with performance results and benchmark outputs (when applicable).

Chapter 7 discusses extensions of our approach to grayscale documents, graphics, networked databases, and hypertext documents. Some preliminary results are given, but most of the extensions are still speculative.

1.6 Summary of contributions

The following is a list of contributions made in this dissertation along with brief explanations.

Compressed data representation: The compression algorithm creates data structures that have specific uses and vary in characteristics and statistics. Our representation of the data contributes to its efficient use for compression and processing.

Pattern clustering: One major component of the compression algorithm is clustering of the set of patterns. A clustering approach is pursued which matches the patterns and classifies them based on their match scores. We introduce a new pattern matching method that attempts to preserve structural properties by weighting component pixels based on their distances to the closest edge of the prototype.

Residual coding: After substitution of cluster prototypes for the patterns, we code the residuals. Since the residuals constitute the majority of the compressed representation, we develop a hierarchy that exhibits desirable compression, lossy compression, and progressive transmission qualities. Three coding methods are derived from a novel idea based on structural components of the residuals. A unique advantage of these codings is the reduction of image pattern variability rather than the traditional resolution reduction to achieve lossy representations.

Rate distortion: We pursue a functional approach to the measurement of distortion as it applies to compression and transmission issues. We derive a common platform for testing and evaluation.

Compressed-domain processing: The tasks that can be performed on images when they are represented in the compressed domain are limited for traditional image compression techniques. As mentioned previously, there has been some work done in this area but the outlook for the development of a general paradigm does not look promising. In our work we promote compressed-domain processing not only to achieve higher compression but also to improve application-specific processing performance.

The contribution of this idea is highly philosophical in that a true compressor should determine redundancy in a way that reflects an underlying message. That message should be accessible since its constituents are redundant and were captured during compression. A noise source (uncorrelated white noise) does not carry any message, hence it cannot be compressed; anything that can be compressed should carry a message.

Some preliminary work has been done on extending our ideas to grayscale document image compression. Other future work entails the extension of our techniques to other media types in order to arrive at a hyperdocument representation suitable for transmission and compressed-domain processing. Graphics and network extensions are not pursued here but appear quite promising.

Chapter 2

Basic compression and data representation

To address various aspects of compression, organization, and transmission in a uniform way, we need to provide a data representation model. Based on the requirements outlined in Chapter 1, we have chosen to use a model based on a pattern matching and substitution algorithm, but we need to analyze the available information to arrive at a suitable organization. This is desirable for a number of reasons. One reason is that it facilitates further research and improvements, but more immediate consequences are identification of key algorithms, their interaction in a unified representation, and preservation of compressed-domain analysis.

Use of pattern matching and substitution for compression requires several levels of information. Since there exists a well-defined set of patterns within a document image we would like to organize the patterns so that they are accessible to document analysis tasks. Once an organization is derived, we can consider what are the constituent processes that yield the necessary data. The most important aspect of the data representation is its use for subsequent processing. Several levels of representation need to be addressed for efficient access. Some algorithms require knowledge of the positions of characters (skew estimation/correction) while other require knowledge of their shapes (keyword searching). A representation thus needs to separate these types of information. There exist several levels of data types, and their successful integration is crucial to effective application-level processing.

This chapter is organized as follows. We will discuss our general approach to compression in Section 2.1 and our representation of the document page in Section 2.2 by describing its organization in terms of data streams. We discuss in Section 2.3 the use of these streams for compression and decompression. We give some examples in Section 2.4 followed by a summary in Section 2.5.

2.1 Approach

The first step in our symbolic document image compression method is to find an initial set of patterns in the image which can be used to form a library. In the case of Latin text, performing a connected component analysis on the binary image provides a reasonable starting set. For connected scripts or text in which the basic units are disconnected, more extensive segmentation would be necessary.

A small portion of a typical document is shown in Figure 2.1a, and the bounding boxes of the connected components are shown in Figure 2.1b. Because these patterns tend to appear repeatedly in the image, they form a basis for compression. In cases where multiple characters touch to form a single component, or where a single character is split into multiple components, representing them as a new component lowers

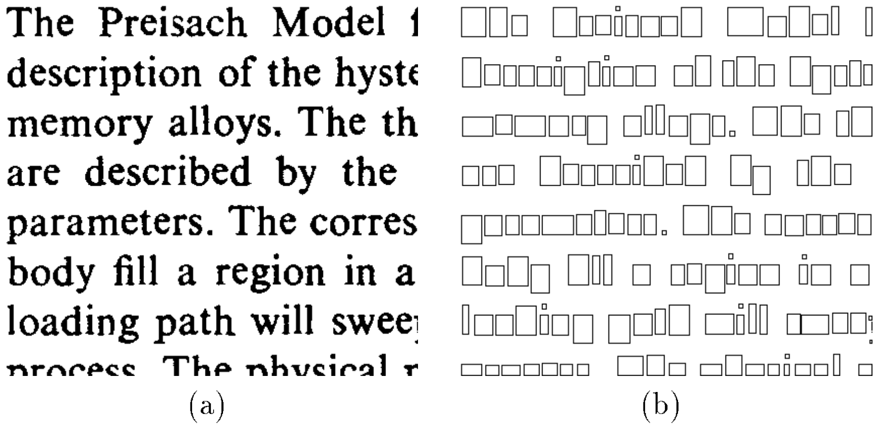

The Preisach Model f
description of the hyste
memory alloys. The th
are described by the
parameters. The corres
body fill a region in a
loading path will swee
process. The physical r

Figure 2.1: a) A sample document image; b) the bounding boxes of its connected components.

the compression factor only slightly. If salt and pepper noise is present, it may give rise to small components; this will reduce compression but will have little or no effect on the readability of the document.

We next treat each component as an observation and try to determine a best set of classes (clusters) of the components and to choose a prototype image for each class. After all observations have been processed, the prototype map typically looks like the one shown in Figure 2.2. The shapes shown in Figure 2.2 resemble English characters because of the primarily English content of the original document. For a document rich in mathematical symbols, some of the prototypes would resemble mathematical symbols, and similarly for non-Latin languages the prototypes would capture their symbol content. Clustering algorithms with different characteristics affect the eventual symbol representation which needs special attention. A number of algorithms are considered and discussed in Chapter 3.

Each cluster of components is represented by a prototype. Depending on the sample space, there will exist some amount of variability in the clusters. For some clusters, the amount of this variation may be large enough that some of the components differ significantly from the prototype and extra information must be recorded to remedy this. A residual map, the difference between a given component and its prototype, is preserved and used to recover the components in a lossless form when necessary. Examples of a component, a prototype, and the corresponding residual map are shown in Figure 2.3. In addition to coding the prototype, we code the residual map separately so that access to individual symbols can be achieved by access to their residual maps. The overall encoding scheme is shown in Figure 2.4.

In our work special attention is given to performing document analysis tasks in the compressed domain, without full decompression. In many situations only parts of the encoded information pertaining to the given task require decompression.

Our approach makes a strong case for the use of symbolic compression in document image coding. We will show that it is possible to code the dominant symbol shapes

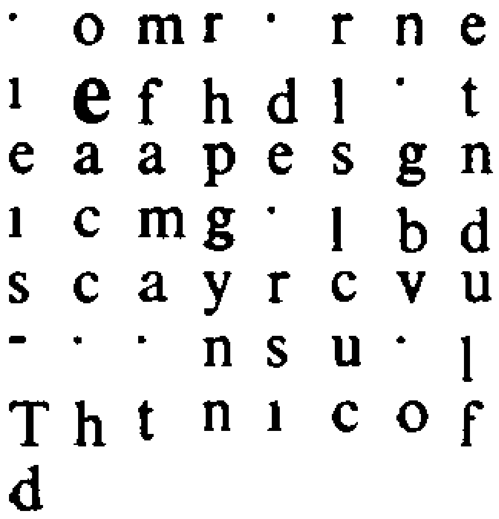


Figure 2.2: Typical cluster prototypes from a textually rich image.

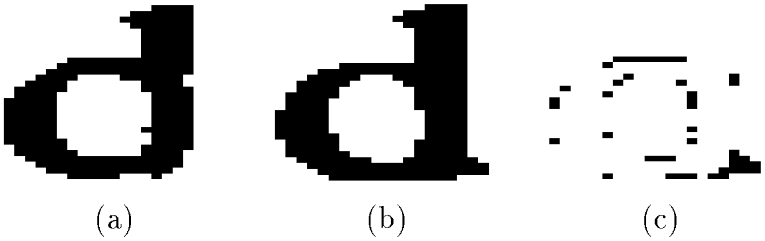


Figure 2.3: a) A component; b) a prototype; c) the residual map.

and their locations within the image using only about one percent of the original image data. Using only this encoding it is possible to implement a large number of common document analysis tasks such as skew estimation/correction [11, 13, 63, 73, 84], Optical Character Recognition (OCR) [71], and layout analysis [74, 80].

In our approach, we use an indexable representation [53] composed of independent streams for the prototypes, the locations of symbol instances, and the residual maps. We have shown that lossy compression, progressive transmission, sub-document retrieval, skew estimation/correction, and keyword searching can all be done efficiently using this representation [53].

Our contribution is in the development of a compression system that promotes compressed-domain analysis by allowing symbol access. Although the approach works best with clean images where patterns repeat, it is flexible enough to adapt to situations where the components correspond to arbitrary patterns in the image as opposed to symbols, or where many symbols are mis-clustered.

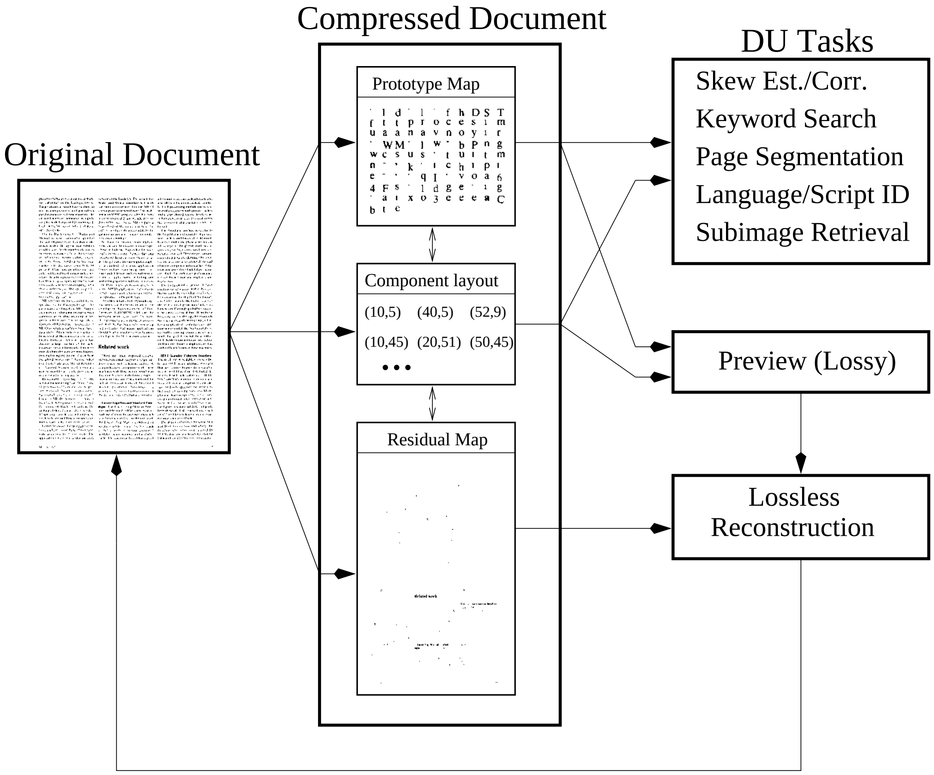
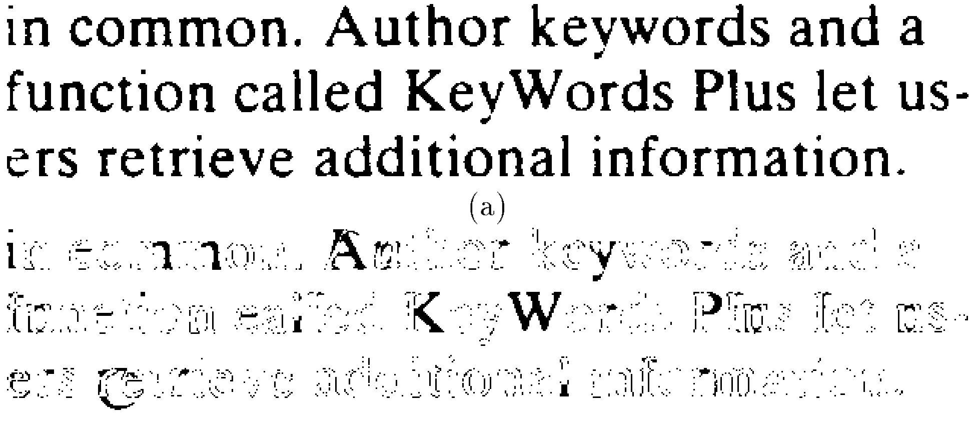


Figure 2.4: Representation of a compressed document image with relevant processing access.



in common. Author keywords and a function called KeyWords Plus let users retrieve additional information.

(a)

(b)

Figure 2.5: Example of ambiguous memberships for (a) a section of an image and (b) the resulting residual components.

2.2 Compressed document representation

Prototypes are easily calculated by the formation of clusters and these prototypes are substituted for the actual components by indexing their locations. The specification of component location is done in two parts. For coding efficiency the page is divided into blocks of 256 × 256 pixels each of which can be addressed by a single byte. A component is indexed by the relative location of the upper left corner of its bounding box with respect to the upper left corner of the block. (The number of components in each block is also stored.) These locations and the block's address yield the absolute addresses of the components. This method of addressing is used in order to provide direct access to the components. Relative position specification is used in compression by Witten et al. [112], by specifying offset location from the bottom right of the previous component to the top right of the next component. Their method has smaller storage requirements but does not provide direct access to the components.

Once we have obtained clusters and chosen prototypes we need to represent the residuals (the differences between the components and their associated prototypes). Regardless of the clustering algorithm, we have observed that there always exist residuals that affect the renditions of the original components, at least as regards readability. An example of a set of residuals is shown in Figure 2.5[a]. A residual map is associated with each component and in a representation based on symbol access, the residuals are indexed based on their associated components. A small region of an image was purposely chosen that had many components that did not cluster well. This in not representative of common residual maps.

[a] In this example, we see three types of residuals - some that do not effect readability (the silhouettes), some that do affect readability (the letter 'K' was replaced by prototype 'l'), and one that did not cluster and relies solely on the residual for rendition (the letter 'W').

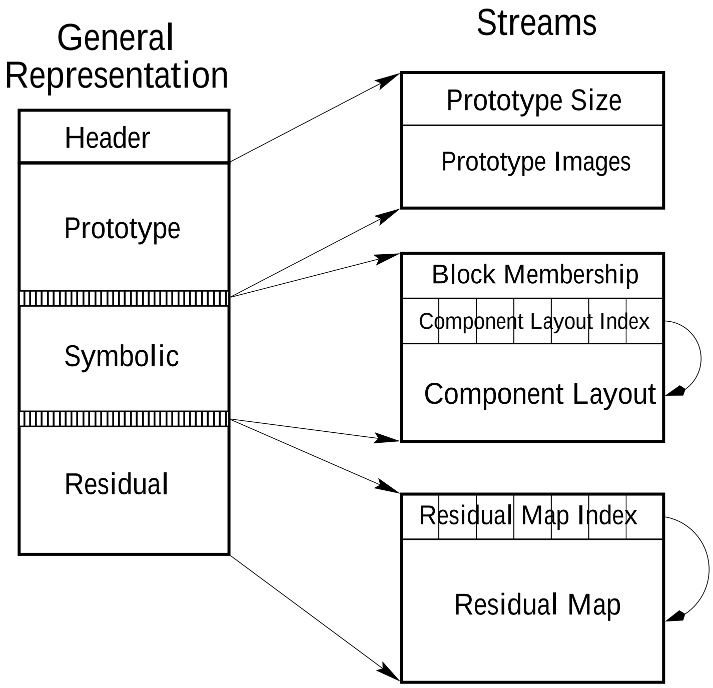


Figure 2.6: Data structure organization.

2.2.1 Page representation

In order to provide spatial access to image components in the compressed domain, it is convenient to encode four types of information in four independent sections: the file header section, the prototype section, the symbolic section (see below), and the residual section. The file header contains general information which provides indexes to the prototype, symbolic, and residual sections and an index to the header section of the subsequent page in a multi-paged document. (In this paper we have considered only single-page documents.) The prototype section is a collection of prototype bitmaps and their sizes. The symbolic section provides an encoding of the locations of components in the image and the prototype of which each component is an instance. The residual section contains the residuals produced after substituting the prototypes for the actual components. Each of these four sections is encoded as a set of streams. Figure 2.6 shows the organization of the streams; they are described in greater detail in the following subsections.

2.2.2 Streams

The file header section contains a single **header stream** and serves as an overall road map for the representation, specifying global document image parameters as well as indices to the prototype, symbolic, and residual sections and the streams they contain. The prototype section encodes the library of prototypes and contains two streams, the **prototype size stream**, which records the size of each prototype bitmap, and the **prototype image stream**, which encodes the actual bitmaps. Two extra NULL prototypes are used to serve as indexes for small and large graphic components which are represented in the residual stream. These prototypes are used to specify appropriate numbers of bytes for the sizes of the graphic components so they are indexable.

The symbolic section specifies the location of each component along with the prototype which represents it. To reduce storage requirements in recording the component locations, the image is divided into blocks of size 256×256 and component locations are specified relative to each block. To calculate the absolute address of a component during decompression, the **block membership stream** provides a block index and the **component layout stream** provides the vertical and horizontal offsets of the components within the block along with their prototype labels. The **component layout index stream** is used to index the component layout stream after it is compressed.

The residual section stores the residual for each component and is indexed in the same way and in the same order as its symbolic counterpart. This requires the **residual map index stream**, an index into the residual map, and the **residual map stream** itself.

The advantage of representing different types of information by different streams and then indexing the streams is that applications which need access only to limited information do not need to decompress the entire document. For tasks such as skew estimation and correction, which may require only the locations of the components, we need access only to the prototype size and component layout streams. For a task such as keyword searching or OCR, the prototype needs to be accessed, and minimal access to the residual map stream may also be needed. To extract and decode subregions of the document image, we can access the needed blocks, and we can then access the components in these blocks efficiently using the index structures.

The size of each stream is specified in a stream header. The header remains uncompressed so that access to each stream is independent of access to the other streams. Each stream can be compressed individually or a common "dictionary" can be implemented and made available in the header. In our implementation, we index into the component layout stream by referencing block boundaries and into the residual map stream by referencing individual components.

2.3 Compressing the streams

Huffman coding is used to compress the streams; it requires two steps. First the streams are scanned to create a common Huffman lookup table. Second the individual streams are transformed according to the newly created table. Creating indexes into the compressed streams will be discussed below.

2.3.1 Header section

Header stream: This stream consists of indices to the streams contained in the prototype, symbolic, and residual sections. The indices are stored as the sizes of the individual streams; they remain uncompressed.

2.3.2 Prototype section

Prototype size stream: Prototypes are limited to 256×256 pixels so we can use single bytes for their horizontal and vertical dimensions. These pairs are ordered as a stream and compressed. The compressed size is recorded in the header. When required, the compressed stream can be decompressed and used to index the prototype image stream.

Prototype image stream: The prototype image stream orders the pixels of each prototype, either by row or column. The resulting array is packed into the prototype image stream and compressed. The compressed size is recorded in the header. For decompression, the compressed stream is indexed by the size stream. Knowing the sizes and number of prototypes from the prototype size stream, and the size of the decompressed prototype image stream, the stream can be unpacked into the set of two-dimensional prototypes.

2.3.3 Symbolic section

Block membership stream: We have divided the document image into non-overlapping blocks of size 256×256. To index the blocks, we record the first row and column of the block in two bytes. We create a stream for the number of components in each block, compress the stream and record the size of the compressed stream in the header. During decompression we extract the stream and use it to index the blocks.

Component layout stream: For each component in a block, we append to a list its position in the block, by specifying a horizontal and vertical offset, and the cluster to which it belongs, by specifying the identification of its prototype. If the component is associated with a NULL prototype, we also record its size (the height and width of its bounding box). For each block, we compress the list and record its compressed size in the header. We add this size to the component layout index stream (described below) in one byte and start a new record for the next block. When all components in all blocks are exhausted, we record the overall stream size. For full decompression, we ignore the index stream and decompress the entire component layout stream. For partial decompression we use the size information in the index stream and decompress only the needed part of the component layout stream.

Component layout index stream: To index the component layout stream we form a stream based on the compressed sizes of the component layout records. We compress the entire stream and record the resulting size in the header. When the information in the stream is needed, we decompress the entire stream.

2.3.4 Residual section

Residual map stream: Using the same encoding as for the component layout stream, we order the residual maps into a stream. For each residual map, we first pack the pixels into bytes, submit that portion of the stream for compression, record the compressed size in one byte, and record the ending bit offset in one byte. We do this for all residual images and record the overall size of the compressed residual map stream in the header. For full decompression, we extract the entire residual map stream based on its compressed size and unpack it. For partial decompression, we use the decompressed residual map index stream to extract the needed segments from the residual map stream.

Residual map index stream: For the index, we form a stream based on the compressed size of the list of residual map streams for each component. We compress the entire stream and record the resulting size in the header. When the information in the stream is needed, we decompress the entire stream.

2.4 Results

To test our system we used text regions from the first 122 scanned images (A001BIN.TIF to A006N.TIF) in the UWASH I database which is available from the University of Washington [32]. For synthetic images we used LaTeX-generated files from the same database (L000SYN.TIF - L00MSYN.TIF). Our basic compression package converts binary TIFF images to and from symbolically compressed images. For synthetic, LaTeX-generated images, it took a total of 855 seconds to compress 23 files for an average of 37 seconds. The average resulting file size was 45312 bytes compared to 57848 bytes in CCITT G4 coding and 82112 bytes in CCITT G3. For comparison, the same images were compressed using packed bits and LZW compression, yielding average file sizes of 198,015 bytes and 110,810 bytes respectively. With original images of 2550 by 3300 pixels, the compression ratios are summarized in Table 2.1. For the case of scanned images, compression time was an average of 66 seconds for the 122 files of Table 2.1. (The time is for an implementation running on a SparcStation 20 with the Solaris 2.5 operating system.)

Image	Symbolic	G4	G3	JBIG	MG	LZW	PACK-BITS
Synthetic	23.2	18.2	12.8	26.9	39	9.5	5.3
Scanned	12.2	17.8	9.7	22.3	27	8.7	5.5

Table 2.1: Average compression ratios for synthetic and scanned images for different compression schemes. The images are taken from the University of Washington Database [32]. Averages were taken over 20 synthetic images and 100 scanned images.

2.5 Summary

We have proposed a data organization that is designed to work with symbolic compression. Given that we can identify sources of redundancy and useful sources of information we designed the organizational model to address both. In our methodology we identified the source of redundancy as the repetition of components found in document images. The source of useful information is found to also be in the components, but more specifically it resides in the locations and shapes of the components.

Redundancy is captured by determining representative classes of observed components, which is easily done by clustering. These clusters are represented by prototypes and are indicative of the components' shapes. Therefore, prototypes need to be preserved and stored to be accessed individually, as we have described. While the shapes of components are captured in prototypes, their locations need to be stored independently to allow for dissemination of useful information. This was done by providing for a symbolic layout section. The remaining information relates to the residual coding. Redundancy removal within the residual section will be addressed in Chapter 5. Some useful information for this purpose can be obtained from the prototype image; to make this information available we must index the residuals based on the components. We proposed a residual coding method which is well suited for partial retrieval and compressed-domain processing.

Having provided a specification of the data organization, we can now proceed with specification of actual algorithms.

Chapter 3

Clustering

3.1 Introduction

A large number of clustering algorithms have been described in the literature [4, 86]. Some algorithms, despite their wide usage for other purposes, are not applicable to symbol clustering. The goal of symbol clustering is to identify the intra-class similarities of the symbols. Due to size and pattern variations in the components, some clustering algorithms may not provide a satisfactory result, as mentioned below for two of the algorithms.

One example of an inappropriate clustering algorithm is principal component analysis, where image rows or columns are concatenated into a vector and for a given set of symbol images a covariance matrix is computed. The eigenvalues of this matrix give the variances along the eigenvector directions. The effectiveness of this method depends on the rows and columns providing consistent information for all of the images. This will not be the case if the images are of different sizes. To remedy this one can pad the observed image to a fixed size with a constant value or scale the input image to a fixed size. The padding presents a problem when the input images are artificially weighted to the size of the padded region. In the example shown in Figure 3.1, it can be seen that all observations have similar shapes (a circular shape will overlap all the observations to some degree) in the upper portion of the image, and that the lower portions of observations 6 and 7 look the most dissimilar (one descender is on the left and one is on the right with no overlap). However, since observations 1 to 5 are padded, the weight of the descenders in observations 6 and 7 is 2/7 of what it should be (not as discriminating as it should be). Normalizing the components is not a viable option, since prototype generation will not be possible in a cluster where the sizes of the components are different. Scaling of images to a normalized size creates a situation in which an appropriate prototype may become impossible to find. The fact of the matter is that the size information is an effective discriminator and it can be used to effectively separate observations {6,7} and {1,2,3,4,5} of Figure 3.1.

A second type of clustering algorithm, vector quantization [30], is also affected by the variability in observed sizes. In vector quantization, each symbol image is first ordered as a vector and this vector is considered to be a point in a high-dimensional space. A small set of "prototype" vectors is arbitrarily chosen and the distances to all the prototypes from all of the observations are calculated. We associate with each observation the prototype closest to it. The prototypes are then refined by taking the means of their associated observations, and the process is repeated. The prototypes and the distance measure divide the high-dimensional space into regions of coarser granularity; hence the name quantization. The results obtained by this method are

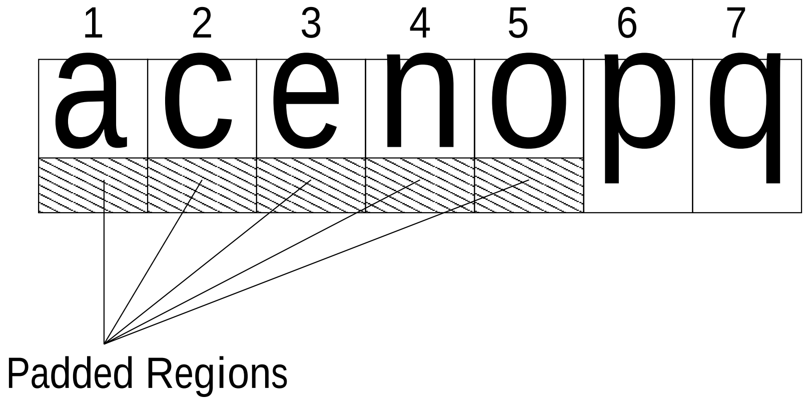


Figure 3.1: Example of padding.

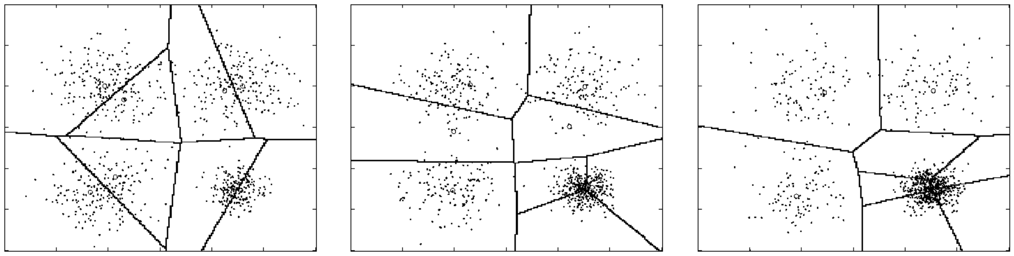

(a) (b) (c)

Figure 3.2: Vector quantization regions in a two-dimensional sample space with a) equal-population clusters; b) a 3:1 population difference; c) a 5:1 population difference.

population-dependent; more prototypes will be found in regions of the space where symbol images occur. Figure 3.2 shows how the regions change when the number of samples in one of the clusters is increased.

A third class of clustering methods, which we will primarily use here, are pattern matching and substitution approaches. If a candidate pattern is a good match to an existing prototype, it is classified as a member of that prototype's class; otherwise, it is considered for possible creation of a new class. The advantages of using this type of method are that we do not need a priori knowledge about the number of classes and that the method works with a variety of image sizes.

We can describe clustering via pattern matching more formally by first defining notation for an observation X and a prototype P:

$$X = \begin{cases} X(n,m) & \text{for } (n,m) \in \mathcal{R}_x = (\{1,...,N_x\},\{1,...,M_x\}) \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

$$P = \begin{cases} P(n,m) & \text{for } (n,m) \in \mathcal{R}_p = (\{1,...,N_p\},\{1,...,M_p\}) \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

where it is assumed that foreground pixels have value 1 and background pixels (and pixels outside the bounds of $\mathcal{R}_x$ and $\mathcal{R}_p$) have value 0. A number of pattern matching methods which can be used to define measures of closeness between an observation and a prototype are discussed below.

The rest of this chapter deals with matching methods that can be used in our pattern matching and substitution clustering process. We first discuss a simple Hamming distance in Section 3.2. We extend this to a weighted Hamming distance in Section 3.3. A different approach is then suggested to take into account background and foreground errors in Section 3.4. In Section 3.5 compression-based (entropy-based) matching techniques are discussed. In Section 3.6 we propose a new matching technique based on the distance transform which has the desired property of weighting structural components appropriately. We conclude with some remarks in Section 3.7.

3.2 Hamming distance

The simplest method of matching two binary images is to measure their dissimilarity by the number of pixels that are not equal. An error map calculated from the exclusive OR (XOR) of the observed image and the prototype is given by

$$E(n,m) = \begin{cases} X(n,m) \oplus P(n,m) & \text{for } (n,m) \in \mathcal{R}_x \cap \mathcal{R}_p \\ X(n,m) & \text{for } (n,m) \in \mathcal{R}_x - \mathcal{R}_p \\ P(n,m) & \text{for } (n,m) \in \mathcal{R}_p - \mathcal{R}_x \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

where $E(n,m)$ is defined for all $(n,m) \in \mathcal{R}_e = \mathcal{R}_x \cup \mathcal{R}_p$. Since the XOR operation returns a value of 1 for a mismatch, summing the error map provides a measure of mismatch,

$$\overline{M} = |\mathcal{R}_e| - M = \sum_{(n,m)\in\mathcal{R}_e} E(n,m) \tag{3.4}$$

where $|\mathcal{R}_e|$ is the highest mismatch score and M is the actual match. In this formulation, maximizing the match is the same as minimizing the mismatch of (3.4). Figure 3.3 shows three examples involving an intra-class observation, a possibly confusing inter-class observation, and an obvious inter-class observation, along with their mismatch scores. The amount of mismatch shows some degree of discrimination among those cases, and by use of a threshold, we are able to identify some classes. However, if the threshold is too high some intra-class confusion may arise, such as clustering 'e' shapes with 'c' shapes. Taking a low threshold would result in too many clusters and would hinder compression efforts. It is desirable to use a distance measure that provides enough separation for dissimilar shapes.

3.3 Weighted Hamming distance

Improving the distance measure to discriminate between similar images ('e' and 'c' of Figure 3.3) is desirable; the weighted Hamming distance [85, 112] provides such an improvement. This distance measure gives greater importance to error pixels

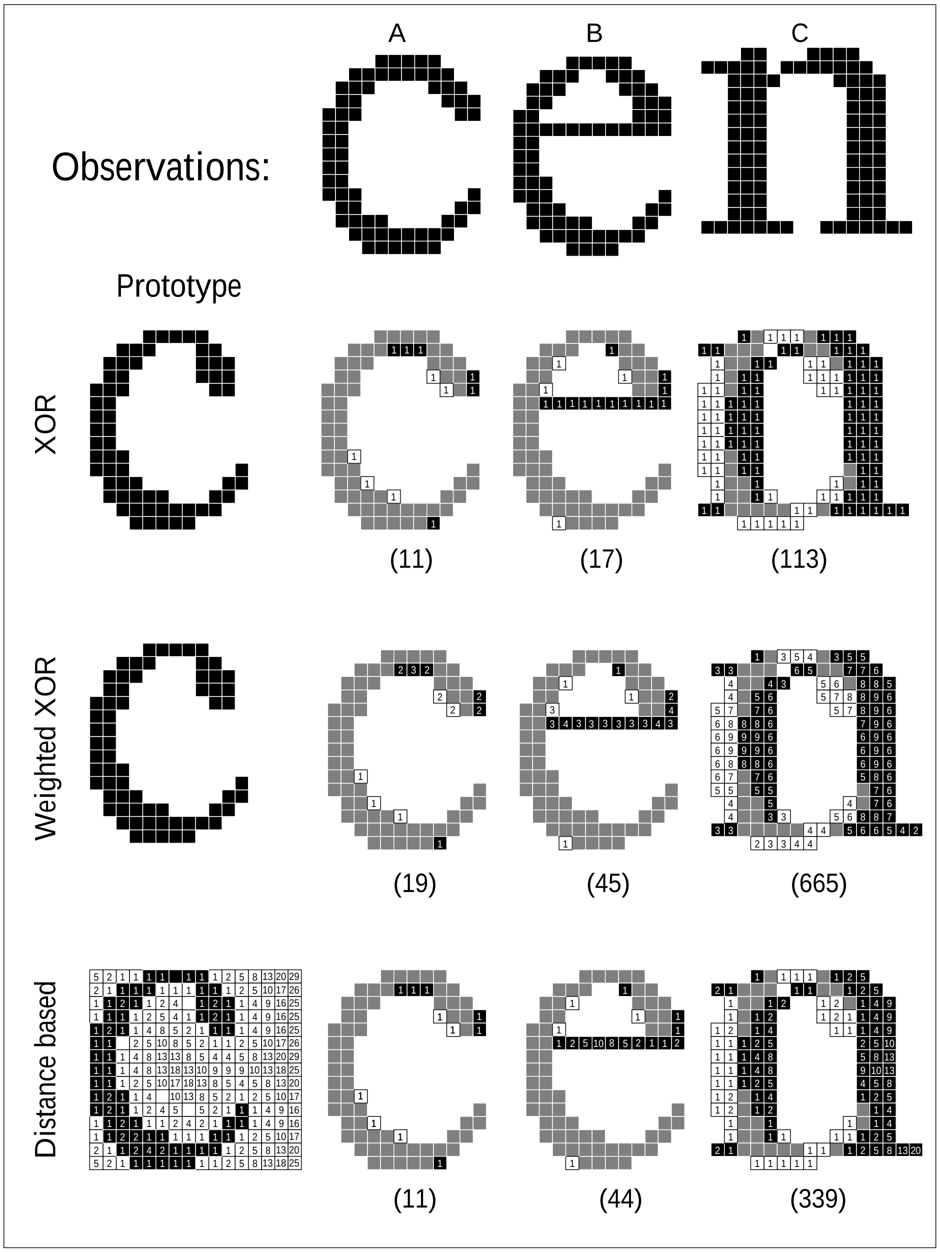


Figure 3.3: Matching results for three examples of observed components. Observation A is similar to the prototype, observation B is close, and observation C is very different. XOR-ed, Weighted XORed, and Distance-based differences are shown, as well as the pixels contributing to the mismatch, and the mismatch value in parentheses.

which appear in close proximity to other error pixels. Error pixels which appear close together tend to correspond to structurally meaningful features.

The weighted Hamming distance operates on the error map of (3.3) by summing over a neighborhood of each error pixel:

$$E_w(n,m) = E(n,m) \times \sum_{(k,l)\in\mathcal{N}(n,m)} E(k,l) \tag{3.5}$$

where $\mathcal{N}(n,m)$ is the 3×3 neighborhood of the (n,m)th pixel. With this weighting strategy, error pixels that occur in a group will give a higher mismatch than isolated error pixels. Figure 3.3 also shows the results of using the weighted Hamming distance measure for the same observations and prototype. Note that the small difference between observations A and B is now much larger when the mismatch score is calculated by (3.4) and using the weighted error map E_w.

3.4 Sum of weighted AND-NOTs

When we use an XOR operation the source of the errors is not considered. In particular no distinction is made between errors in foreground pixels and errors in background pixels. It may be desirable to give more importance to the foreground pixels since most of the information is contained in them. This can be done by using the AND-NOT measure. The weighted AND-NOT map is defined by

$$\begin{aligned} E_{wan} &= \left[(X \wedge \overline{P})(n,m) \times \sum_{(k,l)\in\mathcal{N}(n,m)} (X \wedge \overline{P})(k,l)\right] \\ &\vee \left[(\overline{X} \wedge P)(n,m) \times \sum_{(k,l)\in\mathcal{N}(n,m)} (\overline{X} \wedge P)(k,l)\right] \end{aligned} \tag{3.6}$$

This map is useful in cases where the weighting has elevated the mismatch level due to misalignment, as illustrated in Figure 3.4.

3.5 Compression-based template matching

Compression-based template matching [43] is a method of matching which attempts to measure the mutual information between the prototypes and the observations. This is can be done by minimizing the entropy of the residual. The entropy of a binary signal $E = \{e_i, i = 1, ..., |\mathcal{R}_e|\}$ of distribution $P(e)$ is

$$H(E) = -\sum_{e=0}^{1} P(e)\log_2(P(e)) \tag{3.7}$$

$P(e)$ is estimated given E. Equating $H(E)$ to a mismatch value favors conditions where the complexity of E is low which translates into cases of mostly zeros or mostly ones. This is rather different from previous approaches in which the number of pixels in the residual map is minimized. In compression-based template matching, the idea

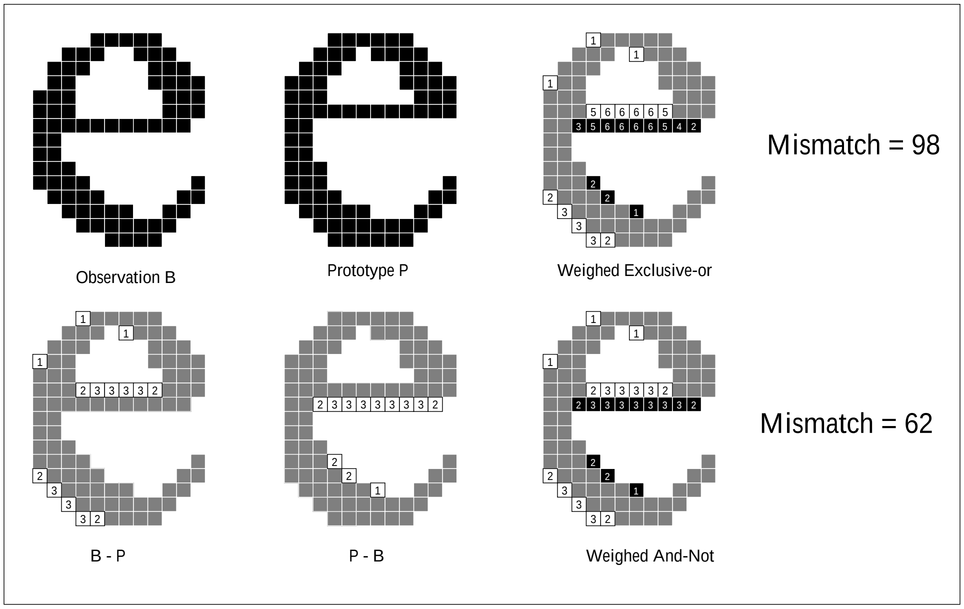


Figure 3.4: Example of an observed component compared to a prototype by measuring mismatch as weighted XOR and as weighted AND-NOT.

is to keep the residual pixels non-random with respect to each other and achieve a lower entropy for the residual map. An immediate enhancement is to replace the probability with one that is conditioned on the formed clusters. Mutual information is then measured by

$$I(E|k) = -\sum_{n,m \in \mathcal{R}_e} \log_2 P(E(n,m)|k) \tag{3.8}$$

where k is the cluster index. While this probability can be estimated from the members assigned to the cluster, it can also be estimated as a function of the prototype image (convolution with some filter [116]).

3.6 Distance-based template matching

Our final method of matching attempts to weight each error pixel according to the probability that it was corrupted by degradation or noise. In this way we give lower weights to error pixels which are likely to have resulted from noise and higher weights to others.

In studies of document image degradation at the character level it was observed that pixel errors occur more often close to the edges of characters [48]. The probability of a pixel changing its value decays as a squared exponential with the distance to an edge. Specifically [48], the probability of a pixel changing value from background to foreground at distance d from an edge is given by

$$P(1|d,\alpha,b) = 1 - P(0|d,\alpha,b) = e^{-\alpha d^2} \tag{3.9}$$

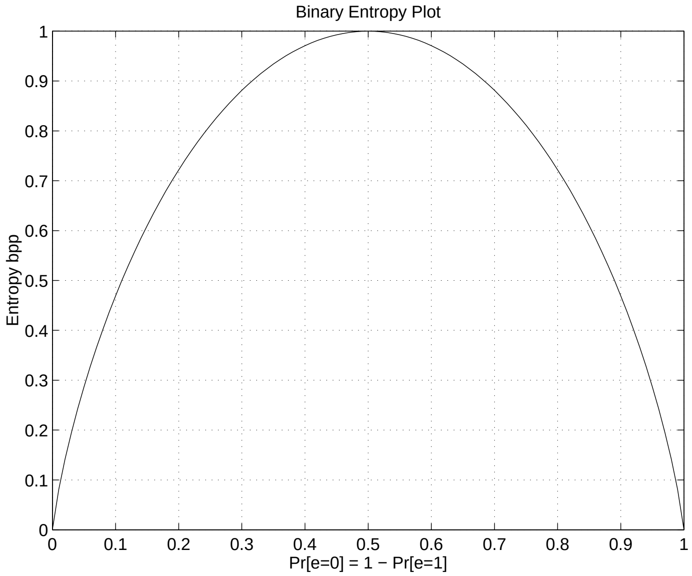


Figure 3.5: Plot of entropy for a binary signal as a function of sample probability

where α is the background decay rate and b denotes the fact that the pixel in question really belongs to the background. Similarly, for foreground pixels, the probability of changing value is

$$P(0|d,\beta,f) = 1 - P(1|d,\beta,f) = e^{-\beta d^2} \tag{3.10}$$

We can use this fact to conclude that observed differences which occur between a prototype and a candidate pattern away from the edge of the prototype are more likely to be the result of meaningful structure and not noise and should therefore be considered with higher priority during matching.

We can use this degradation model as a basis for a matching model which assigns weights based on relative distances from the prototype edge. To simplify the model, we assume that α is equal to β and calculate the weight matrix as a distance transform. This greatly reduces the computational requirement by avoiding a scan to all components to determine α and β and by removing an extra exponential operation. We also use only a relative distance measure; this allows us to avoid the computation of two squares and a square root.

To compute the distance transform, consider a prototype P. Let $\mathcal{F}$ be the set of indices for the foreground pixels of the prototype, and $\mathcal{B}$ the set of indices for the background pixels. Knowing that $\mathcal{F} \cap \mathcal{B} = \emptyset$, the square-of-the-distance transform, or distance map, of prototype P is calculated by

$$\begin{aligned} D_P^2(n,m) = \min\{ \quad & (n-k)^2 + (m-l)^2 \mid \\ & (k,l) \in \mathcal{F} \ , \ (n,m) \in \mathcal{B} \text{ or} \\ & (k,l) \in \mathcal{B} \ , \ (n,m) \in \mathcal{F}\} \end{aligned} \tag{3.11}$$

This measures the closest (squared) distance of an unlike pixel to the current pixel.

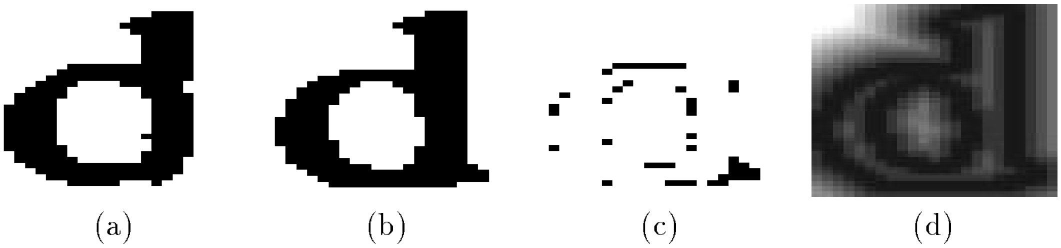


Figure 3.6: Example of a) an observation, b) a prototype, c) a residual map, and d) the distance map of the prototype.

The study of distance transforms dates back to the late 1960s. A classical reference by Rosenfeld [87] discusses a number of algorithms for its computation, while for images represented by quadtrees one can use a method suggested by Samet [90]. Other recent algorithms to compute distance transforms can be found in [16, 33, 60, 104, 107].

Figure 3.6 shows a typical prototype and a map of the distances to the edges. (Figure 3.3 showed the distances overlayed on the prototype image.) Summing the error pixels weighted by their distance values provides a better separation for inter-class observations. For the intra-class observations in Figure 3.4, it is easy to see that the distance-weighted mismatch is 26 since all error pixels occur on the edges and there are only 26 such pixels. Distance weighting separates the closest inter-class observations shown in Figure 3.3 by 11 and 44.

It should be pointed out that distance-based template matching could easily be adapted to other clustering techniques, at the cost of more computation. An example is adaptation to vector quantization. Consider the clusterings of Figure 3.2, and suppose that for high values of x and y we would like to weight the x values more heavily, and for low values we would like to weight the y values more heavily. This would shift the region boundaries of Figure 3.2a in such a way that at the top right we would have vertical thin regions and at the lower left we would have horizontal thin regions. The result would be a partition that is dependent on location in the space as well as on the population of observations in that area of the space. The distance ordering has a number of desirable properties, and provides a good framework for document image processing and handling.

3.7 Summary

In this chapter we considered a number of clustering algorithms that might be used in our compression system. Due to the size variability of the input images and the unknown number of clusters, we choose a match-and-classify approach. This approach allows for the use of different matching functions so that specific features of observations can be exploited. Prototype generation is easy once a set of clusters is determined. We simply average the cluster members to arrive at their representative prototypes.

The simplest match criterion tested was the sum of the Hamming distances, which was immediately improved to a weighted sum of Hamming distances. While Hamming distance sums the mismatch value as the number of pixels that were unequal, weighted Hamming distance puts more weight on groupings of pixels which were unequal. This provided a reasonable improvement. We also considered an alternate distance, namely AND-NOT distance. The reason for considering AND-NOT is that for the XOR function, the output is independent of the fact that foreground and background pixels of either the prototype or the component may not match. There may be situations where a slight misalignment contributes a large amount toward the mismatch value. By using the AND-NOT function, the groupings of unequal pixels are weighted based on their foreground and background memberships.

Another class of matching functions tested was entropy-based. This class maintains a low level of complexity for the residuals; however, it is not guaranteed that the prototypes resemble meaningful components. The matching measure is the complexity of the residual image. A simple count of residual pixels yields a probability mass distribution that is used to calculate an entropy measure. This measure was adapted to include a condition that the entropy of the residuals belonging to a class be minimized. Since all pixels are not created equal, especially for document images, the entropy contributions of all pixels cannot be uniformly combined; they need to be weighted based on their potential contribution toward appropriate prototype generation. Entropy-based template matching may contribute towards the compression of images, but it is more desirable to create prototypes which contribute toward correct recognition. Once a clustering algorithm can guarantee correct assignment of components to clusters (i.e. all clusters consist of components which would be recognized as the same symbol as the prototype by a human reader), we can then employ compression-based clustering to further minimize residual complexity while keeping recognition intact. Such performance has never been achieved; most researchers in this area confirm that correct prototype representation is possible only for document images that are of good quality.

A distance-based approach is suggested here to address some concerns regarding the pixels' contributions. A model based on character degradation is used for this purpose. This model suggests that pixels close to edges of components are most likely to change in value due to degradation effects. We extend our model to hypothesize that pixels far from edges must contain large amounts of recognition information. This is supported by the fact that most document images that are degraded are still readable. Therefore we associate a distance measure with each pixel and weight its difference from the prototype based on that distance. This allows discrimination of shapes that are fairly similar but have strongly dissimilar structural components. This favors the clustering of similar structurally significant patterns. While we still cannot guarantee that cluster members will all be recognized as the same symbol, we have contributed toward achieving a single recognizable component per cluster.

There exist several clustering algorithms which vary in performance, but there exists no clustering algorithm that performs perfectly. Therefore, it is essential to include at least partial residual information to assure a representation that allows for correct recognition of characters. Distance-based matching provides good clustering of components based on their structural features and readability, but it does not operate perfectly, so residual coding is needed to address concerns beyond clustering. This will be covered in Chapter 5. Another concern is the quality of the input components. In Chapter 4 we revisit component segmentation to see if the input components are well suited for our clustering methods.

Chapter 4

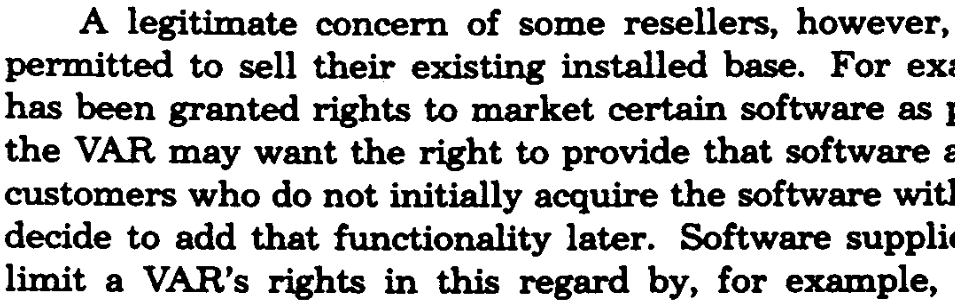
Segmentation-based clustering

4.1 Introduction

Enhancements to clustering, such as the work done by Bloomberg and Vincent [14] and Zhang and Danskin [116], can greatly improve performance both in lossless and lossy compression. While there have been efforts to improve clustering in the symbolic compression of document images, it is unrealistic to expect clustering alone to address all aspects of prototype generation. For characters that are connected or broken in the document image, a segmentation based on connected components will not yield isolated characters after clustering, no matter how good the clustering is. In this case the resulting set of prototype image maps contains a certain level of redundancy which was not adequately captured by the segmentation and clustering algorithms. If a clustering algorithm does not require the output of the segmentation routine, it is considered to be a joint segmenter and clusterer and does not fall in the category of simple clustering algorithms. An ideal clustering routine would interact intelligently with the segmentation routine and derive the best possible prototype images while the segmentation routine provides a better segmentation given the underlying prototype maps. Eventually, in residual coding, which involves a majority of the data, the data can be efficiently organized and coded, as has been demonstrated by Howard [37] and as shown in Chapter 5.

For the general symbolic compression problem, segmentation has not been a major issue. This is because any component regardless of shape and size will be coded appropriately, although not as efficiently as it could be. With the introduction of small amounts of degradation in the original image, the structural integrity of the character shapes degrades and the components either start touching or breaking apart. It is therefore necessary to review alternative methods of segmentation to improve compression performance for such documents. An example of such a document is shown in Figure 4.1.

Initially we used a connected component algorithm for segmentation. Although such segmentation can be done quickly, the process is sensitive to connected or broken characters. While most of the spatial features of connected characters are still preserved, their segmentation remains a non-trivial task. In symbolic compression, fundamental character shapes are identified and clustered to generate prototypes that embody the constituent patterns of the image. In cases of touching characters, clusters will be created for observed instances and constituent shapes will not be identified. We propose to use the prototypes obtained in the first pass to perform partial matching and an alternative segmentation on those components which have not clustered well. The output of the segmentation is fed into the clustering routine again for an



**A legitimate concern of some resellers, however,
permitted to sell their existing installed base. For ex
has been granted rights to market certain software as
the VAR may want the right to provide that software
customers who do not initially acquire the software wit
decide to add that functionality later. Software suppli
limit a VAR's rights in this regard by, for example,**

Figure 4.1: Example of a document image containing a large set of connected characters.

enhanced set of representative prototypes. This process can be repeated to improve the compaction of the prototype library.

To improve clustering, compression, and processing of document images we consider an integrated segmentation and recognition approach where the initial segmentation and identification of clusters identifies stable prototypes. We then iterate a process to refine both segmentation and clustering of components. The prototypes from the original clustering are used to re-segment the previous components and generate a new set. The new set of components is clustered to form new prototypes, and so on. This process is performed iteratively with varying degrees of segmentation to determine the set of prototypes for the image that yield the best compression.

This chapter is organized as follows. In Section 4.2 we describe our general approach toward a joint segmentation and clustering methodology. In Section 4.3 we describe how subcomponent matching is performed to arrive at possible segmentation alternatives and in Section 4.4 we describe the segmentation method itself. In Section 4.5 we study the integrated process of segmentation and clustering which provides a basis for enhanced segmentation. In Section 4.6 we present the results and in Section 4.7 we summarize and make comments about the algorithm, including its limitations and future enhancements.

4.2 Approach

To provide integrated segmentation and clustering, we take the connected components of the document image as the first guess at the components, cluster them, and generate a prototype set as shown in Figure 4.2. Using these prototypes, we attempt to resegment larger components by considering all possible segmentations of the components, matching and attempting to identity sub-instances of the previous prototypes (Figure 4.3). The entire list of components is visited and all combinations of vertical cuts are considered in order to determine the best possible match. The list of match values associated with each segmentation is recorded. During segmentation, the best match of a sub-component to a prototype is chosen and the associated cut-point is used to create a new observation. After considering all components for segmentation,

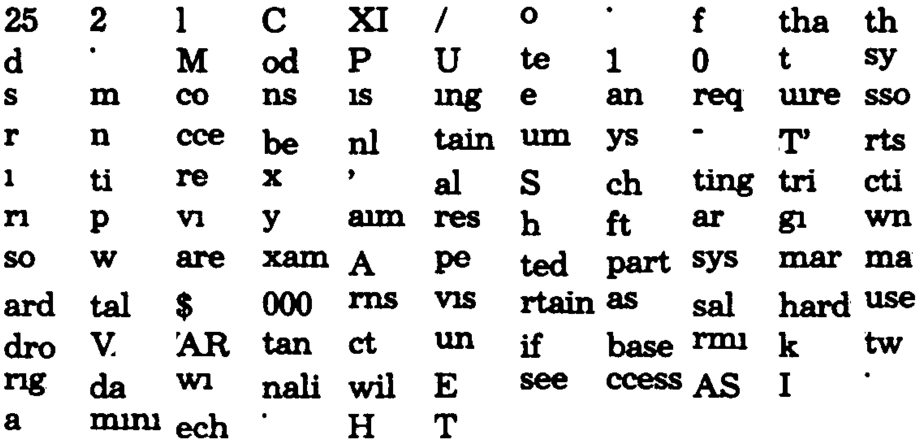

25 2 1 C XI / o · f tha th
d · M od P U te 1 0 t sy
s m co ns ıs ıng e an req uıre sso
r n cce be nl tain um ys - T' rts
ı ti re x ' al S ch ting tri cti
rı p vı y aım res h ft ar gı wn
so w are xam A pe ted part sys mar ma
ard tal $ 000 rns vıs rtain as sal hard use
dro V. 'AR tan ct un if base rmı k tw
rıg da wı nali wil E see ccess AS I ·
a mını ech · H T

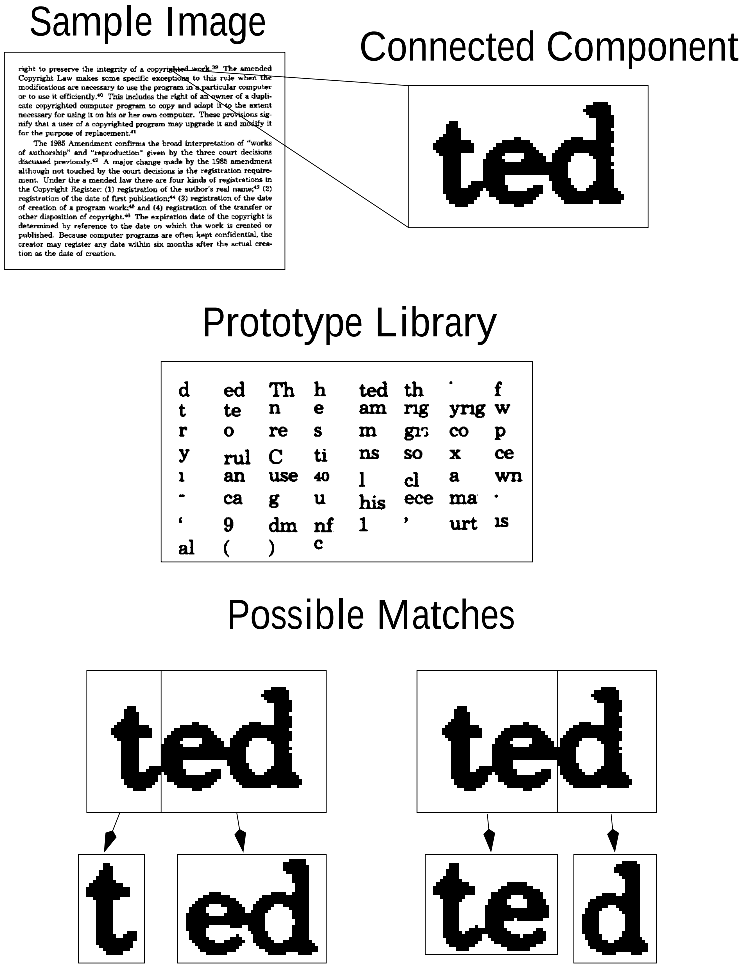
Figure 4.2: Example of a prototype set that is rich in touching components.

a new component list is created which reflects the new instances. These components are then used for clustering and to create a new set of prototypes. The process of segmentation and clustering is repeated several times to allow for segmentation of large components.

4.3 Subcomponent matching

For components that consist of touching characters, we assume that prototypes resembling the characters which they contain are present and that we need to identify them and locate their positions. To identify these sub-instances, it is necessary to match smaller components with parts of larger components. The two factors in finding this match are shape and position. A number of other factors need to be discussed; however, the general methodology is to obtain the largest-size set of prototypes that are independent. This is required since blind decomposition of patterns into their smallest constituents results in a set of the smallest prototypes, such as periods. Figure 4.4 shows several prototypes from the library in Figure 4.2. It is clear that the “ted” cluster, having a number of observed instances assigned to it, is composed of touching components, but how to define the set of its sub-components is unclear. It is clear that decomposition of the “e” prototype (Figure 4.4d) is desired; however, a better decomposition is that of the “ed” prototype (Figure 4.4e), since it is a constituent but one that is bigger in size than the “e” prototype. The reason for this is that if most instances of “e” appear as “ed”, then representing “ed” is more efficient than representing “e”. Since this is not the case for the most part, we can conclude that the “ted” pattern should first be decomposed into “t” and “ed” and then “ed” should be decomposed into “e” and “d”. The components shown in Figure 4.4c and f are also constituents but this level of decomposition is clearly undesirable.

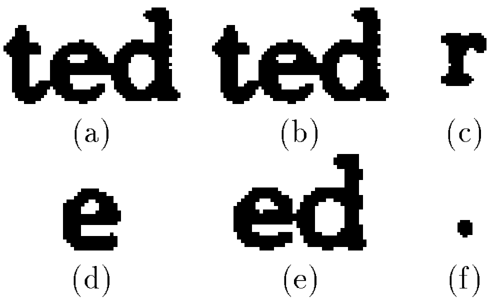
Since we are dealing with rectangular bounding boxes to segment regions of com-



right to preserve the integrity of a copyrighted work.[39] The amended Copyright Law makes some specific exceptions to this rule when the modifications are necessary to use the program in a particular computer or to use it efficiently.[40] This includes the right of an owner of a duplicate copyrighted computer program to copy and adapt it to the extent necessary for using it on his or her own computer. These provisions signify that a user of a copyrighted program may upgrade it and modify it for the purpose of replacement.[41]

The 1985 Amendment confirms the broad interpretation of "works of authorship" and "reproduction" given by the three court decisions discussed previously.[42] A major change made by the 1985 amendment although not touched by the court decisions is the registration requirement. Under the a mended law there are four kinds of registrations in the Copyright Register: (1) registration of the author's real name;[43] (2) registration of the date of first publication;[44] (3) registration of the date of creation of a program work;[45] and (4) registration of the transfer or other disposition of copyright.[46] The expiration date of the copyright is determined by reference to the date on which the work is created or published. Because computer programs are often kept confidential, the creator may register any date within six months after the actual creation as the date of creation.

Figure 4.3: Location of prototype sub-instances in connected components.


(a) (b) (c)

(d) (e) (f)

Figure 4.4: Set of observations considered for segmentation (a-c) and corresponding proposed constituents (d-f).

ponents and the text lines are horizontal, we consider only vertical cuts. The algorithm to generate segmentation hypotheses is as follows. For each vertical cut, we determine the minimum bounding boxes of the two resulting components. The resulting components are then matched to all prototypes which they may contain. The mismatch is measured by the sum of unequal pixels weighted by the distance transform. Given the distance transform of (3.11),

$$\begin{aligned} D_P^2(n,m) = \min\{ \quad & (n-k)^2 + (m-l)^2 \mid \\ & (k,l) \in \mathcal{F} \ , \ (n,m) \in \mathcal{B} \text{ or} \\ & (k,l) \in \mathcal{B} \ , \ (n,m) \in \mathcal{F}\} \end{aligned} \tag{4.1}$$

the mismatch $\overline{M}$ is calculated by

$$\overline{M} = \sum_{n,m} E(n,m) \times D_P^2(n,m) \tag{4.2}$$

where $E(n,m)$ is the error between the segmented component $X(n,m)$ and prototype $P(n,m)$ (3.3),

$$E(n,m) = \begin{cases} (X \oplus P)(n,m) & (n,m) \in \mathcal{R}_x \cap \mathcal{R}_p \\ X(n,m) & (n,m) \in \mathcal{R}_x - \mathcal{R}_p \\ P(n,m) & (n,m) \in \mathcal{R}_p - \mathcal{R}_x \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

A match is identified if the minimum mismatch is lower than a threshold. Figure 4.5 shows the best segmentation, where the segmented portion matched to an existing prototype. In such a scenario, if one segmented component shows a good match, the cut is kept regardless of the match between the other component and any existing prototype. This is necessary in order to decompose larger components which have more than two constituents. For components having three or more constituent elements, decomposition can take place from the right or left and can proceed to the middle.

To ensure proper segmentation and avoid over-segmentation as was shown in Figure 4.4c and f, several factors are considered. The first factor is the position of the segmentation cut. It is undesirable to segment a component at its extreme right or left edges. If cuts at these locations are picked, they are most likely due to small prototypes which do not result in an adequate decomposition. The distance of the cut from the middle of the segmented component takes this factor into account.

The second factor is favors components that are touching along small numbers of pixels. The number of foreground pixels that coincide with the segmentation cut takes this factor into account.

The remaining factors are based on size. It is more desirable to decompose larger components, so we take the size of the component considered for resegmentation into account. It is more desirable to decompose a component into the biggest prototypes that are smaller (at least in area) than the given component, and so the area of the

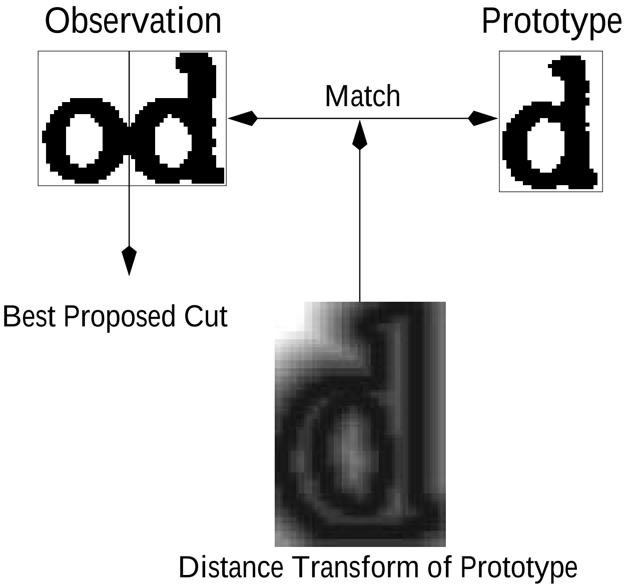


Figure 4.5: Best proposed segmentation cut for an observation and a corresponding template that was matched based on the distance transform.

prototype considered for best segmentation is taken into account. Figure 4.6 shows examples of these factors. In terms of weights, we use a mismatch value of

$$\overline{M}_w = \overline{M} \times \frac{Weight_1 \times Weight_2}{Weight_3 \times Weight_4} \tag{4.4}$$

Upon completion of component matching, the best cut, the mismatch value, and the closest matched prototype are kept. The segmentation process then uses these values to determine the best segmentation.

4.4 Segmentation

For segmentation, we need to consider the cuts considered in partial matching for cases of touching characters. Each component is visited and from among all possible cuts, only the cut that yields the best partial match is considered. Note that only one cut is allowed per visit, greatly reducing the complexity of segmentation versus the more general two-cut scheme. Using two cuts, a number of fail-safe mechanisms need to be considered to avoid empty objects and residual objects created by taking a cut that is not exactly on the edge of a neighboring constituent component. The best match is then thresholded to either keep the cut or discard it. Various thresholding constants were used with varying amounts of effect on compression. Examples of components that were segmented in this way are shown in Figure 4.7.

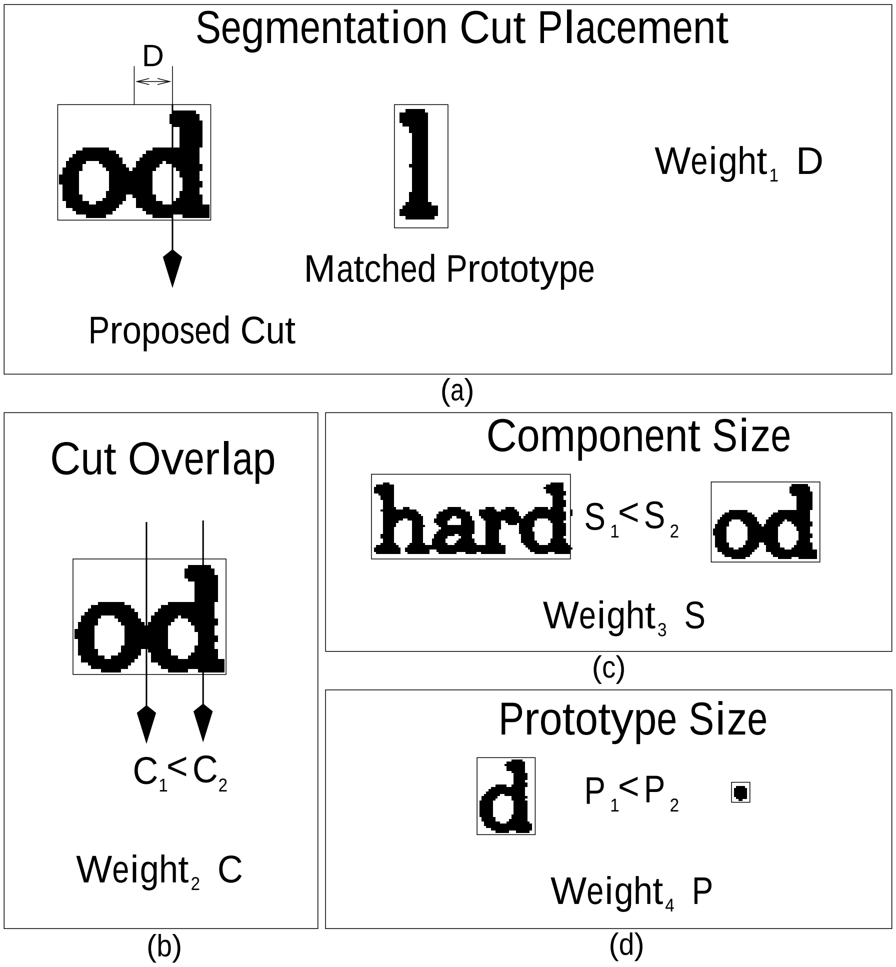


Figure 4.6: Mismatch weighting by considering (a) the cut position, (b) overlap with foreground pixels, (c) component size, and (d) prototype size.

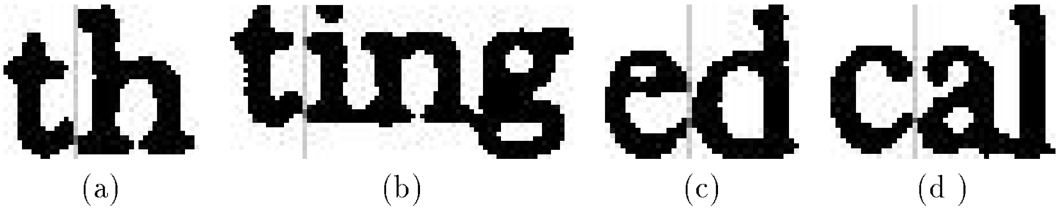


Figure 4.7: Examples of segmented components.

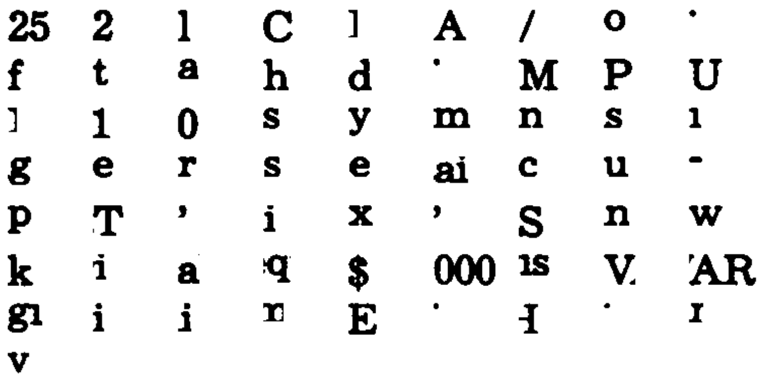


Figure 4.8: Prototypes of document image associated with Figure 4.2 after integrated segmentation and clustering.

4.5 Integrated segmentation and clustering

After the original segmentation (connected component analysis) and clustering, further segmentation is performed by specifying a threshold on the partial matches to the components. All initial components are considered for segmentation. If the mismatch value is below the threshold, the initial component is divided into two parts. After all components have been considered for segmentation and divided as necessary, the clustering algorithm is applied to the new set of observations. This process is iterated repeatedly to allow for sufficient decomposition of large connected components. The prototype library originally shown in Figure 4.2 looks like the one shown in Figure 4.8 after processing.

Integrated segmentation and clustering is also applicable to the case of disconnected characters. A reverse approach needs to be taken in which combining components is done instead of dividing them. A nearest neighbor approach is used to combine components, and the results are matched to the prototype library. The closest match is kept if its mismatch value is below a threshold. The same methodology is used as in the matching of partial components to prototypes except that the cost factor associated with a cut is replaced by one that is associated with component combination. Prototype instances that are larger than the broken component (prior to joining) are considered and the smallest such prototype is favored. This alternative method of resegmentation is also beneficial.

4.6 Results

The algorithm can be evaluated in terms of its performance in prototype generation, component decomposition, effectiveness of prototype library generation, and residual

	Compressed Size		Library Size		Pointer Size		Residual Size	
File	(I)	(F)	(I)	(F)	(I)	(F)	(I)	(F)
S1	44260	41552	4550	2642	3158	3948	34038	31931
S2	39828	38240	5600	5160	2258	2443	20088	28648
S3	27300	26041	3563	2703	1579	1921	20763	19833
S4	30397	28926	4640	2481	1787	2069	22377	22633
S5	27014	26416	2705	1971	2818	3198	19318	18834
S6	25050	24489	3347	1748	2254	2950	17648	17537
S7	34818	33941	3724	3448	3207	3393	25537	24648
S8	40100	38207	3544	3252	3856	3984	29811	27985
S9	51713	49778	4520	4116	3927	4340	40280	38084
SA	23978	22987	3331	2812	1715	2035	17459	16490
Total	390678	376527	44630	34896	30468	34551	281618	280648
% Δ	(4)%		(23)%		14%		(.3)%	

Table 4.1: Statistics of selected files with initial (I) and final (F) compression characteristics after applying integrated segmentation and clustering algorithm.

image generation. While the overall gain in compression is important, it is also important to see how this gain is achieved.

Table 4.1 shows statistics for a selected group of document images. For debugging purposes we considered small samples of document images which had large numbers of connected components. Images that did not have high numbers of connected components showed only a small amount of improvement. For the examples in Table 4.1 we performed symbolic compression with and without segmentation to study the amount of improvement. We used a representation that stored the prototype library, symbol location, and residual information separately, so that the effects of segmentation could be studied across all aspects of the symbolic compression.

The overall compression factor was observed to have decreased by 4.5%. However, this was due to the small image sizes; the clustering process was not able to use the rich population of observed components available in larger images. The compression factor is expected to increase for large images where there exist large numbers of components as well as instances of touching characters that span a large number of characters. The interesting result is in the reduction in the size of the prototype library. Since the prototype library is a small part of the representation, it does not affect overall performance but it is the major portion of the lossy compression representation. For lossy compression consisting of only the prototype images, we need to code 66083 bits when we do not use segmentation, but only 60614 bits when we do use it, resulting in an 8% increase in compression. The effects of segmentation on processing are clear, in that the prototypes are more representative of the characters and provide a better symbolic representation. Basically, any result that is obtained by compressed-domain processing is more dependable with segmentation than without.

4.7 Summary

It is logical to consider alternative methods of segmentation in addition to the basic method of connected components. In most cases, document images are of high quality and the connectedness of characters is preserved. This cannot be guaranteed, however, and it is desirable to devise a method that can gracefully deal with degradation. Because proportional fonts are often used in typesetting, and because of the point spread functions of scanners, we observe a larger reduction in performance due to touching components than due to broken components. In this chapter we have described a method that can segment connected components using the shapes of dominant components. This method has the desirable characteristic that a stable level of representation is achieved in component shape and size. It also improves compression by 4%. It is important to note that this method is independent of the clustering algorithm [116] and data representation [53, 112] used, and that it can benefit from, and contribute to, their operation. For example, a better clustering algorithm that can achieve a higher level of separation can provide better prototypes to be used in partial matching. Better segmentation results can increase the performance of compressed-domain processing in tasks such as key-word searching and skew detection.

The method presented here has not been optimized in any way, but it performs compression in a few minutes on a SparcStation 20. The images used for testing it were derived from the University of Washington Database CD ROM [32], by cropping images to contain rich text content, without scanning and copying artifacts.

Chapter 5

Residual coding

The primary contribution of this chapter is in the development of new methods of residual coding. These methods provide a hierarchical representation and allow for compression and compressed-domain processing, with natural extensions to lossy compression and progressive transmission.

5.1 Introduction

In the previous chapters, we considered a number of methods of determining suitable prototypes. Since the components that appear in document images contain several levels of ambiguity, it becomes unrealistic to pursue a conservative approach towards prototype generation. The advantage of a prototype generating process that guarantees a high level of correct recognition of components is that it results in a large set of prototypes. The factors that affect the choice of an ensemble of prototypes are image degradation, shape redundancy/variability, cluster membership, and an acceptable level of residual complexity.

Considerable complexity exists in the residual images. It is desirable to code the residuals so as to achieve as high a level of compression as possible and provide a hierarchical representation that degrades gracefully and provides correct recognition of components. Much like the preservation of structural components during the prototype generation phase, it is possible to structurally order the residuals of the image components and achieve a graceful degradation of their structural components [52]. We pursue three methods of coding based on structure. We first consider a distance-based coding method that preserves structural components when residuals are coded, and we also consider two other methods that attempt to capture the structural components when residual pixels are coded.

The remainder of this chapter is organized as follows. In Section 5.2 we discuss the basic distance-ordered code. In Section 5.3 we extend the basic distance ordering to structural ordering. In Section 5.4 we analyze the code in terms of its compressibility by performing an entropy analysis. In Section 5.5 we present the results of the analysis. In Section 5.6 we conclude with some final remarks.

5.2 Distance-ordered residual coding

The nature of the residuals is clearly dependent on the quality of the original image, i.e. how well symbols can be clustered, and on the quality of the clustering algorithm itself. In clean document images, components are, for the most part, isolated from each other and they can easily be segmented and clustered to form well-defined groups.

When the components and their prototypes are very similar, little contribution from the residual maps is necessary to accurately encode the symbols. As the images degrade, however, noise is introduced and characters tend to either merge to form bigger components or break into multiple pieces. This increase in the variability of the components in the clusters results in an increase in the complexity of the residual map.

An issue independent of how the residuals may change is how well the clustering algorithm performs in the presence of increased noise. A clustering algorithm needs to capture meaningful similarities between degraded symbols. For document images, the structural aspects of symbols are the most meaningful when the ability to recognize or read the symbols is considered. Whatever levels of image quality and clustering quality are achieved, it is desirable to code the residuals so that a compact and usable representation is achieved.

Document image degradation research suggests that the edge pixels of an ideal image are most susceptible to noise degradation. Nagy et al. [72] and Sarkar [91] used a point spread function to determine the effects of degradation on characters in processes like OCR. Kanungo et al. [48] used a quadratic exponentially decaying function to model the probabilities that foreground and background pixels are affected by noise. Extrapolating their findings, it can be hypothesized that 1) pixels close to edges are more likely to be affected, and pixels at the same distance from edges have similar noise probabilities; 2) pixels far from edges contribute the most to the structure (and recognition) of the components. In support of the second hypothesis, it can be observed that most document images are still readable after introducing a large amount of degradation at or near edges.

The above degradation formulation can be used to create a model that attempts to distinguish noise pixels from informative structure-contributing pixels during clustering and coding. The formulation predicts that pixels farther from an edge of a component contribute more to the readability of the component than do close ones. We therefore order the residual map by distance so that the pixels which contribute the least to recognition will be coded last. This is implemented by taking the distance transforms of the prototypes and ordering the residuals by distance, from farthest to closest. Consider the definitions of observation X, prototype P, error map E, and distance transform given in (3.1)-(3.3) and (3.11). The contents of E are ordered based on the distance transform of the prototype $D^2_P(n,m)$.

A typical prototype, cluster member, residual map, and distance transform map are shown in Figure 5.1. The residual in this example conforms well to the model since the prototype and observation are of the same class and most of the residual pixels lie close to the edges of the prototype. It is clear that for the examples in Figure 5.1 the residual pixels are primarily noise since the prototype and the cluster member both resemble the character 'd'. However, if an 'o' were clustered with the 'd', the residual content would contribute significantly to the ability to recognize it correctly. Given these observations, a distance ordering is necessary to exploit the residual statistical distribution and allow residual pixels with the smallest contributions to recognition to be coded last.

An example comparing column-major and distance ordering is shown in Figure

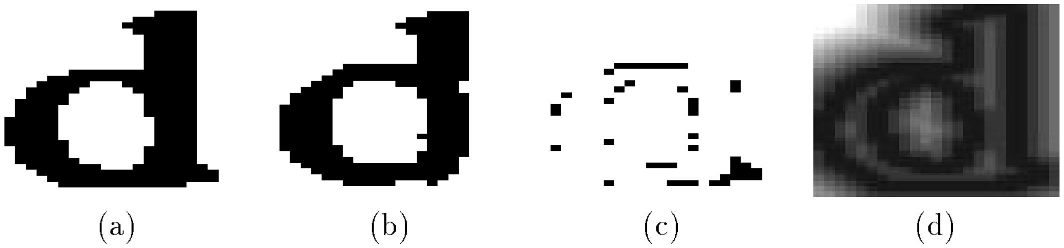


Figure 5.1: Example of a) prototype, b) cluster member, c) residual map, and d) distance transform of prototype image.

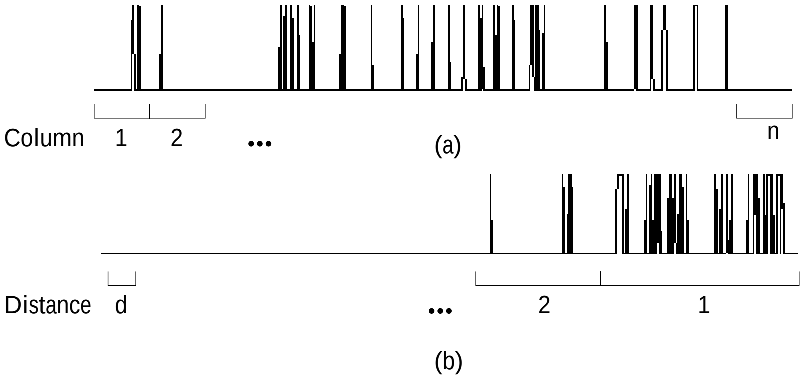


Figure 5.2: Coding of residual in Figure 5.1 for a) column-major ordering and b) distance ordering.

5.2. The column-major order results in residual components being spread throughout the code. In the distance order the relative probability of a residue increases as the code progresses (from left to right, provided the clustering is correct), as can be seen in Figure 5.2b. The ordering results in entropy reduction during compression and allows us to first transmit pixels which contribute the most to symbol recognition.

The quality of the clustering algorithm affects the residuals' complexity; there may exist components that have been matched to prototypes of different shapes. Figure 5.3 shows an example of this situation where a 'w' was classified as a 'v'. The distance-ordered code is complex. Figure 5.4 shows the progressive reconstruction of this component by using fractional portions of the code. It can be seen that from the start of the code, structural components that differentiate a 'v' from a 'w' are being formed, aiding in the correct recognition of the component. It can also be seen that this residual map does not fall into the same category as the residual map shown in Figure 5.1. From a degradation point of view, there exists a large amount of noise far from edge pixels and the distribution does not fit the model. Similar results would be obtained whenever a 'w' is misclassified as a 'v'.

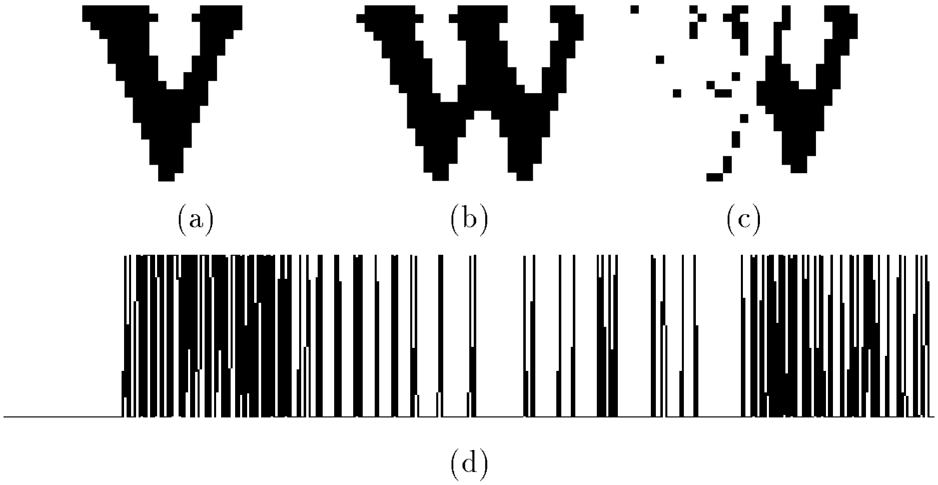


Figure 5.3: Example of inter-class observation: a) cluster prototype, b) observation, c) residual map, and d) distance-ordered residual stream.

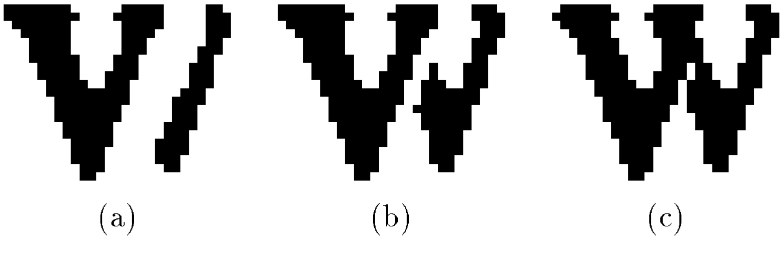


Figure 5.4: Structural contribution of the residual stream shown at reconstruction of a) 30% b) 60% and c) 90% of the residual stream.

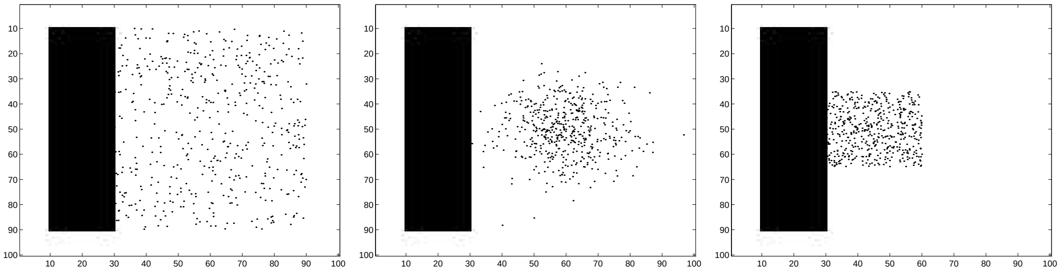


Figure 5.5: a) Low, b) medium, and c) high structural impact.

5.3 Structural coding

In the previous section we hypothesized that residual pixels that are farthest from the prototype's edge contribute the most to the correct recognition of the component. In coding the residual, we ordered the pixels by their distances. We can do better if we take into account the structure of the residual, as discussed in this section.

5.3.1 Continuous ordering

The structure of a residual involves more than its distance ordering. Structure is a hierarchical concept; a group of pixels has structure, while a single pixel can only contribute to structure. We will not attempt to give a general definition of "structure" here, but we will suggest some principles which are applicable to the character domain. To illustrate these principles, consider the three examples in Figure 5.5. All three cases show about the same number of residual pixels, and include pixels that have large distances from the component. In 5.5a, the pixels are randomly scattered; in 5.5b they are tightly clustered to form an almost connected group; in 5.5c they form a group which is almost connected to the prototype. They clearly exhibit a varying degree of structural impact.

We have experimented with a simple residual encoding algorithm based on the fact that structurally significant residual pixels are often connected to the prototype. Our algorithm is based on distance-ordered transmission. When we transmit the first residual pixel (farthest from the prototype), we connect it (by a digitized straight line segment(s)) to the pixel(s) of the prototype closest to it, yielding an augmented prototype that now includes the original residual and the line segment(s). The residual too then changes; 1's that belong to the augmented prototype now need not be transmitted, but 0's that belong to it (i.e. pixels on the line segment that have value 0 in the image) are now residual pixels and need to be transmitted. As we continue to transmit residual pixels in (the original) distance order, we connect them to the (original) prototype with line segments consisting of 1's or 0's; when the latter occurs, it may cause 1's to change back to 0's. Note that such changes affect only pixels that are closer to the prototype; thus the distances to residual pixels never increase, and the process eventually terminates when the closest residual pixels have been transmitted.

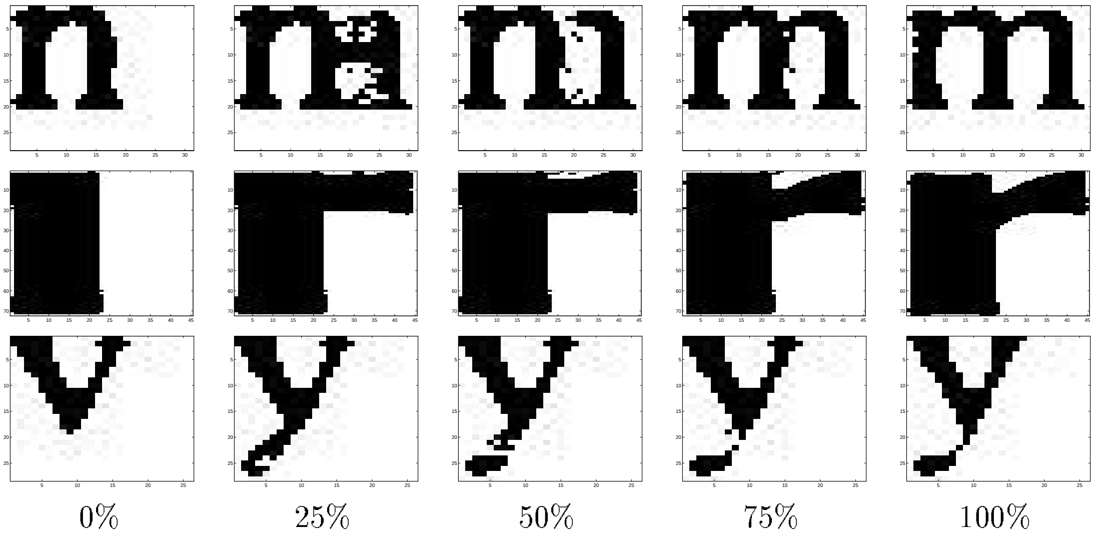


Figure 5.6: Three examples of structural coding using various percentages of the maximal distance.

The results of using this simple algorithm show that (at least) the widths of structural elements are captured correctly. Figure 5.6 shows three examples of components represented by using a percentage of the maximal distance. The first example is an 'm' component that was misclassified as an 'n' prototype. The original residual map consisted of a structural component that was connected at the top of the prototype. The representation of this component using residual pixels having distances within 25% of the maximal distance shows a large number of errors in structure estimation. However, subsequent transmissions correct the estimation and produce a shape close to that of the component. The second example, an 'r' misclassified as a dotless 'i', benefits a great deal from the structural ordering; from the onset of coding, the represented shape is very close to the original component. The third example, a 'y' misclassified as a 'v', does not take as great advantage of the structural ordering.

5.3.2 Packet-mode ordering

An enhanced variation of this approach transmits the residual pixels in distance-ordered packets (of user-defined size). When the first packet is transmitted, the closest pixels in each connected component of the packet are joined to the prototype by straight line segments. This defines an augmented prototype and a new residual; the next packet can then be transmitted. Examples of coding based on packet-mode ordering are shown in Figure 5.7 for packet sizes of 100 and 60 pixels. The size of the packet should be based on the size of the component and should be large enough that a packet can contain residuals having a range of distances, and small enough to allow a number of structural coding steps to take place.

The efficiency of the coding using packet mode ordering, in terms of both compression and representation, falls between those of continuous ordering and distance-

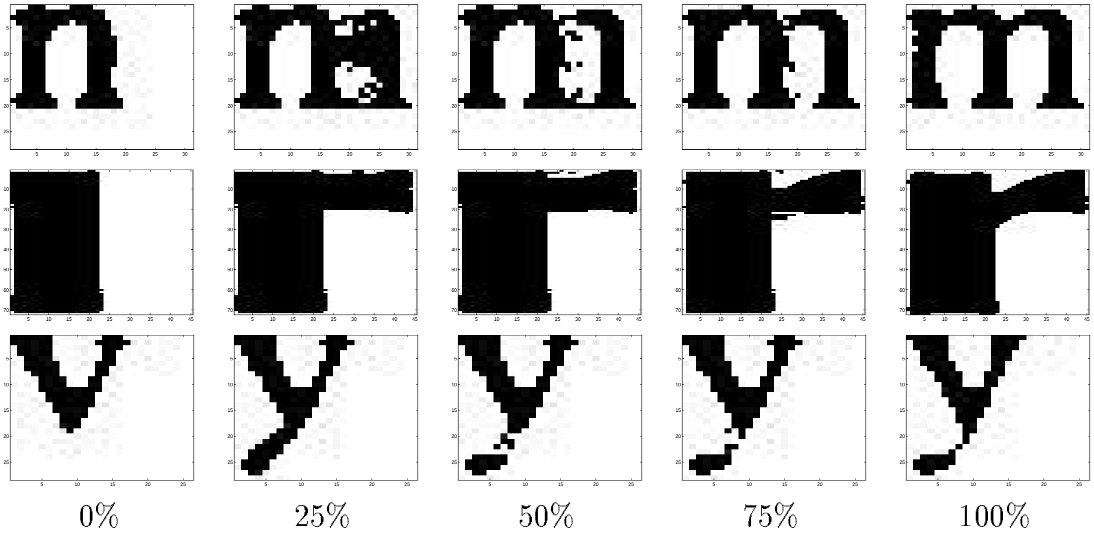


Figure 5.7: Three examples of packet-mode structural coding using various percentages of the maximal distance, and using 100-pixel (middle row) and 60-pixel (top and bottom rows) packets.

ordered transmission, since a packet of size 1 implies continuous ordering, and a packet size as big as the prototype implies distance ordering. For intermediate values, our results show that both entropy and rendition quality are more favorable, especially for large components.

5.4 Analysis

It is desirable to analyze the residual code to determine the expected amount of compression that may be achievable by these and similar coding methods. This can easily be done by either measuring the entropy as the alphabet approaches infinity, or by using probabilistic measures. In this section we will pursue both approaches by setting up a simple model and then extending it to our coding method.

5.4.1 Entropy

Distance ordering has a number of desirable properties, with the compactness of the code being the most significant. The characteristics and complexity of the residual map are directly related to the clustering algorithm. For a clustering algorithm that is optimized based on the degradation effects, the residual pixels can be characterized as having a distribution consistent with the degradation model. For unoptimized clustering algorithms residual maps might contain significant structural components. Assuming that the clustering algorithm is consistent in making cluster assignments, residual maps that contain a significant amount of structural information will have consistent mappings. In general, a clustering algorithm that can group similar shapes

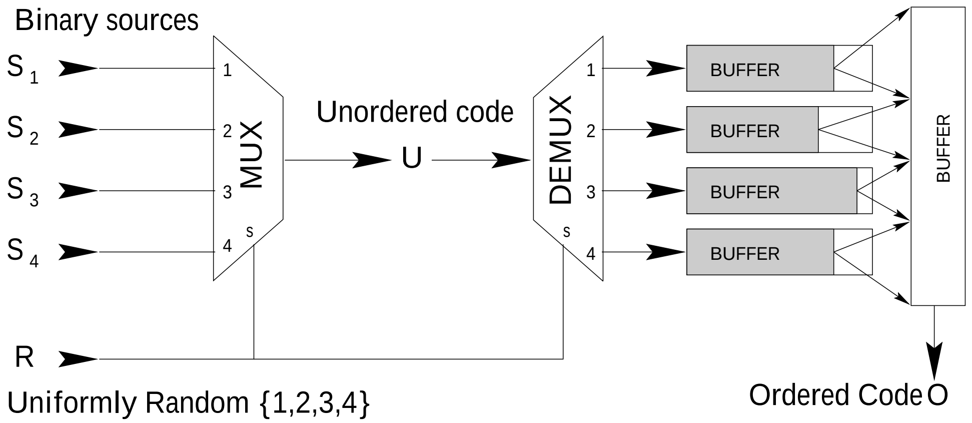


Figure 5.8: System realization of unordered and ordered binary code obtained from four independent and statistically different sources.

will be able to capitalize on the consistent distribution of residual pixels, particularly for residual maps that have significant structural components.

Up to now, the consistent distribution of the residual pixels has been shown only by observing the resulting code and verifying that the beginning of the code has consistently contained zero values in components that are clustered well. We can, however, do better. Assume that our residual pixels are samples of binary sources with varying distributions that depend on the distance model (i.e., residual pixels having distance d from an edge are samples of the same binary source). Since components belonging to the same cluster are not necessarily of the same size, the row or column ordering of the residual pixels can be regarded as a random selection from the binary sources. The distance ordering is an attempt to improve this random selection. A hypothetical scenario with only four binary sources is shown in Figure 5.8. In reality, the number of sources is unknown and the randomizer R is non-uniform and dependent on the prototypes. Let the unordered code be denoted by U and the ordered code by O.

To quantify the compactness of O over U we need to determine their entropies. The entropy of a discrete binary signal B can be calculated by

$$H_{\mathcal{B}}(B) = \sum_{b \in \mathcal{B}} -P_B(b) \log_2(P_B(b)) \tag{5.1}$$

where $\mathcal{B}$ is the binary space $\{0,1\}$, and $P_B(b)$ is the distribution of the samples based on the binary signal B. This measure of entropy is valid for a binary signal that is stationary and independent. For signals involving dependency or non-stationary processes, this measure does not show the true entropy. Consider the entropy of a discrete signal X with samples in $\mathcal{X}$,

$$H_{\mathcal{X}}(X) \;\; = \;\; E_X\left[\log_2\left(\frac{1}{P_X(X)}\right)\right]$$

$$= \sum_{x \in \mathcal{X}} -P_X(x) \log_2 (P_X(x)) \quad (5.2)$$

For $\mathcal{X} = \{0,1\}^1$ we achieve the first-order entropy which is identical to the entropy of (5.1), and for $\mathcal{X} = \{0,1\}^n$ we achieve the nth-order entropy, $H_n(X)$. In other words, a signal X which consists of samples belonging to an alphabet of size $|\{0,1\}^n| = 2^n$ needs at least $H_n(X)$ bits per sample (in alphabet) to be coded without any loss. Then for the n-bit alphabet, the compression factor is $\frac{n}{H_n(X)}$ (the inverse of the entropy rate). For the stationary and independent random variable, the entropy rate stays constant, but for our ordered binary source the entropy rate decreases as the order increases. For these signals the true entropy is

$$H(X) = \lim_{n \to \infty} \frac{1}{n} H_n(X) \quad (5.3)$$

5.4.2 Simulation

Computing the entropy for U and O is not easy since in calculation of $H_n(X)$ we need to estimate the distribution of symbols in an alphabet of size of 2^n, which is very large. For example, for $n = 15$ the alphabet size grows to 30,000 symbols, and a good estimate of the probability of each symbol requires a sample of size at least 3 million. For small values of n, we simulate the entropy of U and O as shown in Figure 5.8, using four binary sources with the probability assignments of $\{S_1, S_2, S_3, S_4\}$ shown in Table 5.1. We can then calculate the entropy rates shown in Figure 5.9. Note that the first order entropy rates of both codes are the same since $P(1) = 1 - P(0)$ is equal to the number of 1's divided by the total number of samples, and they are equal regardless of the ordering mechanism. Note also that the entropy rate of the unordered code remains reasonably constant (with an average of 0.97), implying a stationary and independent process (this is true because of the random selector R). The entropy rate of the ordered code is lower than that of the unordered code (in fact, lower than 0.75), but the exact value is not certain. The reason for the lower entropy is that the first quarter of the O code usually has zero entropy (see Table 5.1) and the average entropy of the code (the average of the entropies in the $H(S)$ column) is 0.65. The probability assignments shown in Table 5.1 were chosen to involve large differences between sources and to allow for a variation in the measured entropy rates of the unordered and ordered codes.

5.4.3 Empirical analysis

Next, we empirically derive the entropy of U and O to arrive at a measure of improvement. We can first analyze the system of Figure 5.8 and generalize $H(U)$ and $H(O)$ for the case of residual pixel ordering. Consider the four binary sources $\{S_1, S_2, S_3, S_4\}$ with associated probability distributions of $\{P_{S_1}(s), P_{S_2}(s), P_{S_3}(s), P_{S_4}(s) \mid s \in \{0,1\}\}$. Using (5.1) we can calculate the entropy of each source separately as $\{H(S_1), H(S_2), H(S_3), H(S_4)\}$. It can be easily shown that these are true entropies (stationary and independent sources) and remain constant as n approaches infinity. Since the selector

Source	P(0)	P(1)	H(S)
S_1	1	0	0
S_2	0.7	0.3	0.88
S_3	0.5	0.5	1.00
S_4	0.2	0.8	0.72
$\overline{S}$	0.6	0.4	0.97

Table 5.1: Probability assignments used to generate code for U and O.

R is uniform, $P_R(r) = \frac{1}{4}$, the probability distribution of the bits in U is

$$\begin{aligned} P_U(u) &= E_R\left[P_{S_R}(u)\right] \\ &= \sum_{r=1}^{4} P_R(r) P_{S_r}(u) \\ &= \frac{1}{4}\sum_{r=1}^{4} P_{S_r}(u) \end{aligned} \tag{5.4}$$

resulting in an entropy of

$$H(U) = -\sum_{u=0}^{1} P_U(u) \times \log_2(P_U(u)) \tag{5.5}$$

The entropy of O is a measure of the complexity of U given R. In this way the extra complexity of R is removed from the code and a lower entropy results. By definition [21]

$$\begin{aligned} H(O) &= H(U|R) \\ &= \sum_{r=1}^{4} H(U|R=r) P_R(r) \\ &= E_R\left[H(U|R)\right] \end{aligned} \tag{5.6}$$

The complexity of a sample of U, given that the multiplexed source was obtained from R, is the complexity of S_r, and $H(U|R = r) = H(S_r)$. The entropy of the ordered code O is therefore the expected value of the entropies of the multiplexed binary sources:

$$\begin{aligned} H(O) &= \sum_{r=1}^{4} H(S_r) P_R(r) \\ &= E_R\left[H(S_R)\right] \end{aligned} \tag{5.7}$$

It is important to note that the entropy indicates the maximum compression possible. While there exists no general algorithm that achieves this, there are algorithms which can get arbitrarily close [21], and when the probabilities are powers of 2, the Huffman

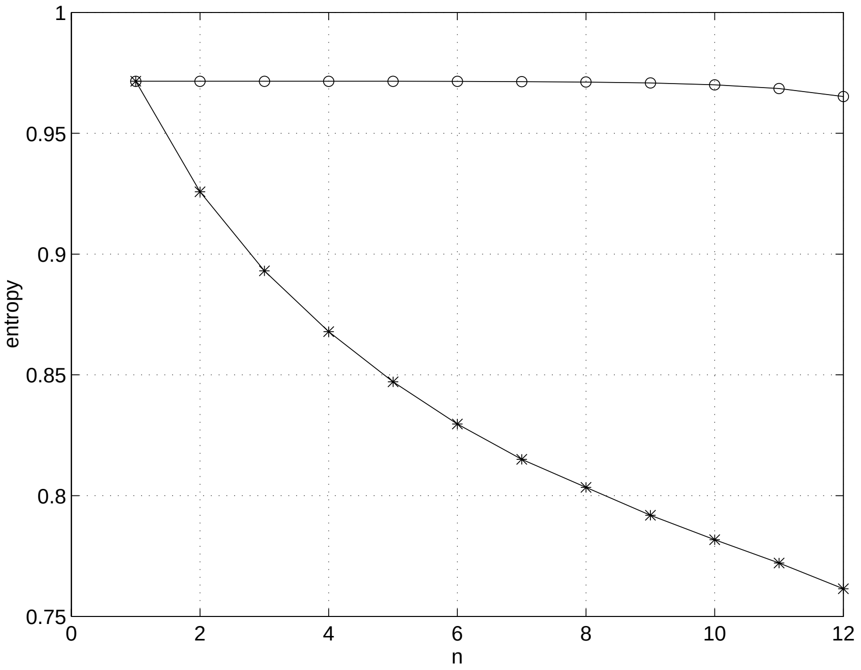


Figure 5.9: Entropy rate of U (o's) and O (*'s) as function of n.

code achieves the lowest possible bit rate. More important is the fact [21] that if an algorithm can achieve a rate of $H(U)+\epsilon$ for some $\epsilon > 0$, then it can also achieve a rate of $H(O)+\epsilon$. We can therefore use the difference between these entropies to measure the expected improvement. This difference is given by

$$\begin{aligned} H(U) - H(O) &= H(U) - H(U|R) \\ &= I(U;R) \\ &= \sum_{u \in \mathcal{U}} \sum_{r \in \mathcal{R}} P_{UR}(u,r) \log_2 \frac{P_{UR}(u,r)}{P_U(u)P_R(r)} \end{aligned} \tag{5.8}$$

The joint probability of U and R is $P_{UR}(u,r) = P_U(u) \times P_R(r)$ since they are independent. $P_U(u)$, defined in (5.4), can be simplified to

$$I(U;R) = \frac{1}{4} \sum_{s=0}^{1} \sum_{r=1}^{4} P_{S_r}(s) \log_2 \left(\frac{4P_{S_r}(s)}{\sum\limits_{i=1}^{4} P_{S_i}(s)} \right) \tag{5.9}$$

For the system shown in Figure 5.8 the probabilities $\{P_{S_1}(s), ..., P_{S_4}(s)\}$ are specified in Table 5.1. The entropies can be calculated from (5.5) and (5.7) and the improvement can be calculated from the mutual information of (5.9). The calculated entropy of U is $H(U) = 0.971$, the entropy of O is $H(O) = 0.6508$, and the mutual information is $I(U;R) = 0.3202$. This translates into an improvement of almost 33%, which implies that on the average one can expect to need 33% less storage for coding O versus coding U. In other words, if a compressor were to compress U to 100 bits, compressing O would yield 67 bits on the average.

To determine the improvement resulting from distance-based ordering, we need to consider an arbitrary probability distribution $P_R(r)$ where R is the distance value indexed by the distance transform map. By examining an image and counting the distance indexes used for ordering the residual pixels of all components, we can estimate $P_R(r)$. For the distribution of binary sources, all residual pixels are considered and for each distance $R = r$, we estimate $P_{S_r}(s)$. To use these distributions in our analysis we relax the requirement of a fixed number of binary sources and a uniform selector. Given $P_R(r)$, $P_{S_r}(s)$, and a maximum observed distance, R_{max}, we calculate

$$\begin{aligned} P_U(u) &= E_R\left[P_{S_R}(u)\right] \\ &= \sum_{r=1}^{R_{max}} P_R(r) P_{S_r}(u) \end{aligned} \tag{5.10}$$

$$H(U) = -\sum_{s=0}^{1} P_U(s) \log_2 P_U(s) \tag{5.11}$$

$$\begin{aligned} H(O) &= H(U|R) \\ &= E_R\left[H(S_R)\right] \\ &= -\sum_{r=1}^{R_{max}} P_R(r) \sum_{s=0}^{1} P_{S_r}(s) \log_2\left(P_{S_r}(s)\right) \end{aligned} \tag{5.12}$$

and

$$\begin{aligned} I(U;R) &= H(U) - H(U|R) \\ &= \sum_{s=0}^{1} \sum_{r=1}^{R_{max}} P_R(r) P_{S_r}(s) \log_2\left(\frac{P_{S_r}(s)}{P_U(s)}\right) \end{aligned} \tag{5.13}$$

Cumulative event counts are used to calculate an average $P_R(r)$ and $P_{S_r}(s)$ over a set of images to get a better estimate for $H(U)$, $H(U|R)$, and $I(U;R)$. Unfortunately, since we have only a limited number of residual pixel observations, especially for large distances, this method may result in artificially inflated probability values and may yield a skewed measure of entropy. Nevertheless, in practice we have observed improvements in excess of 10%, as will be shown in Section 5.5.

To calculate empirical values of entropy from (5.10)-(5.13), we estimate $P(R)$, the probability of the pixel sample belonging to source R, and $P_r(u)$, the probability of source r having value u. By counting how many distance values $r \in \{1, ..., R_{max}\}$ were observed, we can estimate $P(R)$ by dividing the counts by the sum of all counts. Figure 5.12a shows the overall distribution of R, calculated cumulatively over 120 images. Calculating $P_r(u)$ requires counting 1's and 0's separately as a function of associated distance value r. The plot of $P_r(u = 1)$ is shown in Figure 5.12b. Using our distance model we expect decreasing probability as r increases. This is observed primarily for distances less than 2000. The reason for the unusually high values of the probabilities observed for $r > 2000$ is the low number of observations at those distances.

5.5 Results

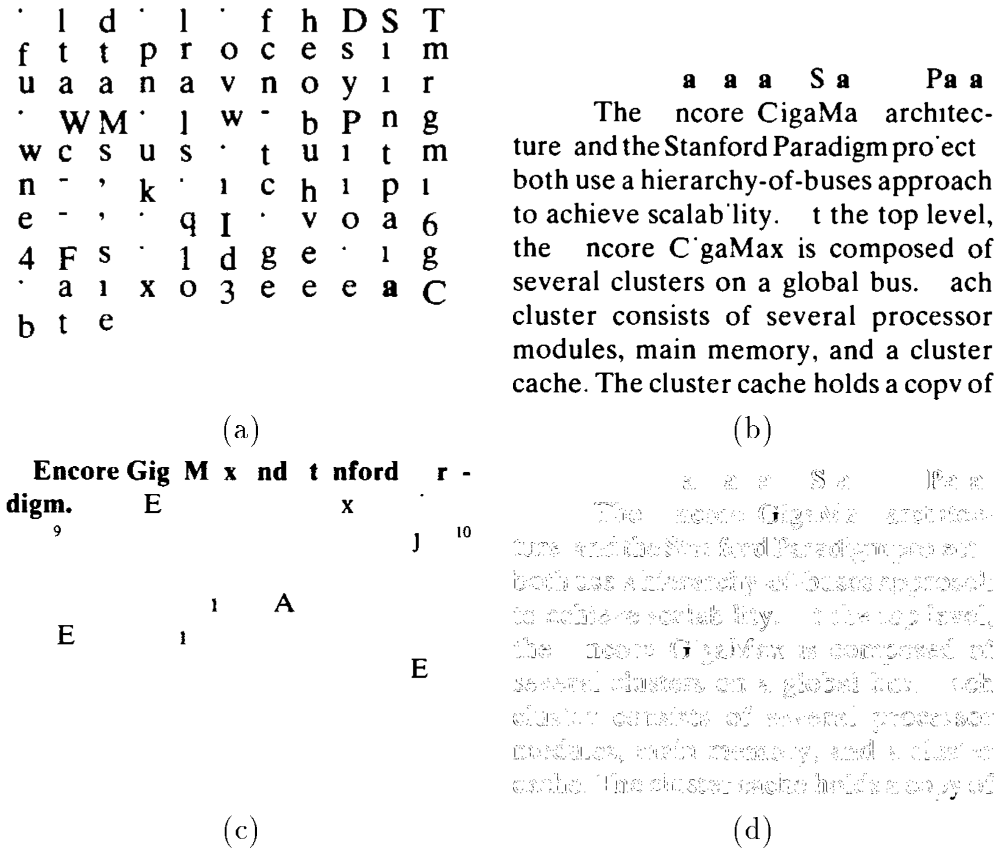
To test the effectiveness of our structural coding methods, we used the scanned images in the database and performed symbolic compression [54]. We extracted the residual information associated with non-NULL prototypes (NULL prototypes are those classified as graphics; see [50, 54]), and coded the residuals based on the distance, continuous, and packet mode orderings (using packet sizes of 20). We measured their entropies by simulating the entropy rate, calculating the conditional probability densities, and using (5.10)-(5.13). Figure 5.10 shows a set of prototypes obtained from the symbolic compression, as well as the corresponding residuals. The residual coding methods discussed in this paper were applied to the residuals shown in Figure 5.10d, which constitute the majority of the represented information.

We first present the results of the entropy rate simulation described in Section 5.4.1 and 5.4.2 and then present the empirical results described in Section 5.4.3 for the ordering mechanisms discussed in Sections 5.2, 5.3.1, and 5.3.2. Given a set of ordered residual streams we calculate the entropy of (5.2) for $\mathcal{X} = \{0,1\}^n$ and increasing n. Figure 5.11 shows the simulated entropy rates up to order $n = 10$, obtained by extracting the residual components of five document images (a total of 1 million residual pixels). It is obvious that the limit of the entropy rate of the ordered code cannot be determined from this graph, but it can be estimated using our empirical tools. Note that the unordered code exhibits a downward trend which implies the existence of a higher-order process. The reason for this is that the "ordered" results were actually based on row-ordered code, and the components exhibited correlation between the first rows of residuals belonging to the same prototype.

Figure 5.13 shows a plot of entropies for 120 document images and Table 5.2 summarizes the empirical results. To estimate the overall improvement, $H(U|R)$ for each ordering mechanism was compared to $H(U)$ for row ordering, yielding the results of Table 5.3, where 17 to 27 percent improvement was observed across all coding mechanisms and entropy measurements. Figure 5.14 shows the entropies of the codes obtained by the continuous and packet-mode ordering methods. The horizontal lines denote the average and cumulative values of the entropy for the unordered codes.

5.6 Summary

We find that the most important contribution to the residuals is due to symbol degradation. By extending existing degradation models [48, 91, 72] we arrive at a model that takes residual pixels less likely to be noise as informative pixels. By ordering the pixels in decreasing order of distance, we are able to classify the residual pixels into groups that have relatively stationary distributions. A model using independent binary sources is used to justify the distance ordering. We have shown that a code that


(a)
(b)
Encore Gig M x nd t nford r -
digm. E x
(c)
(d)

a a a S a Pa a

The ncore CigaMa architecture and the Stanford Paradigm pro'ect both use a hierarchy-of-buses approach to achieve scalab'lity. t the top level, the ncore C'gaMax is composed of several clusters on a global bus. ach cluster consists of several processor modules, main memory, and a cluster cache. The cluster cache holds a copv of

Figure 5.10: Example of a) a set of prototype maps, b) symbolic representation, c) components that were classified as graphics ("NULL" prototypes), and d) residuals of components that were assigned to non-NULL prototypes.

Cumulative	$H(U)$	$H(O)$	$I(U;R)$
Distance	.3407	.2828	.0579
Structural	.3226	.2714	.0512
Packet	.3224	.2712	.0512
Average	$H(U)$	$H(O)$	$I(U;R)$
Distance	.3626	.2980	.0646
Structural	.3244	.2636	.0608
Packet	.3246	.2637	.0608

Table 5.2: Summary of entropy values, in cumulative and average versions, for all coding methods.

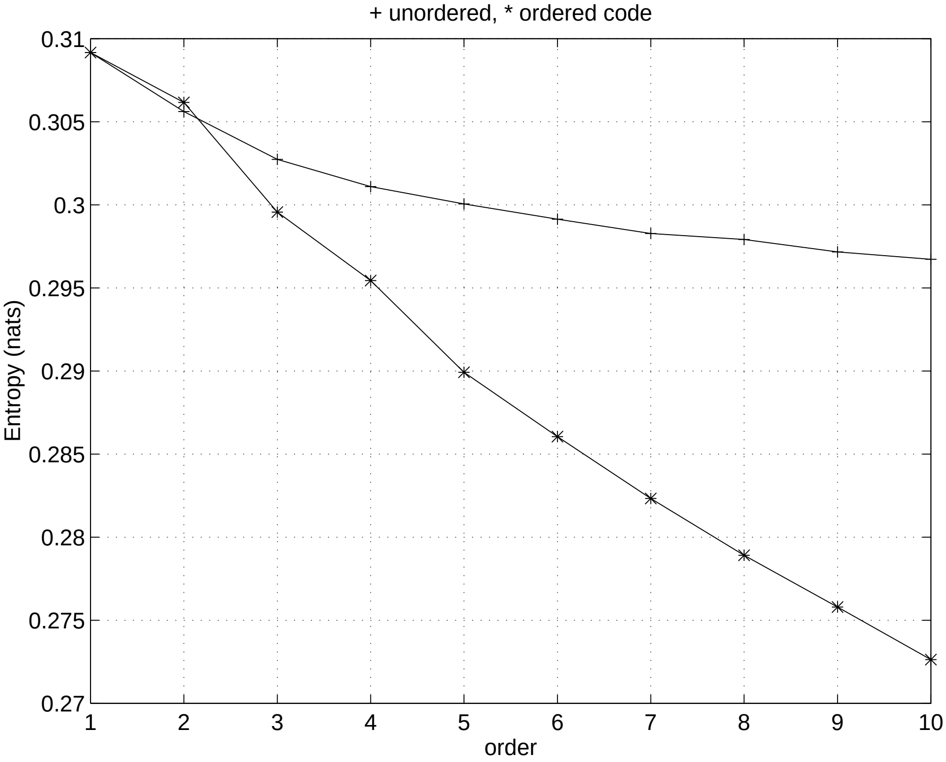
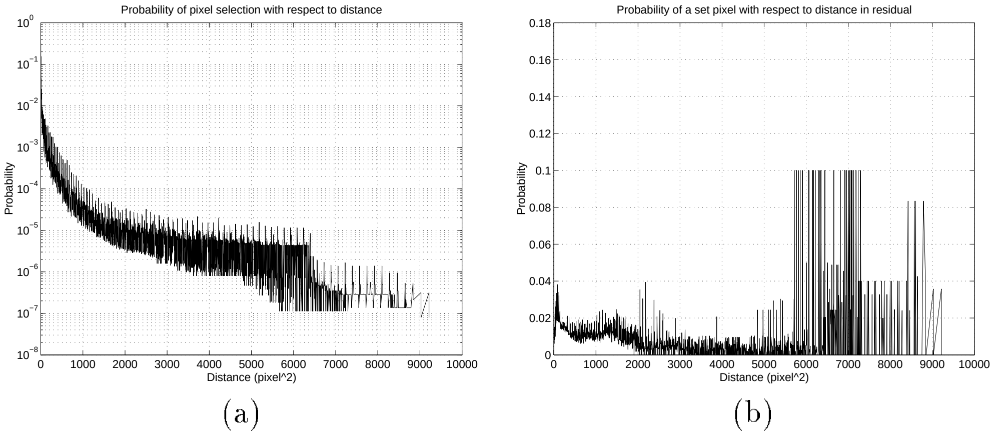
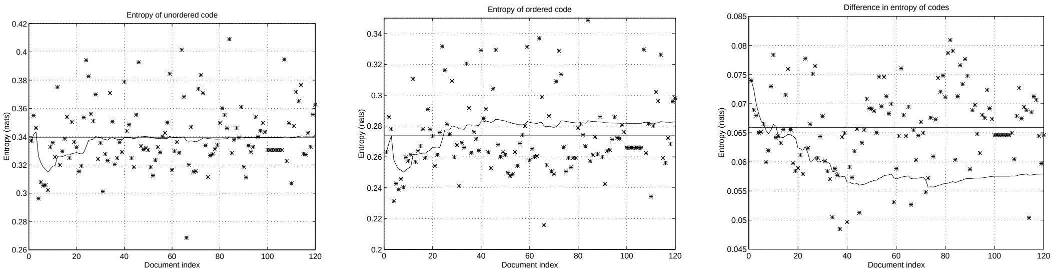


Figure 5.11: Entropy rates for unordered and distance-ordered coding.



(a) (b)

Figure 5.12: Probability distribution function of a) selector used for ordering the code and b) the binary distribution for each selection as calculated cumulatively over 120 document images.



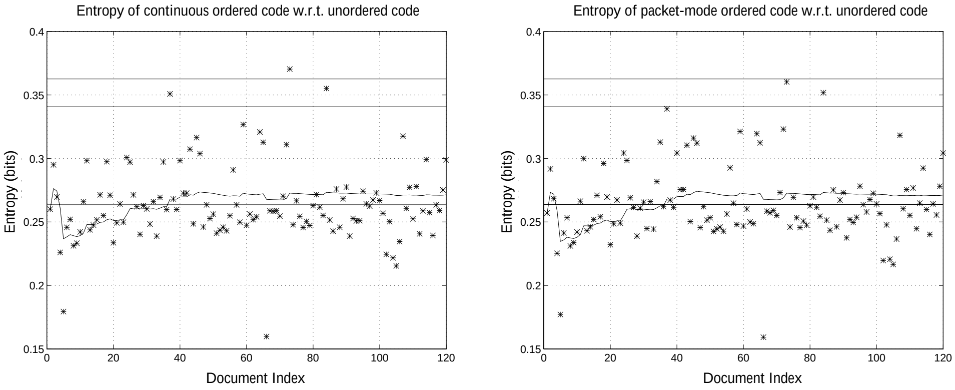
(a) (b) (c)

Figure 5.13: Comparison of codes in terms of information: a) entropy of the unordered code U, b) entropy of the ordered code O, c) improvement in entropy (the mutual information between the unordered and ordered codes).

Percent improvement

	Cumulative	Averaged
Distance	17%	18%
Structural	20%	27%
Packet	20%	27%

Table 5.3: Improvements in coding methodologies with respect to unordered code.



(a) (b)

Figure 5.14: Entropy of a) continuous and b) packet-mode structural coding, along with their cumulative and average values and the cumulative and average values of entropy for unpredicted and unordered code.

does not use the probability assignments has higher entropy and less compressibility. Furthermore, the group distributions need not be known; the fact that statistically similar pixels are grouped together is sufficient to provide higher compression. An important characteristic of this method of residual coding is that the distance ordering depends only on the prototypes and is therefore available to both the transmitter and receiver.

The notion of structure has been considered as a method of assessing the contributions of pixels to structural components. Distance-ordered coding provides an important basis for predicting this structure. A predictive method is suggested that makes use of the fact that structural components are connected. Any pixel far away from an edge makes a high contribution to readability; but connected sets of pixels form structural components. Packet-mode prediction, which can be regarded as a tradeoff between distance ordering and predictive ordering, provides a more flexible approach than pixel-by-pixel prediction.

As a generalization of our method, if a probabilistic model of the residual information can be determined for a lossy compression algorithm, it should be possible to benefit from our coding framework. A lossy algorithm provides a base image which can then be used to estimate the residual order based on the probability model. This can apply to images beyond the document image domain.

Chapter 6

SCoDI: Symbolic Compression of Document Imagery

6.1 Introduction

Symbolic Compression of Document Images (SCoDI) is an implementation of the techniques described in the previous chapters to create a framework for document image compression using the symbolic approach. We have previously discussed the general approach and its data representation requirements along with detailed discussions of the segmentation, clustering, and residual coding tasks. The interrelationship between these components needs to be mentioned in order to arrive at a comprehensive description.

During segmentation we rely on connected components to provide a set of patterns that resemble an underlying language and that can be used in a clustering algorithm to arrive at prototypes which resemble elements of the language. Connected components are a naive approach to achieving this result. In Chapter 4 we showed that obtaining good prototypes does depend on the segmentation and eventually affects compression and processing performance.

In our system we use a pattern matching and classification algorithm which basically tests the amount of match between a presented pattern and one that is stored as a cluster prototype. If a match exists then we assign the pattern to the cluster; otherwise we create a new cluster and let the prototype be equal to the presented pattern. A large number of matching functions are considered and their effect on the performance of our system is considered.

Finally, the task of residual coding is considered. Since the majority of the compressed information is in the residual code, it is necessary to address concerns about its coding. We propose methods that organize the residuals in a way that is compact (compressible), hierarchically organized (compressible and processable), and associated with their symbolic counterparts (processable).

In implementing a system that can take advantage of our representation, we must create some building blocks. The categorization of these blocks provides a detailed road map for future research and enhancements to the system, since they form the fundamental architecture for our approach. The overall structure is shown in Figure 6.1. While the baseline idea is a derivative of pattern matching and substitution (PMS) algorithms [10, 64], this representation provides a general framework for achieving compression while allowing for compressed-domain processing.

In this chapter we will discuss the implementation issues we encountered. Specifically, we discuss segmentation in Section 6.2, clustering in Section 6.3, component representation in Section 6.4, and residual coding in Section 6.5. We address some compressed-domain processing requirements in Section 6.6, and we summarize our

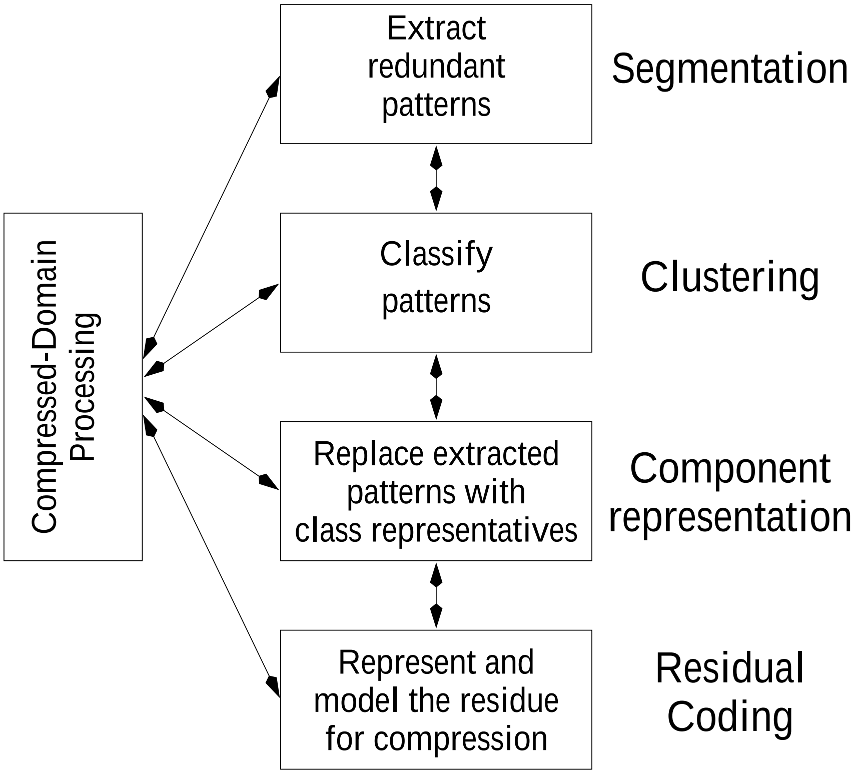

Extract redundant patterns
Segmentation
Classify patterns
Clustering
Compressed-Domain Processing
Replace extracted patterns with class representatives
Component representation
Represent and model the residue for compression
Residual Coding

Figure 6.1: Overall design structure for symbolic compression of images.

implementation in Section 6.7.

6.2 Segmentation

The first step in our method, as described in Section 2.1, is to extract patterns from an image. Segmentation is a generic term and is highly dependent on image type and characteristics. For bi-level document images, a simple and effective segmentation method is extraction of connected components. The bounding boxes for connected components then should contain the patterns that repeat in the document image. In our approach we have implemented a connected component analysis routine which scans a pixel row of image and tags runs of foreground pixels. The same process occurs on the succeeding scan-line with the exception that tags which are equal across scanlines are combined. Updating corner locations provides the bounding boxes.

Other segmentation methods can be pursued to extract repetitive patterns. An example of such a technique is discussed in Chapter 3 where given a set of prototypes we re-evaluate the components to arrive at a better segmentation. This method is especially useful for degraded images where components have started to touch each other and the representative prototypes are images of touching characters. For an extension to grayscale images, it is desirable to use the grayscale information to achieve a better segmentation. In effect, a good segmentation is one that provides fundamental patterns within the image that are redundant. In terms of compressed-domain processing, an effective segmentation is one that extracts usable information to be accessed later. In the case of document images, the connected-component segmentation allowed for extraction of redundant patterns and usable information.

6.3 Clustering

Once a suitable segmentation is obtained, a clustering algorithm needs to be used to determine similarities between the extracted patterns. In general, a clustering algorithm needs to operate on the segmentation output and to capitalize on pattern redundancy while behaving robustly when inconsistencies occur. For example, a situation that occurs in document image segmentation and clustering is that the extracted patterns were of different sizes, so our clustering algorithm was required to provide a degree of robustness to the input component sizes. A number of clustering algorithms are considered in Chapter 3, and a suitable one is used for our compression.

In our implementation we input the bitmap image, the bounding boxes of the connected components, and a threshold value to a routine. The routine scans all the components and matches them against the prototype images. Each match is compared to a threshold, normalized by the size of the the component (i.e. if the number of matched pixels is above a percentage of the component size, the component is classified as belonging to that cluster). If a close enough match occurs the component is classified in the corresponding cluster, otherwise it is used to create a new cluster. After a number of components are added to a cluster, the cluster prototype is recalculated. In our system we recalculate prototypes every time five components are added and do not recalculate after the tenth calculation. To avoid generating a large number of prototypes, if the ratio of the number of prototypes generated to the number of components visited exceeds by fifty percent the ratio of the maximum number of prototypes to the total number of components, the threshold value is increased linearly by this amount and the clustering is repeated. By starting with a small threshold we can end the clustering process with an adequate number of clusters.

The relationship between segmentation and clustering is more complicated than we have indicated here. In our baseline approach, clustering is dependent on segmentation, but as hinted in the previous section, segmentation can be improved if a clustering result is available. In Chapter 4 we discussed a joint segmentation-clustering approach which not only improves the compression results but promises to improve processing performance. In essence there should exist a higher-level process overseeing the segmentation and clustering tasks, making sure that they access each other's intermediate results, as shown in Figure 6.2.

6.4 Component representation

Once we have determined the clusters, it is relatively easy to define a representative prototype for each cluster. In our method we averaged the members of each cluster to calculate its prototype image. Hobby and Baird call this “naive averaging” [34]. Zhang and Danskin [116] use an alternative method based on scaling and filtering.

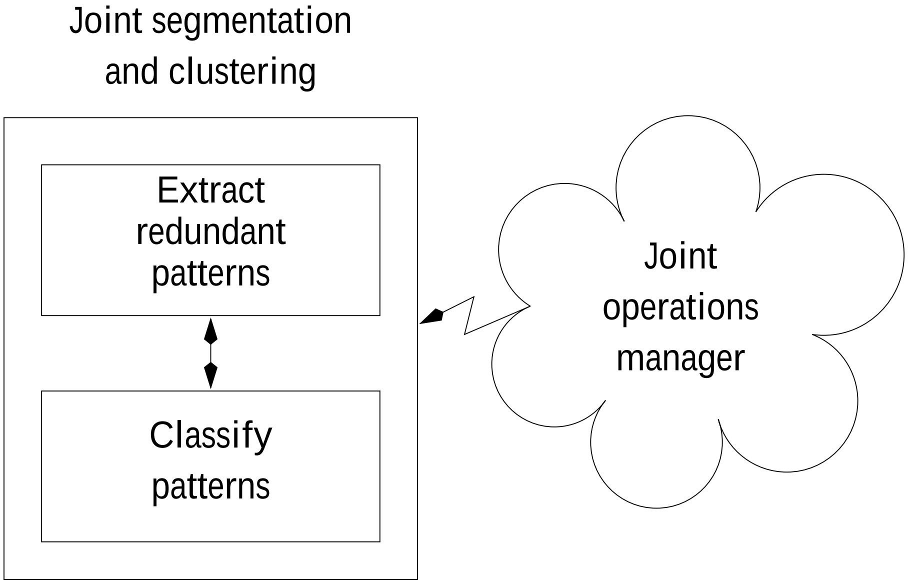


Figure 6.2: Joint segmentation and clustering relationship.

Our "naive averaging" has produced good results mostly due to the highly structured character layout of document images.

Given the prototypes, we need to represent their locations within the document image. To preserve access to the prototypes (i.e., to the components that are represented by them), we give the address of each component, as described in Section 2.2.1. This representation is somewhat redundant. A highly structured document image has components which lie on lines. It is therefore much simpler to give relative addresses of components rather than absolute addresses. Work done by Witten et al. [112] discusses the optimization of component indexing in this context; however, due to the small storage space needed to record the addresses, this was not pursued. It is also possible to pursue a rate-distortion analysis with regard to component indexing, that is to say, provide addresses which are not exact and allow for a certain level of distortion, while achieving a higher compression ratio.

6.5 Residual coding

It is possible to achieve very high compression by substituting for each segmented patterns its associated prototype. In document image compression, this is effective to a great extent; however, exact rendition (at least for reading purposes) cannot be guaranteed since there always exist ambiguous characters which are assigned to the same cluster. Readability is not effectively achieved if too many 'e' characters are replaced by 'o' characters, even though we can effectively read the 'o' characters. To provide graceful degradation in reading quality, we make use of a residual code.

In our experiments we observed that the largest amount of information is contained in the residual representation. To achieve a more efficient representation we pursued a model-driven approach to achieve compression and to provide graceful degradation when residues are partially represented. This is logical in that given a segmentation and clustering, each prototype represents a class of observations which is defined by the variability in the prototype's cluster. This variability is finite and has a specific shape, so for a class of images (to be compressed) it is possible to model the variations effectively to provide a hierarchical representation for use in graceful degradation of tasks like lossy representation and progressive transmission. In our implementation we take as input the original bitmapped image, the prototypes, and the component memberships and create a stream in which residuals of components are appended and indexed according to their order of appearance. First the prototypes are used to generate their distance transforms, and these distance transforms are used to order the residuals for distance-based coding. For structural coding, we used the residuals, prototypes, and component memberships as inputs to Matlab routines and measured their progressive entropy by coding the residuals based on continuous and packet-mode coding. Special implementation of residual coding is done to allow for packing of bits into bytes while keeping track of where the residual information for components starts and finishes, without wasting bits and keeping the information accessible after performing the Huffman code. This is done by keeping a byte and a bit counter for each residual set, Huffman coding the set, and recalculating the byte and bit counter after coding.

While the complexity of the residuals is dependent on the quality of segmentation, clustering, and component representation, it is easy to see that their relationships cannot be easily modeled to fit a the framework shown in Figure 6.3. The only plausible outlook for achieving such a "Global Manager" is to model the effects of individual components so that one component need not wait for the effects of others to propagate. For example, a segmentation that is based on existing clusters should also consider the number and complexity of the residuals and the component layout in order to make an optimal determination of the correct segmentation. If a global model is derived that can be trained, a global manager can use it for optimal interaction between components.

6.6 Compressed-domain processing

Our eventual goal is to perform a set of compressed-domain processing tasks. In our implementation we created a set of routines with generic functions. One function was to compress and decompress simple streams. A second routine compressed a stream while modifying its indexes; this usually involved simple compression of the indexes. A third routine decompressed only a section of the stream. Using combinations of these routines we are able to feed appropriate portions of the information to specific document image processing tasks. Since we are able to divide the information into logical partitions, we have access to individual information sections, as required by the processing elements and by the need to perform extra decompression. For large information sets, we are able to index them so that unnecessary decompression and processing is minimized.

Since most of the information in document images lies in the components, we are able to achieve compressed-domain processing. This is not readily obvious for other

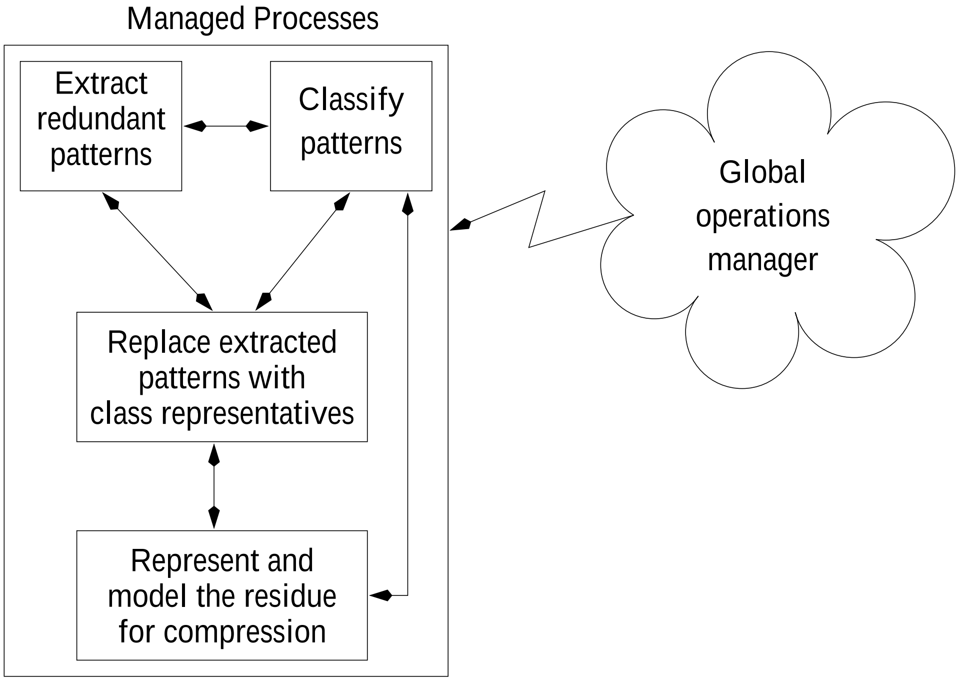

Managed Processes
Extract redundant patterns
Classify patterns
Replace extracted patterns with class representatives
Represent and model the residue for compression
Global operations manager

Figure 6.3: Management of compression modules to achieve a global goal.

classes of images. However, it is possible to identify a class of images that are to be used for a specific purpose, and model their information accordingly so that the dissemination of information is done efficiently. Such situations abound in today's networked databases, and in server-client information exchange systems.

6.7 Summary

The implementation of our compression system has been discussed in this chapter. Detailed discussions of the implementation of segmentation, clustering, residual coding, and component representation are given to support our earlier claims about compression and processing. At every step the implementation is discussed together with supporting issues and concerns, as well as suggestions for improvement.

We discuss and explain our algorithmic approach to connected component analysis and bounding box calculation for segmentation. It is important to note that this analysis is ideal for binary images that have small amounts of degradation. Earlier we discussed alternate methods of segmentation under heavy degradation and examined how these methods fit into our compression scheme. We observed that in presence of degradation an operations manager can provide control over better segmentation and clustering.

Residual coding and its implementational concerns is also discussed. We suggest that for an optimized system interaction between subcomponents of the compression system needs to exist in order to allow them to complement each other's weaknesses and strengths. Such an elaborate system is only possible with current computational technology. Finally, we discuss the general implementation issues of performing compressed-domain processing tasks.

Once our system is implemented, we can consider testing its performance at compression and processing. These validations include rate-distortion analysis and compressed-domain processing applications.

Chapter 7

Rate-distortion analysis

7.1 Introduction

Two common problems in compression research are achieving lossy compression and providing a representation for progressive transmission. The performance of any solution is characterized by a rate-distortion trade-off in which we attempt to optimize the correct recognition of image contents. Using symbolic compression in document images we have shown that compression can be achieved while allowing for compressed-domain processing [54]. While the ability to perform progressive transmission and scalable lossy compression has also been demonstrated, a rigorous rate-distortion trade-off analysis is necessary. Since the symbolic representation contains distinct data types with different characteristics, rate-distortion analysis should be performed on all parts and be integrated coherently to promote a comprehensive measure. We perform the analysis here only on the residual section of the data, where about 80% of the data is contained.

The problem that must be addressed is how to define an appropriate distortion function for document images. The traditional signal-to-noise ratio and its derivatives are not desirable. For a constant power level there exist different representations that render varying amounts of the image. Pixels carry varying amounts of information which affects the correct recognition of the image components. The distortion function should reflect the correct recognition of the lossy representations. Using an OCR engine and OCR evaluation software, we were able to link the correct recognition of an image with an associated entropy. This was also compared to distance-weighted distortion.

This chapter is organized as follows. In Section 7.2 we provide a brief motivation for our approach. In Section 7.3 we discuss in detail our procedure and results. In Section 7.4 we provide similar results based on pixel-based matching (distance-based matching) instead of the OCR-based approach. We conclude in Section 7.5 with some final remarks.

7.2 Motivation

In the previous chapters we have argued that since characters degrade around the edges [48, 72, 91] the pixels far from edges contribute more toward correct recognition than pixels close to edges. This was also supported by the fact that most degraded images are still recognizable. It is however not trivial to extend this concept to address rate-distortion concerns. This is because the concept of distortion is eventually related to correct recognition which is hard to measure from a presented image. Here we use

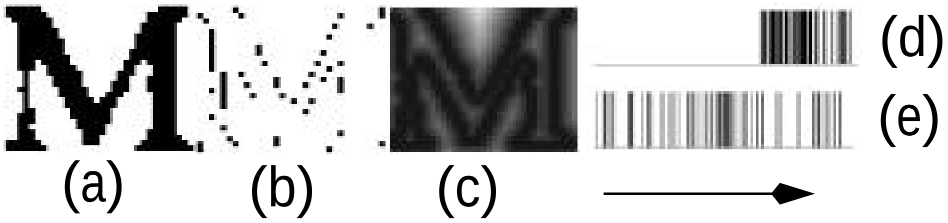


Figure 7.1: Example of a) an observation, b) a residual map, c) the distance map of the prototype, d) distance-ordered residual code, and e) row-ordered residual code. The codes in (d) and (e) are ordered from left to right.

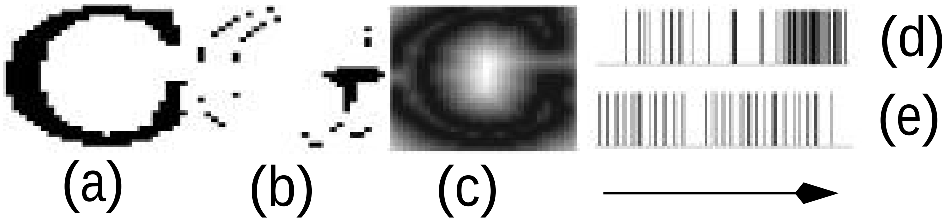


Figure 7.2: Example of a) an observation, b) a residual map, c) the distance map of the prototype, d) distance-ordered residual code, and e) row-ordered residual code. The codes in (d) and (e) are ordered from left to right.

OCR to provide a measure of correct recognition of components from a presented image.

7.2.1 Hierarchy

A typical observation, residual map, distance transform, distance-order code, and row-order code are shown in Figures 7.1 and 7.2. It is easy to see that the residual pixels in Figure 7.1b do not contribute significantly to recognition, whereas the residual pixels in Figure 7.2b do. For the distance order, the residuals of Figure 7.1b are coded towards the end of the stream, as shown in Figure 7.1d, while for the row-ordered stream they are spread over the entire stream (Figure 7.1e). For the distance-ordered residual of Figure 7.2d, the pixels which distinguish the 'C' from the 'G' are coded first, in contrast to the row-ordered residual code of Figure 7.2e which contains these pixels in the middle of the code. Examples of residuals are shown in Figure 7.3. Letters 'E' and 'i' were classified as graphics, 'G' was classified in a 'C' prototype, and the rest of the components have only silhouette residuals.

Figure 7.4 shows an example of extreme lossy/lossless representations with Figure 7.4a requiring a lower entropy than Figure 7.4b. A coding method interpolates between these extremes to provide an intermediate representation of a desired entropy and an associated distortion. The point of a hierarchy is that given a representation

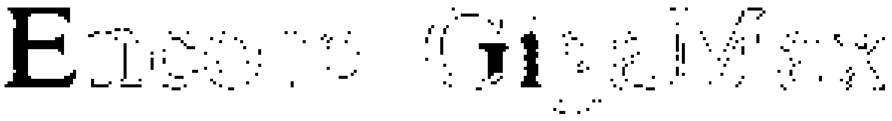
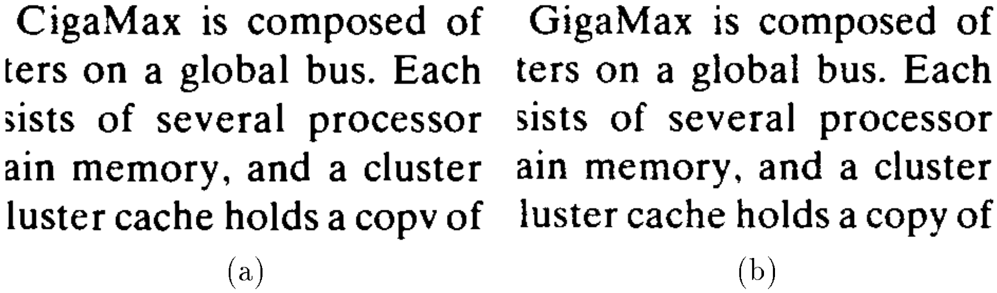


Figure 7.3: Examples of various types of residuals.



CigaMax is composed of
ters on a global bus. Each
sists of several processor
ain memory, and a cluster
luster cache holds a copv of

(a)

GigaMax is composed of
ters on a global bus. Each
sists of several processor
ain memory, and a cluster
luster cache holds a copy of

(b)

Figure 7.4: Section of document image rendered with (a) almost no residuals and (b) almost all residuals.

with a rate-distortion pair of (r, d) it is desired to achieve $(r + \delta r)$ entropy and $(d - \delta d)$ distortion by transmitting only information of δr entropy. A simple hierarchy is created by traversing from left to right in the distance-ordered and row-ordered codes of Figure 7.1. For an intermediate representation we use a fractional part of the residual code. The hierarchy can then be tested by comparing the rate-distortion curves of forward and reverse ordering; forward ordering should clearly perform better than reverse ordering.

7.2.2 Analysis

Document images are intended for reading and we have therefore chosen the distortion to be a function of recognition as defined by OCR accuracy. We first need to determine the baseline text by considering either ground truth data or the OCR results on the lossless image. For each intermediate representation we perform OCR. We then evaluate the performance of the OCR output with respect to the baseline text by string matching [17, 20, 56].

Figure 7.5 shows examples of the words “GigaMax” and “copy” in three different instances yielding almost equal entropies. It is clear that distance-ordered code (Figures 7.5a and b) is likely to produce fewer OCR errors than reverse distance-ordered code. Row-ordered code will have errors, but since reconstruction of the letter ‘G’ has started, it may perform better than reverse distance-ordering. Errors are not widespread even in the most lossy representation. This is ideal since in Chapter 8 we will show that many document image processing tasks can be done without the use of residuals and that the OCR engine will be working at high accuracies with reliable

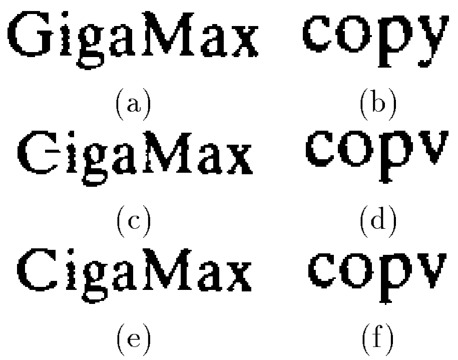


Figure 7.5: Example of portions of an image (Figure 7.4) with the use of roughly equal entropy in lossy rendition using (a,b) distance ordering, (c,d) row ordering, and (e,f) reverse distance ordering.

output. OCR evaluation is done by string matching by use of available software which outputs the numbers of matched, deleted, inserted, and changed symbols. We define the inverse of distortion (performance) to be the number of deleted, inserted, and changed symbols minus the number of matched symbols.

7.3 Procedure and results

In determining the rate-distortion relation, two procedures were used. The first was to render an intermediate representation with an associated entropy and to perform OCR. The second was to compare the OCR outputs with the ground truth data and determine the OCR distortion. In creating the intermediate representations, we used the University of Washington [32] images and represented them using symbolic compression. Given fractions of the residual codes were zeroed (for each component), and the distribution of the residuals (number of residuals and number of zeros) was computed. Only the residuals that corresponded to components assigned to a prototype were used in the intermediate representation. This was necessary to avoid contamination by components that were not clustered appropriately. The temporary representation was then fed to an OCR engine (ScanWorx by Xerox, API software) and the text output was saved for later comparisons. The OCR outputs were evaluated using software available in the University of Washington database [32]. The evaluation was done with respect to the ground truth data with the use of zones to avoid contamination by graphs and images. For each representation set we determined the achieved entropy, and using the evaluation result we determined a rate-distortion relation.

Since a linear increase in the fraction of residual code does not translate into a linear increase in entropy we had to specify fractional amounts that would provide a better distribution of points on the entropy axis. This was done for distance ordering only. The plot of distortion with respect to entropy is shown in Figure 7.6. This result was obtained on 72 images containing 214,292 ground-truthed characters. The

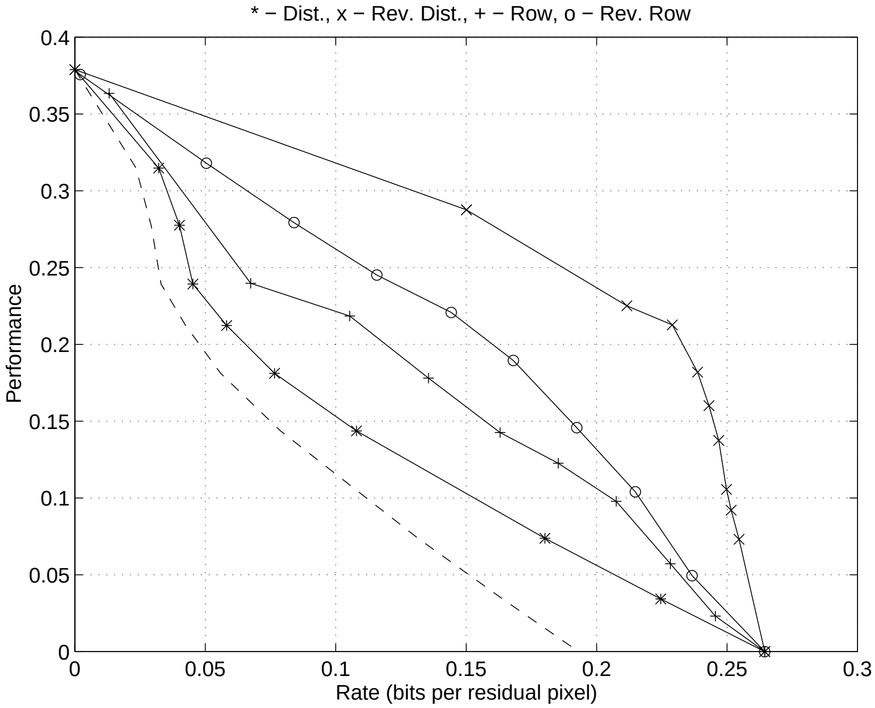


Figure 7.6: Rate-distortion plot. Symbol '*' denotes distance ordering, 'x' denotes row ordering, and '+' denotes reverse distance ordering.

performance of distance ordering is clearly superior to that of the other codes. An interesting observation is that distance ordering clearly performs better than reverse distance ordering. In the case of row and column ordering, that their reversals perform neither better nor worse. This suggests that row or column ordering does not provide an adequate representational hierarchy. The fact that reverse distance ordering does much worse than forward distance ordering suggests that there exists an inherent representational hierarchy. Another observation is that in Chapter 5 we were able to achieve an 18% to 27% improvement in entropy over distance ordering by performing structural coding. The expected amount of improvement is shown in Figure 7.6 in the form of dashed lines. The dashed line only reflects the improvement in entropy, but structural coding should also improve distortion (OCR on distance-based).

Figure 7.6 shows the relationship between distortion level and achieved entropy rate as partial residual coding progresses using different coding schemes. This result indicates the range of coding efficiency achieved by the coding methods considered here. While the rate-distortion function cannot be derived from these results, it can be roughly estimated. The rate-distortion function gives the minimum distortion at a given entropy rate. It is unrealistic to consider all combinations of residuals that have the same entropy and determine their distortions, due to the large amount of processing needed for residual representation and for OCR. However, each bit order can be used to model the best possible representation given an entropy rate. All possible residual codes that achieve a given entropy can be replaced by a single code, ordered in a hierarchy, that has that entropy rate. The possible improvement is shown in dashed lines in Figure 7.6. The functions shown in Figure 7.6 need not be strictly convex for two reasons. The first reason is that they are based on empirical data. The second reason is that the distortion is averaged over a large number of images that vary in quality; while convexity may be observed for individual images, it may be lost over a set of images.

7.4 Distance-based rate-distortion

Distance-based ordering has natural extensions to lossy compression and progressive transmission applications. In the case of lossy compression, the residual stream can be terminated at a predefined point. Recall that in the previous sections, we showed that for lossy compression, the distance order residual is structurally more informative (has a lower distortion) and is more compact (has lower entropy rate). These properties are advantageous in progressive transmission as well as lossy compression. For each symbol and given code length we would like to retain maximal quality (i.e. lower distortion).

Figure 7.7 shows a typical rate-distortion function achieved for a sample image. This curve shows a significant change in distortion per unit entropy rate for entropy rates up to 0.0018 bits per residual pixel. An example of a lossy compressed component was shown in Figure 5.1b, and its associated residual map in Figure 5.1c.

The rate-distortion calculation using distance-based matching was done for both the unordered and ordered (continuous and packet-mode) codes. The distortion measure used was the same as the distance-based mismatch measure defined in [54], where the total distortion is measured as the sum of distances for the pixels that were not coded. The entropy rate, r, was calculated as

$$r = \frac{H_1(E) \times \text{Number of residual pixels}}{\text{Image size}} \tag{7.1}$$

in units of bits per pixel. For the case of unordered residual coding, the pixels are coded in row order. Figure 7.8 shows the rate-distortion functions for unordered, distance-ordered, continuous and packet-mode structural coding. The diagonal line represents the unordered code, for which the tradeoff between distortion and transmission rate is almost a constant. For distance ordering, we coded bits that contributed to the highest distortion first and calculated their rate as specified above. Continuous coding and packet-mode structural coding were also used to calculate distortion and achievable entropy rate. Figure 7.8 shows the performances of these coding mechanisms. The structural methods give the best result, slightly better than that of distance-ordered coding. Note that the packet-mode and continuous methods are not distinguishable in the plot shown in Figure 7.8.

This analysis is incomplete in that the measure of distortion is not comprehensive enough. The primary conclusion from Figure 7.8 is that distance ordering contributes much more toward reduction of distortion than row ordering. This is obvious, because

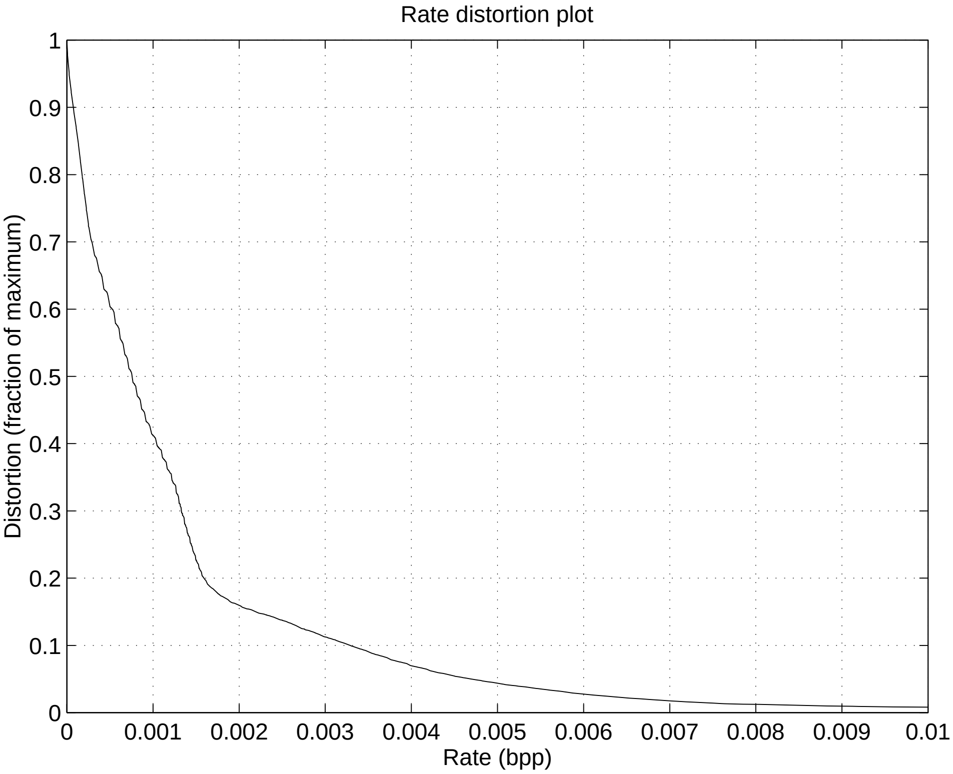


Figure 7.7: Plot of rate-distortion function for A006BIN.TIF image in database [32], by considering residual information due to components belonging to a prototype.

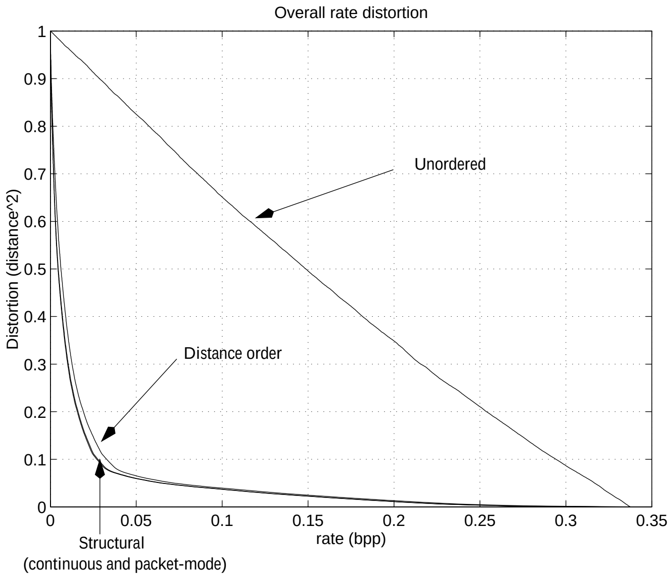


Figure 7.8: Rate-distortion plots of unordered, distance-based, continuous, and packet-mode structural coding.

the distortion is based on the distance transform. It remains to establish that the distortion measure used in Figure 7.8 measures something useful such as the reading quality of the image which in turn affects the correct recognition of individual symbols. In the previous section OCR errors were shown to be less for distance ordering than row ordering for the same entropy rate. It follows that the distortion measure used for Figure 7.8 may calculate a measure close to reading quality. An automatic choice of the best distortion-rate pair is not obvious in OCR-based distortion, but is quite easy in distance-based distortion.

7.5 Summary

In compression, it is often desirable to perform lossy compression and progressive transmission. Traditionally, algorithms which provide these capabilities use a rate-distortion plot to determine the exact compression factor achieved by allowing a fixed amount of distortion measured by the signal-to-noise power ratio. This is easy to do, since the signal is modeled by equiprobable samples and the amount of deviation from the model is taken to be the distortion by taking a simple distance measure.

For document images a signal model which can distinguish signal from noise does not exist. While a document image is a bit-mapped image, the useful information resides in the symbols found in it. Therefore a model must include the symbol representation of the document image. Here, we have proposed an OCR model to determine a distortion measure by performing lossy representation to achieve a fixed entropy. Our analysis has shown that the rate-distortion measure based on the ability of an OCR engine to recognize correctly is more favorable to distance ordering than to row ordering. Furthermore, the results suggest an inherent hierarchy within the code since the reverse ordering clearly performs more poorly than the forward ordering. The reason that reverse row ordering did worse than forward row ordering is that in our segmentation of components, we referenced the upper left corner of the bounding box, and in our prototype estimation we padded the lower right corner. This results in an informational preference toward the top of the component rather than the bottom, which results in reverse row ordering doing worse than forward ordering. This also agrees with the entropy rate of row-ordered residual coding computed in Chapter 5, in that the entropy is slightly reduced for higher entropy rates, implying a higher order process within the row-ordered code.

We have also considered a distance-based distortion function in determining the rate-distortion characteristics of our coding method. The rate-distortion tradeoff can be used to automatically represent a lossy image in terms of the lowest rate and distortion. Row-ordered coding is not likely to provide the lowest rate and distortion point since for a fractional entropy rate there exists fractional distortion across all entropy rates. On the other hand, the fractional measured distortion with respect to a fractional entropy rate is clearly unproportional across all entropy rates when we use the distance ordering. Distance-based matching and coding may eventually provide a measure of the OCRability of a given document image prior to performing actual OCR.

Chapter 8

Compressed-domain processing

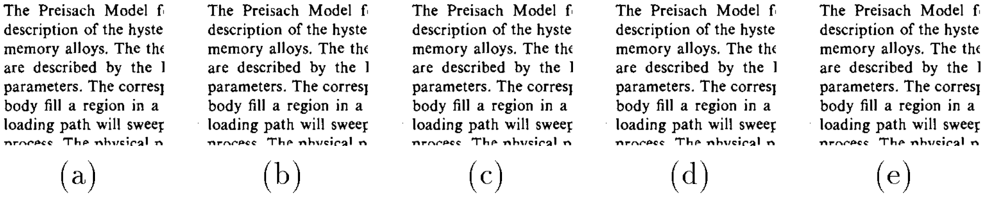
In the document domain, algorithms which implement CCITT Group 3 and 4 fax compression have been used widely. In the document domain, the measure of a good compression algorithm may have a number of parameters beyond the traditional space reduction. For example, we find that digital libraries which contain document images benefit not only from compression which reduces the amount of storage necessary, but also from the ability to process and search the underlying documents easily and efficiently.

Due to the large volume of document images and the computationally intensive tasks that need to be performed on them, there exists a genuine need for performing document image processing in an efficient manner. Compressed-domain processing has been able to address this by accessing features used by the compression algorithm and using them to satisfy processing needs, as was mentioned in Chapter 1. Symbolic compression of document images provides a rich source of these features and a large number of compressed-domain processing tasks can be done using them.

This chapter contains a discussion of a number of applications that use the representations described previously. In Section 8.1 and 8.2 we discuss the qualities of our representation when we perform lossy compression and progressive transmission. In Section 8.3 we demonstrate the ability to perform scalable retrieval for large document images. In Section 8.4 we show how skew estimation and correction can be performed while the data is in the compressed domain. In Section 8.5 we show how to perform keyword searching, and in Section 8.6 we consider the problem of duplicate detection in an image database. We conclude with final remarks in Section 8.7.

8.1 Lossy compression

A useful option in compressing an image is to be able to specify a quality of compression, or a required size limitation. In document image compression, it is hard to identify a measure of quality, which has to be related to the readability of the document. Most measures of quality are related to the overall “power” in the error image, defined by the difference between the lossless and lossy representations. We have found, however, that some pixels contribute more to the readability of a document than others, and even if two representations give the same “error power”, one may be more readable than the other. As described previously, in our representation we have ordered the residual pixels associated with each component and can use different fractions of the residual map stream to achieve different degrees of lossy compression. We take the rendition of the document without the use of any residual pixels as a baseline and add linearly increasing fractions of the residual stream.



The Preisach Model f
description of the hyste
memory alloys. The the
are described by the I
parameters. The corresj
body fill a region in a
loading path will sweep
process. The physical n

Figure 8.1: Lossy compression of images using a) 80%, b) 60%, c) 40%, d) 20%, and e) 1% of the error streams associated with the templates.

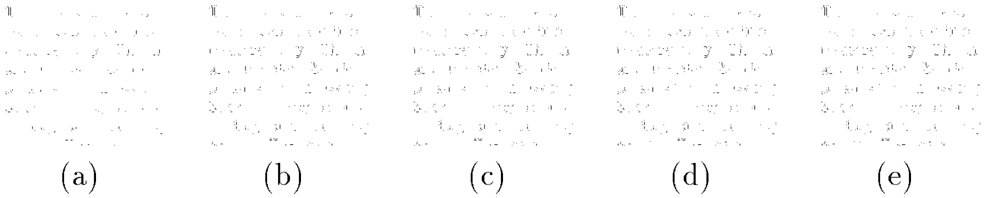


Figure 8.2: Difference between the original portion of the image and its lossy compression using a) 80%, b) 60%, c) 40%, d) 20%, and e) 1% of the error streams associated with the templates.

Figure 8.1 shows an image which has been lossily compressed to varying degrees, with (a) being the least compressed and (e) being the most. The percentage indicates the amount of residual information kept for each non-NULL-clustered symbol (i.e., 100% would yield a lossless image). Note that there is no observable difference between the least and most compressed images as regards readability. The residual images shown in Figure 8.2 show that the residuals lie mostly on the edges of the symbols, and that even at the highest levels of lossy compression, the rendition is quite readable without any compromise in resolution.

Figure 8.3 shows examples of lossless, symbolic-only, and lossy compression of an image using symbolic compression. It is not easy to see, but the symbolic-only representation contains an enormous amount of information about the content of the image. Figure 8.4 shows a comparison of symbolic compression with a leading contender that uses resolution reduction. Note the reduction in correct recognition of components within the image. Only a small portion of the image is shown to allow direct comparison.

For 122 images in the University of Washington document database we achieved an average compression ratio of 29.56 when we used 1% of the residual stream, 25.71 for 20%, 22.88 for 40%, 19.96 for 60%, and 15.67 for 80%. This is in comparison to lossless compression ratios of 12.2 and 17.8 for symbolic compression and for Group 4 standard respectively.

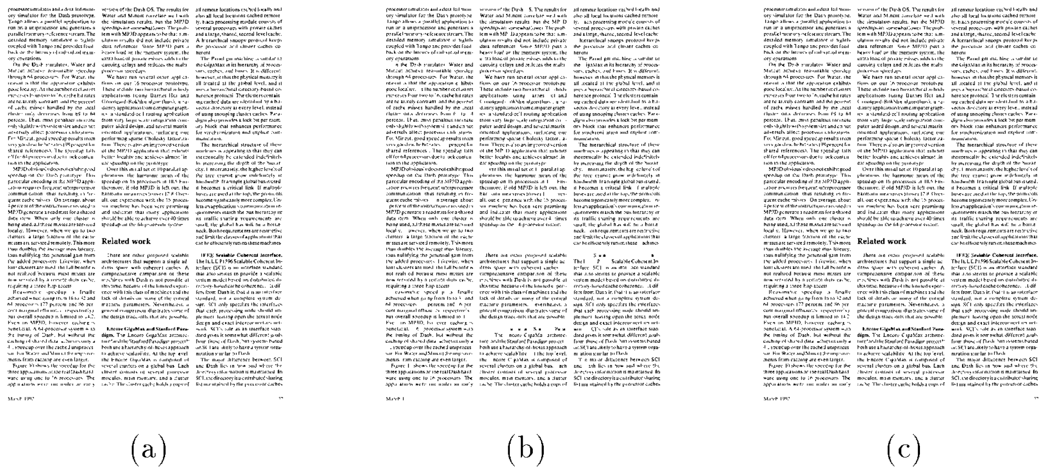


Figure 8.3: Example of a document at levels of symbolic lossiness. a) Lossless representation taking 120KB of storage space, b) purely symbolic representation taking 33 KB of storage space, and c) lossy representation taking 44 KB of storage space.

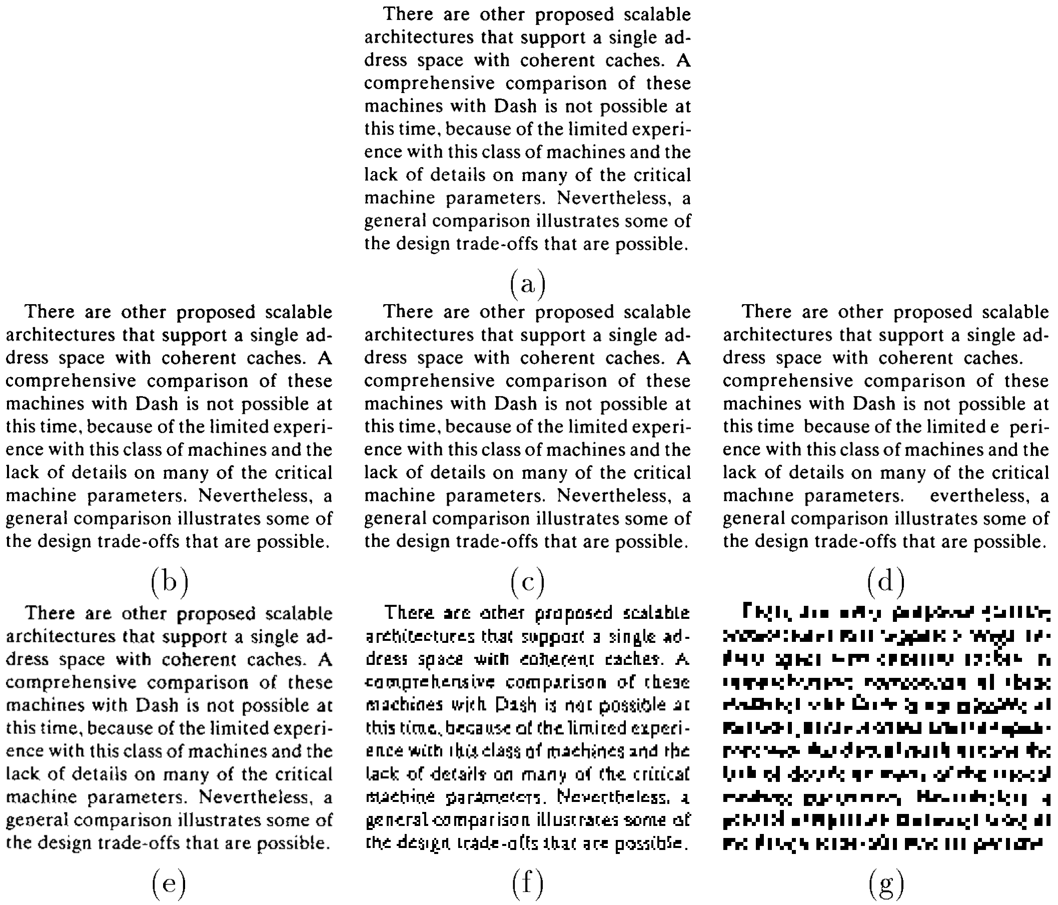
8.2 Progressive transmission

In transmitting images, an ordered stream of information should be used to convey increasing amounts of information, so that the image can be rendered losslessly if the entire stream is transmitted. In transmission of textured regions, taking a blockwise DCT (Discrete Cosine Transform) of the image and sending one coefficient per block at each iteration provides a reasonable progressive representation of the image. In the case of document images and the symbolic representation, we first transmit all the prototype and symbolic information, and we order the residual maps based on the distance transforms of the prototypes (or more precisely: of their complements). Sending the residual maps in this order, largest distance first, renders the document quite readable at the start and provides additional detail at each transmission iteration.

We have successfully implemented a progressive transmission system based on our representation. Figure 8.5 shows a progression for a test image which consists of instances of similar symbols; we transmit pixels that are farthest from the prototypes' edges first. Since larger components have residual pixels with larger absolute distances, they are reconstructed first, which is desirable since larger components are perceptually more salient than smaller components.

8.3 Sub-document retrieval

A common document browsing task requires retrieval and decoding of a subregion of the document image. This may be useful, for example, for expanding a region such as a single article or picture from a thumbnail of a compressed newspaper page. This



There are other proposed scalable architectures that support a single address space with coherent caches. A comprehensive comparison of these machines with Dash is not possible at this time, because of the limited experience with this class of machines and the lack of details on many of the critical machine parameters. Nevertheless, a general comparison illustrates some of the design trade-offs that are possible.

(a)

There are other proposed scalable architectures that support a single address space with coherent caches. A comprehensive comparison of these machines with Dash is not possible at this time, because of the limited experience with this class of machines and the lack of details on many of the critical machine parameters. Nevertheless, a general comparison illustrates some of the design trade-offs that are possible.

(b)

There are other proposed scalable architectures that support a single address space with coherent caches. A comprehensive comparison of these machines with Dash is not possible at this time, because of the limited experience with this class of machines and the lack of details on many of the critical machine parameters. Nevertheless, a general comparison illustrates some of the design trade-offs that are possible.

(c)

There are other proposed scalable architectures that support a single address space with coherent caches. comprehensive comparison of these machines with Dash is not possible at this time because of the limited e perience with this class of machines and the lack of details on many of the critical machme parameters. evertheless, a general comparison illustrates some of the design trade-offs that are possible.

(d)

There are other proposed scalable architectures that support a single address space with coherent caches. A comprehensive comparison of these machines with Dash is not possible at this time, because of the limited experience with this class of machines and the lack of details on many of the critical machine parameters. Nevertheless, a general comparison illustrates some of the design trade-offs that are possible.

(e)

There are other proposed scalable architectures that support a single address space with coherent caches. A comprehensive comparison of these machines with Dash is not possible at this time, because of the limited experience with this class of machines and the lack of details on many of the critical machine parameters. Nevertheless, a general comparison illustrates some of the design trade-offs that are possible.

(f)

[illegible]

(g)

Figure 8.4: Comparison of lossy compression between symbolic and JBIG compression at almost constant levels. a) Lossless representation of a portion; (b-d) range of lossy representations, from least to most, using symbolic compression; (e-g) range of lossy representations, from least to most, using JBIG compression.

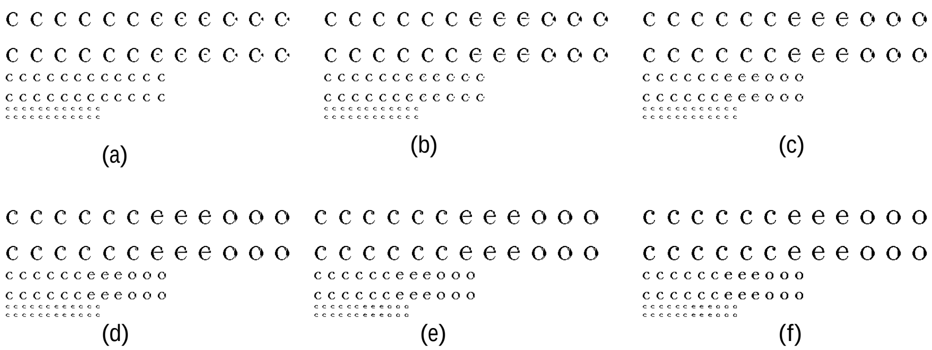
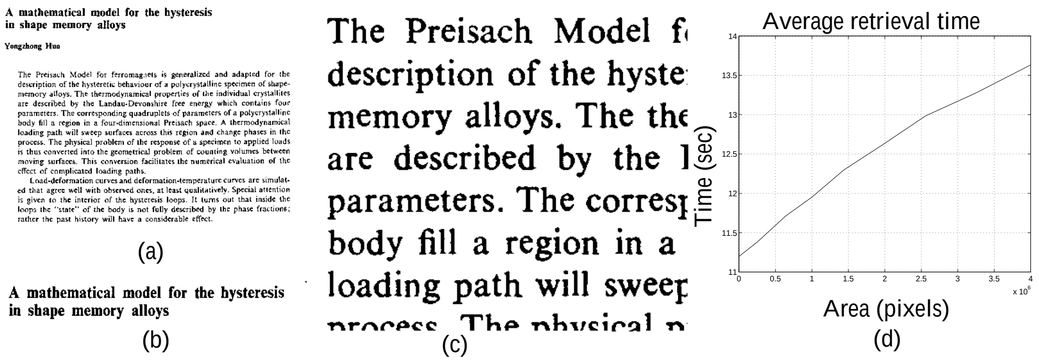

c c c c c c c e e e o o o
(a)
(b)
(c)
(d)
(e)
(f)

Figure 8.5: Progressively sending bits ((a) to (f)); (f) is the lossless rendition.



A mathematical model for the hysteresis in shape memory alloys

Yongzhong Huo

The Preisach Model for ferromagnets is generalized and adapted for the description of the hysteretic behaviour of a polycrystalline specimen of shape-memory alloys. The thermodynamical properties of the individual crystallites are described by the Landau-Devonshire free energy which contains four parameters. The corresponding quadruplets of parameters of a polycrystalline body fill a region in a four-dimensional Preisach space. A thermodynamical loading path will sweep surfaces across this region and change phases in the process. The physical problem of the response of a specimen to applied loads is thus converted into the geometrical problem of counting volumes between moving surfaces. This conversion facilitates the numerical evaluation of the effect of complicated loading paths.

Load-deformation curves and deformation-temperature curves are simulated that agree well with observed ones, at least qualitatively. Special attention is given to the interior of the hysteresis loops. It turns out that inside the loops the "state" of the body is not fully described by the phase fractions; rather the past history will have a considerable effect.

(a)

A mathematical model for the hysteresis in shape memory alloys

(b)

The Preisach Model f
description of the hyste
memory alloys. The the
are described by the l
parameters. The corresp
body fill a region in a
loading path will sweep
process. The physical p

(c)

(d)

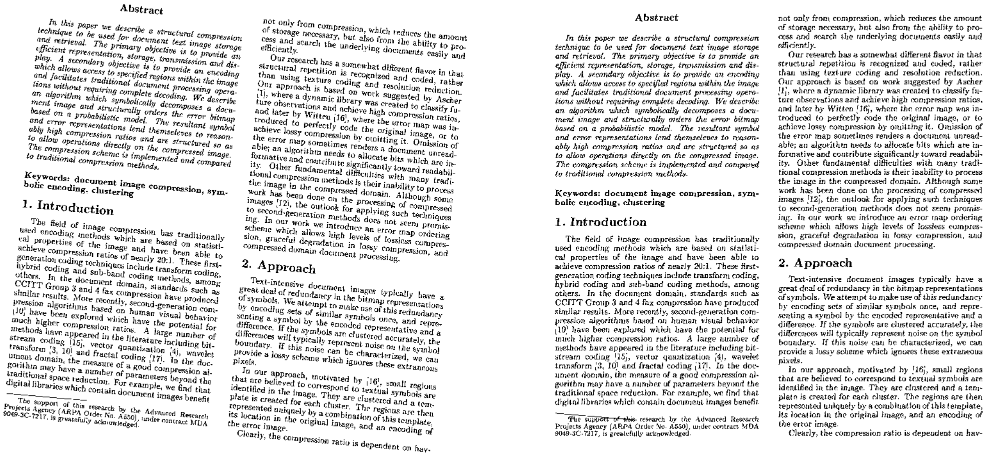
Figure 8.6: A001BIN.TIF image at a) (700,700)-(2300,2000) b) (700,700)-(1900,950) c) (830,1090)-(1280,1490) d) Retrieval time as a function of area.

can be done using our representation by partially decoding the appropriate streams. Specifically, we decompress the prototype size, prototype image, block membership, component layout index, and residual map index streams, and only partially decompress the largest streams, the component layout and residual map streams. Using the layout index stream we decompress components in blocks which overlap with the subimage, and according to the overlap of each component with the subimage we decompress its residual map. An example is shown in Figure 8.6a. If the coordinates of a subtitle are known, we may decompress only the subtitle (Figure 8.6b), and if the top left corner of the first paragraph is needed, we can decompress accordingly (Figure 8.6c). We can decompress only the top part of the first page in a document to determine the title and authors of the document. The processing time depends on the area of the region of interest. Figure 8.6d shows a plot of decompression time versus retrieval area. The time required to decompress small regions can be viewed as overhead time, and the increase in time with area defines the scalability.

8.4 Skew estimation and correction

Skew estimation and correction are also important tasks in an OCR system. By using an algorithm based on the Hough transform, we are able to estimate skew and correct it using our representation, without having to fully decompress. We use only the position and size of each component. To compute their values, we decompress the prototype size, block membership, and component layout streams. We input the coordinates of the middle of the bottom of each component to a Hough transform, and thus compute the skew. For the 122 images in our database, it took an average of 2.5 seconds and 9152 bytes per image to calculate skew to an accuracy of 1/640 vertical units per horizontal unit. On the same set of test documents the average error was measured to be 0.1786 degrees.

For skew correction, we modified the component layout and residual map streams and reordered them if the components moved from one block to another. Reordering is computationally expensive and does not contribute much correction for small skew angles. To improve performance, we bounded the movement of the components across



Abstract

In this paper we describe a structural compression technique to be used for document text image storage and retrieval. The primary objective is to provide an efficient representation, storage, transmission and display. A secondary objective is to provide an encoding which allows access to specified regions within the image and facilitates traditional document processing operations without requiring complete decoding. We describe an algorithm which symbolically decomposes a document image and structurally orders the error bitmap based on a probabilistic model. The resultant symbol and error representations lend themselves to reasonably high compression ratios and are structured so as to allow operations directly on the compressed image. The compression scheme is implemented and compared to traditional compression methods.

Keywords: document image compression, symbolic encoding, clustering

1. Introduction

The field of image compression has traditionally used encoding methods which are based on statistical properties of the image and have been able to achieve compression ratios of nearly 20:1. These first-generation coding techniques include transform coding, hybrid coding and sub-band coding methods, among others. In the document domain, standards such as CCITT Group 3 and 4 fax compression have produced similar results. More recently, second-generation compression algorithms based on human visual behavior [10] have been explored which have the potential for much higher compression ratios. A large number of methods have appeared in the literature including bit-stream coding [15], vector quantization [4], wavelet transform [3, 10] and fractal coding [17]. In the document domain, the measure of a good compression algorithm may have a number of parameters beyond the traditional space reduction. For example, we find that digital libraries which contain document images benefit not only from compression, which reduces the amount of storage necessary, but also from the ability to process and search the underlying documents easily and efficiently.

Our research has a somewhat different flavor in that structural repetition is recognized and coded, rather than using texture coding and resolution reduction. Our approach is based on work suggested by Ascher [1], where a dynamic library was created to classify future observations and achieve high compression ratios, and later by Witten [16], where the error map was introduced to perfectly code the original image, or to achieve lossy compression by omitting it. Omission of the error map sometimes renders a document unreadable; an algorithm needs to allocate bits which are informative and contribute significantly toward readability. Other fundamental difficulties with many traditional compression methods is their inability to process the image in the compressed domain. Although some work has been done on the processing of compressed images [12], the outlook for applying such techniques to second-generation methods does not seem promising. In our work we introduce an error map ordering scheme which allows high levels of lossless compression, graceful degradation in lossy compression, and compressed domain document processing.

2. Approach

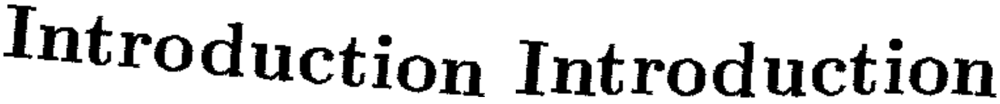
Text-intensive document images typically have a great deal of redundancy in the bitmap representations of symbols. We attempt to make use of this redundancy by encoding sets of similar symbols once, and representing a symbol by the encoded representative and a difference. If the symbols are clustered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

In our approach, motivated by [16], small regions that are believed to correspond to textual symbols are identified in the image. They are clustered and a template is created for each cluster. The regions are then represented uniquely by a combination of this template, its location in the original image, and an encoding of the error image.

Clearly, the compression ratio is dependent on hav-

~~The support of this~~ research by the Advanced Research Projects Agency (ARPA Order No. A550), under contract MDA 9049-3C-7217, is greatefully acknowledged.

Figure 8.7: Deskewing example: a) Skew at five degrees, b) deskew.



(a) (b)

Figure 8.8: Small portion of the deskewing example: a) Skew at five degrees, b) deskew.

a block boundary to five pixels; below this bound, movement is terminated at the edge of the block. For instance, if a component is located close to a block boundary, and by deskewing, it moves two pixels into another block, it is moved to the edge of its original block. We take the middle of the image as the reference point. The translation in the vertical direction is computed by multiplying the horizontal distance from the middle of the page by the slope of the line containing the reference point and component location.

Figure 8.7 shows two examples of skewed documents and the results of deskewing them. For the 122 test images, it took an average of 3.3 seconds to deskew an image. Figure 8.8 shows two small examples. Note that the local skew is not corrected.

8.5 Keyword search

Keyword search is another important document-related task. A method reported by Spitz [98] requires scan-line ordering of components and matching of ordered components to queried components. To implement this method using our representation, we decompress the prototype size, prototype image, block membership and component layout streams. The skew angle of the document is first estimated, as described above, and the bounding boxes are horizontally projected. The projection is then scanned to determine the locations of lines of text. We scan the lines from top to bottom and record the components which each line contains. This method picks up disjoint components, such as the dots in the 'i' and 'j', very nicely. It is also very stable to errors in locations of objects and in scan direction.

Once the components are ordered we apply a feature-based matching algorithm. We classify each component as being an x-height, an ascender, a descender, an "i", a "j", or punctuation. A second-level classification checks if there exists a hole in the component, and a third checks for the existence of a concavity from the right. These features are computed only for the prototypes rather than for the individual components. The heights of the components are measured only for the first ten instances of each prototype in the image, and their values are averaged. These features are shown in Figure 8.9. The input query is mapped into this feature space using a simple lookup table; 90% of the maximum score is taken to be a match. Figure 8.10 shows search results on the 122 database images for the query "approach". It took an average of 2.7 seconds per image to obtain the results. There are fundamental ambiguities associated with using our small set of features; for example, we are treating characters like "a" and "o" as belonging to the same feature class. However, the effects of these ambiguities are greatly reduced when a string of features is used in matching.

8.6 Duplicate detection

Consider a situation where thousands of documents are being imaged and added to a database, possibly from a distributed environment. If multiple instances of the same document exist, they may be re-entered into the database unnecessarily. This may not be desirable for a number of reasons, including increased storage cost, difficulties in maintaining database integrity, increased processing cost for database operations, and cost of indexing multiple images with the same underlying content. The definition of a "duplicate instance" is open to some interpretation and we only consider duplicates where multiple instances of an effectively identical original source are scanned for incorporation into a database. The original documents may have been written on, stapled, torn, taped or may have pages missing or had a cover added. The document may have been copied repeatedly, so different-generation copies are involved. The document may have been scanned at different times and on different devices, so resolution, illumination, and contrast are also issues.

One approach to the problem of duplicate detection is to address it from the symbolic point of view. In a database that has organized the image information based on the symbolic representation, it is possible to analyze the prototype image maps and their distribution throughout the image. This distribution can be compared to the distribution of another image by associating similar prototypes. Figure 8.11 shows two prototype image maps belonging to different document images. The similarities and dissimilarities of the prototypes are obvious. To create a correspondence between them, a base set of prototypes is created which is common to the two images. This base set also takes into account the similarities between prototypes in the same image.

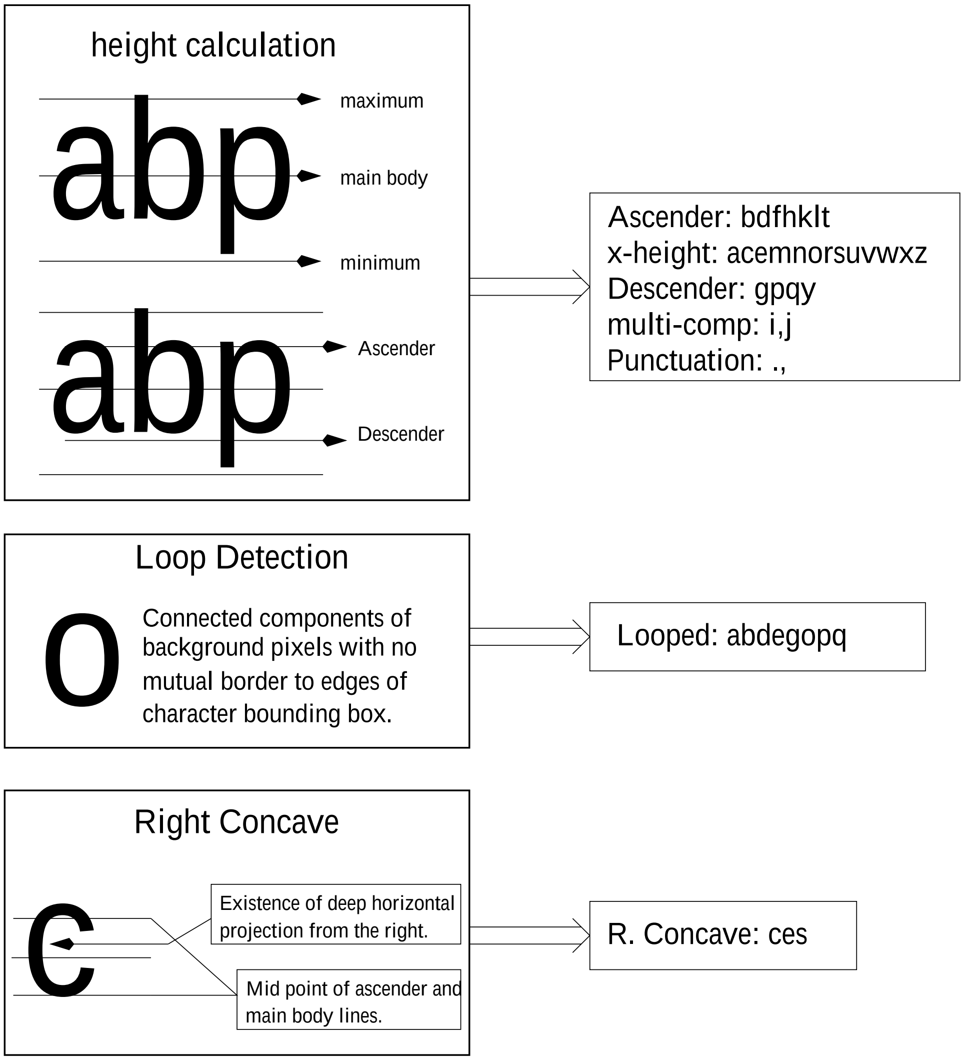


Figure 8.9: Shape code description for use in keyword searching.

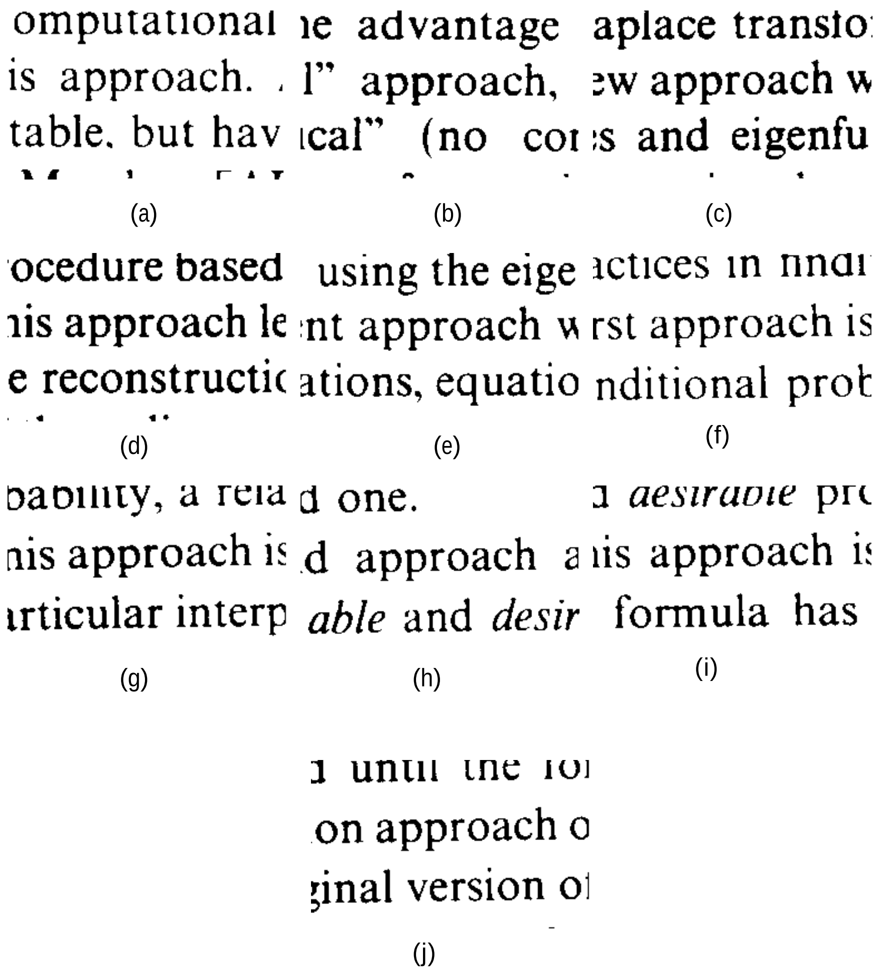
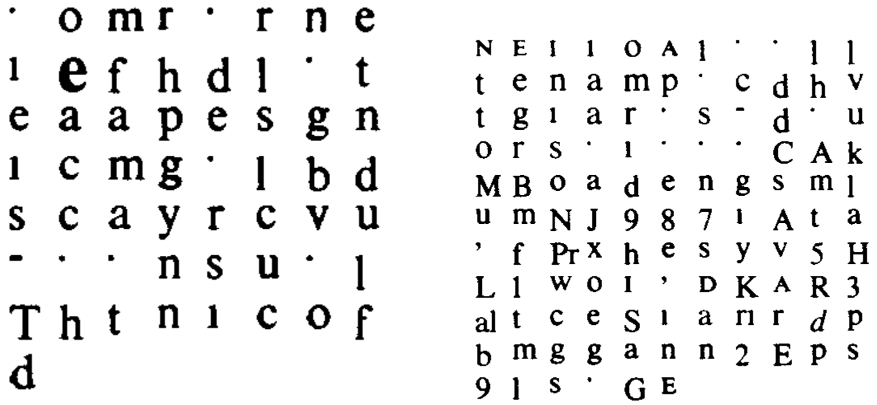

omputational
is approach.
table. but hav
(a)
ie advantage
l" approach,
ical" (no coı
(b)
aplace transfo
ew approach w
s and eigenfu
(c)
ocedure based
iis approach le
e reconstructic
(d)
using the eige
nt approach w
ations, equatio
(e)
actices in final
rst approach is
nditional prot
(f)
bability, a rela
nis approach is
irticular interp
(g)
d one.
d approach a
able and desir
(h)
a desirable pro
iis approach is
formula has
(i)
d until the fol
on approach o
ginal version of
(j)

Figure 8.10: Match results for string "approach": (a) from A03L, (b) from A03P, (c)-(d) from A053, (e) from A054, (f)-(j) from A06I.



(a) (b)

Figure 8.11: Examples of two prototype sets.

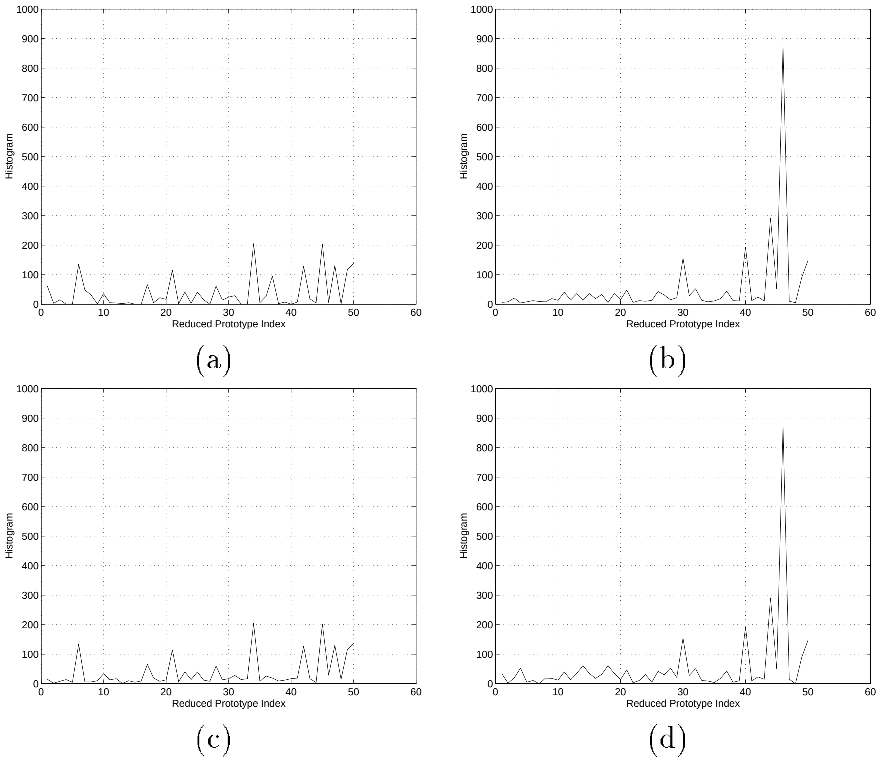
The resulting base set (a smaller set than the prototype sets of both images) is then used to determine the distributions of the components. The distributions of these base prototypes are then compared for a possible match.

Figure 8.12 shows the distributions in the images of the prototypes that are shown in Figures 8.11(a) (first image) and (b) (second image). Figure 8.12a shows the distribution with respect to a prototype base set derived primarily from the first image and Figure 8.12b is the distribution of the second image with respect to the same prototype set. Figures 8.12c and d are the distributions with respect to a prototype base set derived from the second image. There exists some level of discrepancy between the distributions which might be used for discrimination. It only remains to determine the level of similarity after some degradation corresponding to realistic duplication.

Figure 8.13 shows distributions for an image before and after various amounts of degradation. There exists a large amount of similarity between the distributions. However, a test of distribution similarity is very time consuming, rendering this method unrealistic for determining duplicates in databases containing millions of images. This test does, however, provide a good tool for performing comparisons at a second or third level after greatly reducing the pool of suspect documents.

8.7 Summary

Computer systems are being increasingly used to process and analyze images of many types. These tasks vary in computational complexity and in storage and communication requirements. Advances in data compression have been able to reduce these requirements; a compressed file takes less storage space, takes less time to read and process, and takes less channel capacity to transmit. Compression is especially important in handling document images because of the large file sizes and the intensive



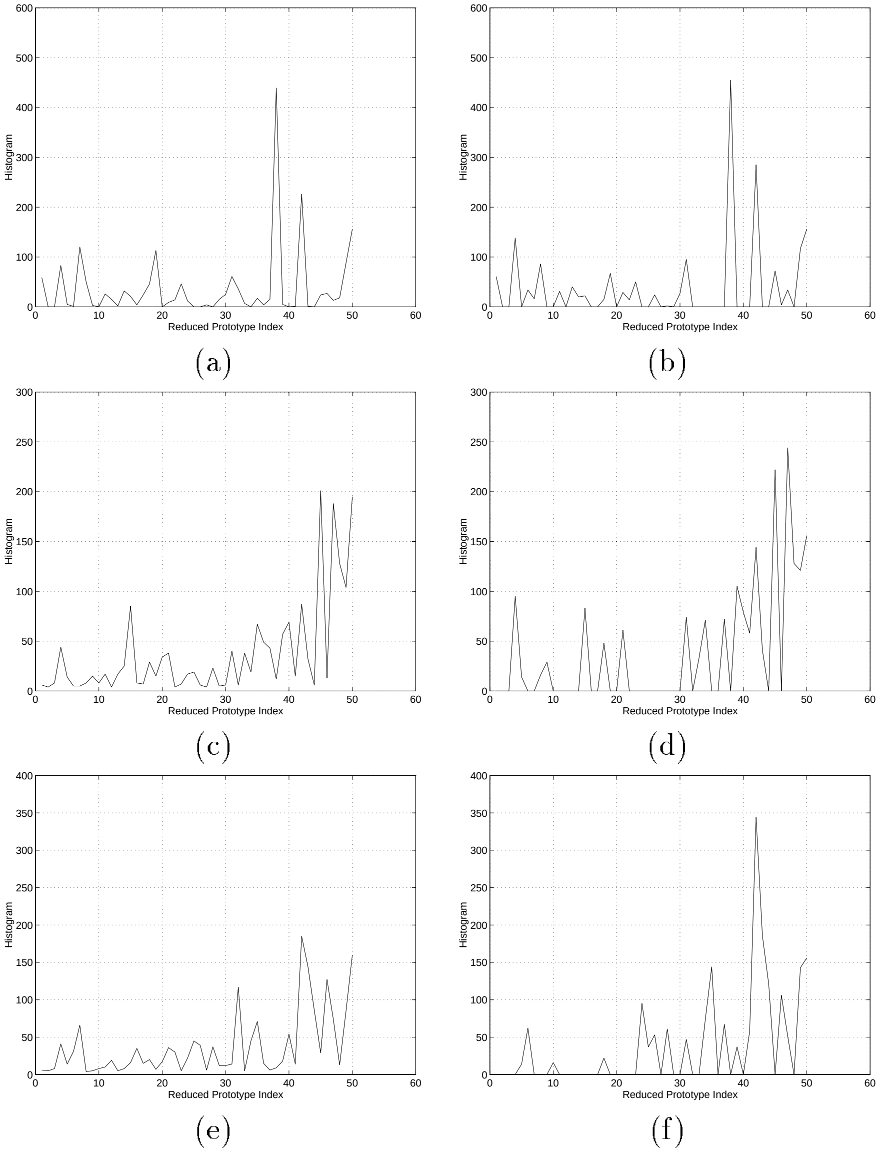
(a)

(b)

(c)

(d)

Figure 8.12: Histograms of components assigned to base prototype image sets. (a,b) and (c,d) have common prototype bases but the histograms are obtained from different images.



(a)

(b)

(c)

(d)

(e)

(f)

Figure 8.13: Histograms of components assigned to base prototype image sets for an image that has undergone various levels of degradation. The (a,b) pair has the lowest amount of degradation and the (e,f) pair has the highest amount of degradation. (a,c,e) are the original image histograms and (b,d,f) are the degraded image histograms.

processing they require. With the increasing popularity of digital libraries, retaining and presenting scanned documents in image form remains a cost-effective and usually accurate way to distribute document information that was originally in hard copy.

We have developed a system that addresses most aspects of document image compression, especially those related to transmission and processing. Our system achieves compression while still allowing for efficient processing. It is evident that the important information in documents is at the symbol level which implies that traditional resolution-reduction methods may alter symbol shapes and reduce performance in document processing tasks. By symbol coding, derived in part from pattern-matching and substitution algorithms, we have shown that image coding and retrieval can be done efficiently. Image analysis tasks such as skew estimation, correction, keyword searching, and duplicate detection can also be performed efficiently since the symbolic representation preserves sufficient information to be used in these tasks.

In this chapter we have addressed the problem of compression strictly in the image domain, and for good reason. It is clear that the ultimate form of symbolic compression is simply to recognize the symbols and represent them by their ASCII codes. However, such a compressed representation suffers by not preserving information about the fonts, point sizes, locations on the page, etc. of the symbols. To varying degrees such features can be preserved, but the representation does not allow us to reconstruct a truly lossless version of the original. A hybrid compression scheme which integrates ASCII symbols which can be recognized with high accuracy into a scheme such as ours would be ideal.

Chapter 9

Applications and extensions

A number of applications and enhancements can be built into our current compression system or can benefit from its structure. In our base system we were able to create an infrastructure that can be used for compression, transmission, and processing.

An application that was considered earlier for compressed-domain processing was document duplicate detection. While it was not able to perform fast enough to handle millions of documents, it can be modified to handle such computational demands. One source of overhead is the number of prototype images. If we can map the prototypes into a fixed number for all the documents, this will reduce computation dramatically. Another source of computational reduction is in the number of components. We can consider limiting the number of components, at the cost of increasing the false detection rate. However, preserving the order of the components as they appear on a scan line in the image can help to dramatically reduce the false detection rate.

A possible extension of our base compression method is to grayscale document images. Since documents may contain images, it may be unrealistic to scan than as binary images. For multi-level documents, many essential document image characteristics can be preserved in grayscale. For such images, resolution reduction is still not desirable, but grayscale re-quantization is an option for lossy representation.

Graphic images can also benefit from symbolic compression. A graphic image usually contains lines, arrows, legends, and a small amount of textual information. There are very few of these constituent patterns and there is hope of achieving high levels of compression.

Networked databases could also benefit from the hierarchical data representation of our base compression scheme and from enhancements to the residual coding. Some problems which could be addressed are source-channel coding, design of variable error resiliency codes, client-server task optimization, packet, TCP/IP, and ATM transmission optimization.

Finally, we have experimented briefly with extensions to different media types, including textured images, video, and sound. These media domains already have potential for developing a hierarchical model. The extension considered here is the intelligent merging of media types so that problems of compression, transmission, and processing are addressed across media types. A hyper-document [92] serves the purpose of integrating multiple media types, so that compression, transmission, and processing of hyperdocuments can be considered.

In this chapter, we present preliminary results in each of these areas. In Section 9.1 we provide an alternate approach to duplicate document detection, in addition to that suggested in Chapter 8, which can process millions of documents on user demand. In Section 9.2 we present some preliminary results regarding the compression of grayscale document images. In Section 9.3 we give an argument for using the same compression methodology for graphic images. In Section 9.4 we propose a way to address the communication of document images in a networked environment along with related problems and advantages. In Section 9.5 we extend the classical document function to include other media types and fit these media sources within the framework of our symbolic compression. We conclude in Section 9.6 with some final remarks.

9.1 Duplicate document detection

Depending on what information is available a priori in a database system, the problem of duplicate detection can be approached in a number of ways. If, for example, basic index information such as the document number, date, title, authors or number of pages is entered manually prior to scanning, this information can serve as a preliminary filter for duplicates. In most cases, however, high-volume operations prohibit such manual entry prior to scanning. Instead, we would like to identify duplicates from their images prior to any manual entry. Although we have basic quantitative information such as the number of pages, at this point we consider only the analysis of the image itself.

One possible solution which has been proposed is to apply OCR to the document image and match as much text as possible between the documents. Although this matching can be done relatively quickly, OCR performance suffers on degraded documents in terms of both accuracy and speed. For this reason, we do not feel that OCR is feasible as a first-level filter, but it may be used as a secondary filter to reduce possible matches from hundreds to tens of documents.

A more realistic solution (in terms of resources and time requirements) would require an approach that does not process every image every time duplicate detection is initiated. This is logical in the sense that for each detection procedure all images in the corpus need to be processed redundantly. If an algorithm can produce features that need only be calculated once, duplication detection could benefit from this. The detection process would not be directly dependent on the presented image but only on the set of features derived from it. To consider this approach we need to determine a signature that describes the presented image accurately and is consistent across its duplicate instances, but that also provides sufficient separability to distinguish it from non-duplicates. Then we can address concerns about determining this signature by performing compressed-domain processing of the image in symbolic (or other) form. (Hull [40] operated on CCITT G4 images to detect duplicates.) A large number of features can be used to define the signature to be used for duplicate detection. We are constrained by the fact that we may be dealing with millions of documents, many of which may be highly degraded. To cope with such situations, we must have a *signature* for each document which is

Robust - The signature should be reliably extractable, even when the document becomes degraded.

Unique - Although we cannot realistically expect the signatures to be unique unless we use an excessively large feature set, a given signature should be associated with several tens of documents at most.

Compact - The storage capacity required to hold the signatures of millions of objects may be very large, so the index keys should be as small as possible.

In addition, the algorithms which extract the signature must be

Fast - Algorithms which take minutes to extract a signature, and then attempt to match it against each document in the database, are not acceptable. The application demands rapid extraction, and near constant time indexing of previously entered documents

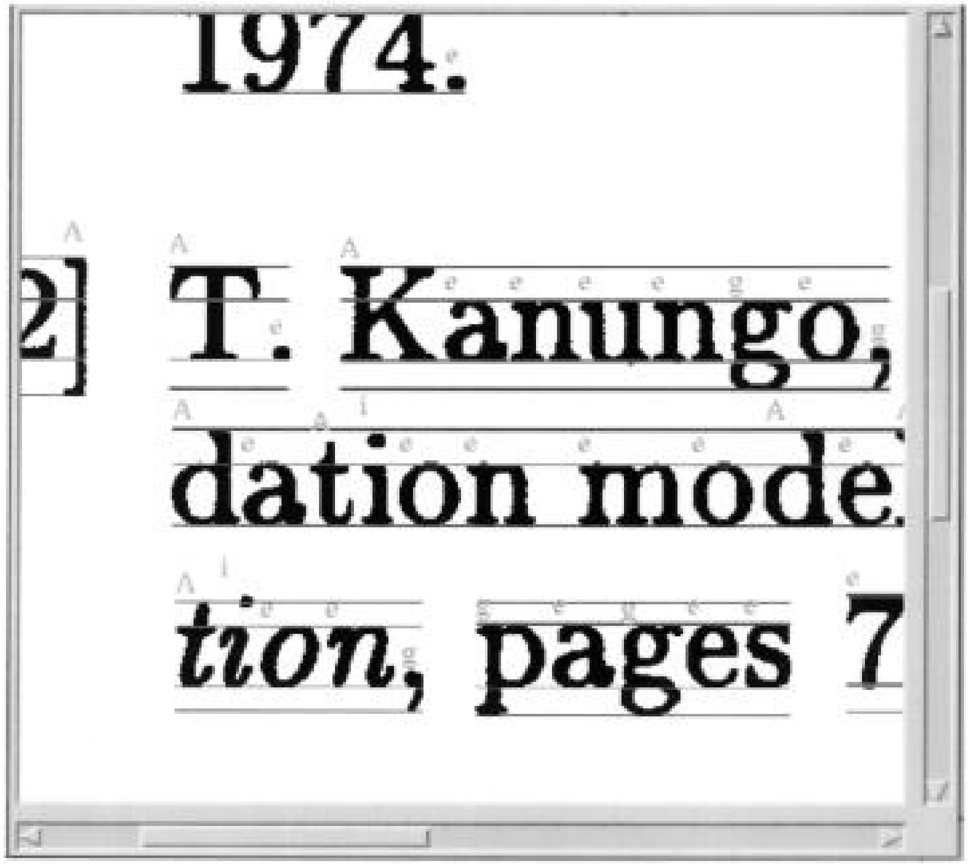
Scalable - Initially the algorithms will work on hundreds of document, but as more documents are processed, the size of the database could grow to tens of millions.

Accurate - It is acceptable to miss a small percentage of duplicates since the result is only that the same document is entered twice, but identifying documents as duplicates when they are not (false alarms) is not acceptable.

9.1.1 Related work

The detection of duplicate or near-duplicate documents has been a problem of interest for some time in many fields, but has not been addressed for collections of images. Some example domains include education, for detection of plagiarism [78]; publishing, for detection of unauthorized copies [89, 96]; databases, for maintaining database integrity; information retrieval, for information filtering [115]; and in the USENIX community, for detecting duplicate files [67].

In the document community, most of the work on identifying similar documents has been done using either ASCII documents or "water-marked" electronic representations. Much less work has been done with document images. A notable exception is Hull [39], who describes a method for matching documents which have the same character content but which may have been reformatted or distorted prior to re-imaging (content-variant documents). Hull's approach represents each document by a set of robust local features which can be used to hash into a database of descriptors. The features in both the query example and the database must be invariant to geometric distortions; by extracting multiple descriptors from each document, they can also be made robust to errors in feature extraction. The measure of similarity is simply the number of features the query document and the database instance have in common. Experiments were performed using the symbolic approach where distributions of components assigned to a common set of prototypes were examined. However, this process relied heavily on the assumption that the images should be represented in symbolic format; this may not provide an efficient approach. Another experiment was performed using as features the character counts for each word in short sequences of words; this provided a set of simple yet robust features that was adequate for small databases. With as few as ten features, 100% accuracy was obtained for a clean query string and a small clean database. Unfortunately, as the number of documents grows, the character count metric becomes less discriminatory.



1974.

2] T. Kanungo,

dation mode

tion, pages 7

Figure 9.1: Sample character shape code assignment.

9.1.2 Basic approach

Another approach is based on the extraction of a signature from a representative line of text in the document image using a shape coding technique. The technique has been used by a number of authors including Tanaka [103] and Spitz [99] for other document analysis applications. Shape coding labels the symbols in the line of text based on very simple shape properties, such as whether they are ascenders, descenders, limited to the x-line, multi-component, or punctuation, for example. These properties are much more robust to noise than the features necessary for OCR, and can be extracted fairly rapidly.

To extract the signature, the document is scanned for a representative sample of text, typically on the order of 50 symbols, on a single line or across several lines. For this sample, the base-line, x-line, ascender-line and descender-line are identified, and each character component is assigned a shape code as shown in Figure 9.1.

The string of shape codes assigned to the characters of the text sample is used as a signature for the document. A second level of robustness is added by indexing based on n-grams of the codes, rather than attempting to use an index based on the entire string. Each shape code n-gram serves as an index *key* into the database. A single dropped or inserted code will affect a small number n of these keys but will not affect

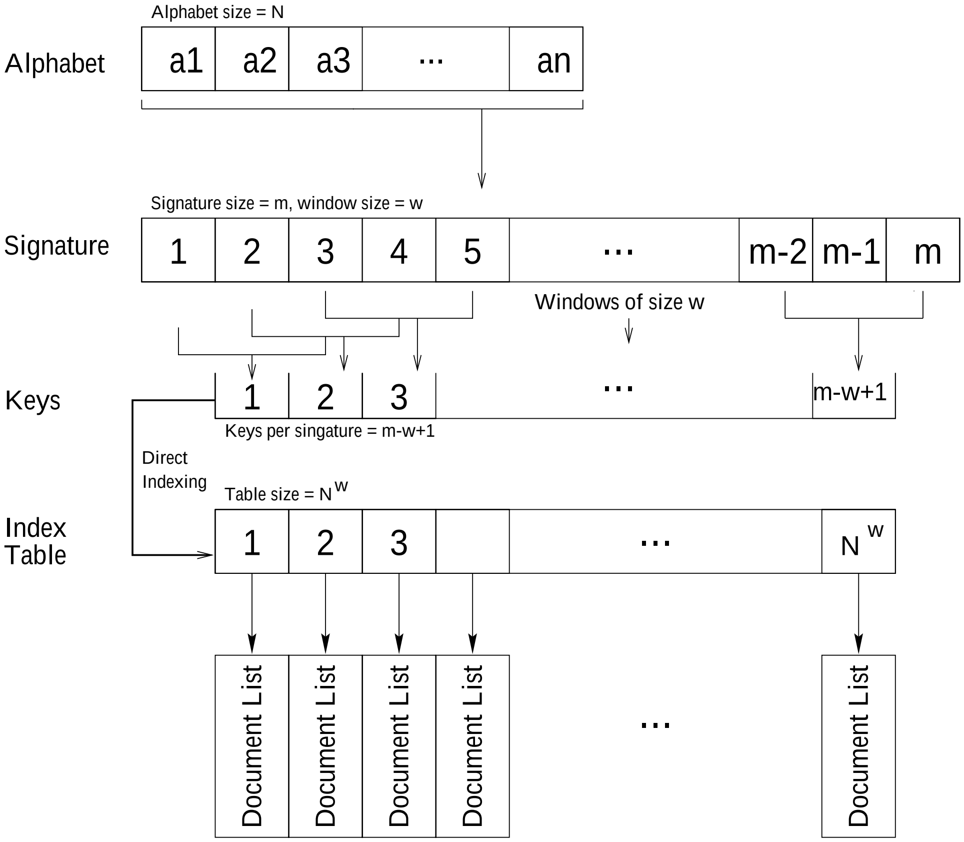


Figure 9.2: Overview of indexing scheme.

the entire signature. Figure 9.2 shows the relationship between the signature, its keys, and the database. When a set of keys is presented for indexing, each key results in a set of hits from the database. Each hit is counted as a vote for the resulting document, and a ranked set can be returned.

Clearly, a number of additional issues should be addressed in developing a complete system that satisfies the criteria set forth above. These include:

- use of global classifiers - number of pages, page component statistics, etc. as first-level filters to reduce the duplicate search space.
- choice of a shape code alphabet - selection of features to incorporate into the signature which provide maximum discrimination.
- extraction of features - how to select the signature in the image of a document.
- database organization and indexing - how to create efficient ways to index into large collections.
- verification of candidate duplicates.

All of these issues are addressed while designing a working system.

9.1.3 Feasibility analysis

Before implementing and testing this approach on real data, we performed a theoretical analysis to see if our approach is realistic and if it is robust to errors in signature extraction. The parameters that were varied in our analysis included system-dependent variables such as file size limitations of the operating system, disk access time and disk transfer rate; database variables such as the number of documents, the size of the index table and the average size of the documents; and algorithm variables such as the size of the signature alphabet, the size of the signature and the key or n-gram size.

The analysis yielded qualitative estimates of the expected size of the database, the computational requirements for matching signatures, and the number of missed and false duplicate detections as functions of the database size. It was found that the system could be implemented with generally available hardware.

The analysis was performed assuming we have extracted a candidate signature. The following is a list of variables which can be used in the feasibility and performance analysis of the algorithm. Here the signature is the vector of feature values used to represent the document and a key is a term, possibly resulting from a partitioning of the document signature, used to index into the database. A majority of the parameters reflect physical constraints of the system and are necessary to explore scalability.

Algorithm-Dependent Variables

N : Number of documents.
N_f : Maximum number of files for the index table.
S_f : Maximum file size for storing b buckets.
E : Size of index table entries in bytes.
d : Average size of documents in bytes.
l : Number of buckets that will be kept in the main memory.
b : Number of index table buckets that will be stored in a file in main memory.

Independent Variables

a : Size of alphabet used to construct the signature.
m : Size of the signature.
w : Window size.
k : Number of keys for a given signature size $(m - w + 1)$.

The matching algorithm relies on the index structure that is generated as documents are added to the database. Each signature is partitioned into equal-sized overlapping windows of size w, to be used as index keys. This partitioning provides robustness in the sense that errors in the signature will only be propagated within the window, at the expense of a less unique set of sub-keys. For a signature of size m we have $k = (m - w + 1)$ possible keys (Figure 9.2) to be indexed. The index table has

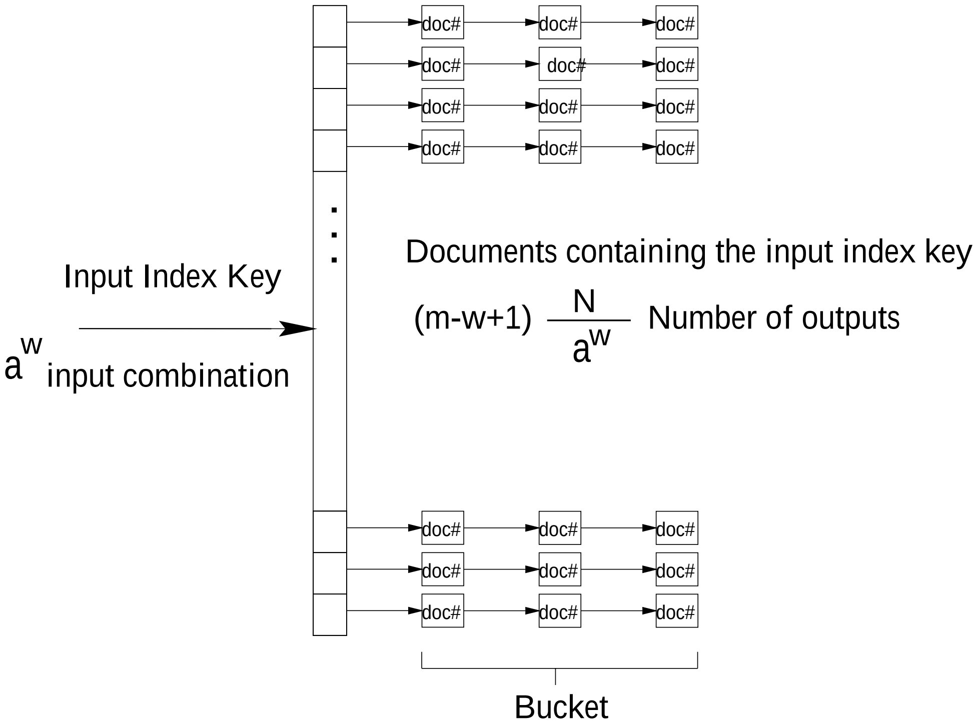

doc#
Documents containing the input index key
Input Index Key
$(m-w+1)\frac{N}{a^w}$ Number of outputs
a^w input combination
Bucket

Figure 9.3: The index table.

on the average $k \times (\frac{N}{a^w})$ entries per bucket with each entry being the identification of a document which contains that key. Figure 9.3 shows the index table structure. A table with a maximum size of $\text{Min}\{N, (k^2 \times (\frac{N}{a^w}))\}$ keeps a count of the documents that have matched the input key. When the document's keys are indexed, a counter is incremented for each document which contains that key. The most frequently occurring documents will then be the candidate duplicates. A more in-depth discussion is contained in [23].

9.1.4 Performance analysis

To characterize the performance of a specified system we first need to formulate several probabilistic models and define a design criterion. This is done by using hypothesis testing and deriving an appropriate distribution function such that specific performance measures can be computed. For a hypothesis test consider two events, called the null and alternative events. The null event, H_0, signifies a situation where there is no duplicate for a given document, and the alternative event, H_1, signifies that there is a duplicate.

$$\begin{array}{lcl} H_0: & \text{Null Hypothesis} & P(\eta|H_0) \\ H_1: & \text{Alternative Hypothesis} & P(\eta|H_1) \end{array} \tag{9.1}$$

Associated with each hypothesis is a probability measure which characterizes the variable used for detection, in this case η. In detection, a thresholding operation is used to determine which hypothesis is valid. In this case η signifies a matching score assigned to an observation. By choosing an appropriate threshold η_T we make a decision that if $\eta \geq \eta_T$ then accept the alternative hypothesis and if $\eta < \eta_T$ then accept the null hypothesis. The probabilities of detection and of false alarm effectively capture the performance of this detector:

$$\begin{aligned} P_D(\eta_T) &= P(\eta \geq \eta_T | H_1) \\ P_{FA}(\eta_T) &= P(\eta \geq \eta_T | H_0) \end{aligned} \tag{9.2}$$

This set of numbers will be used as operating specifications given various parameters, but to fully specify P_D and P_{FA} we need to analyze or make assumptions about the available data and the matching processes.

Given a signature of size m from an alphabet of size a and an observation window of size w, define x_i to be the ith symbol and y_j to be the jth index key. With $x_i \ \in X_i = \{0,1\}^{\log_2(a)}$ it is easy to see that all the x_i's are independent. In fact

$$P(X = x) = \prod_{i=1}^{m} P(X_i = x_i) \qquad \text{for } x \in X = \bigcup_{i=1}^{m} X_i = \{0,1\}^{m \log_2(a)} \tag{9.3}$$

Let us define $y_i = \{x_i, x_{i+1}, ..., x_{i+w-1}\}$; this is clearly dependent on y_{i+1} to y_{i+w-1}. As described in the previous sections, the y_i's will be used for indexing into a signature database and the number of hits determines η. In indexing, however, there is no preference as to the order of the observed y_i values, and in a hypothetical situation two documents could have large numbers of hits by matching y_i's out of order. Smaller w values would increase this effect and larger values would decrease it; however, in most cases we can argue that the y_i values used to calculate η are independent. We shall assume that it is equally probable (with probability $\frac{1}{a^w}$) to observe any y_i and shall now present our probability analysis.

Under the conditions stated above, given a signature we observe a set of features y_i for $i = 1, 2, ..., k$. Since each y_i is now treated as an independent observation and we have k observations, we can identify the probability of η matches out of k possible matches to be binomial, $b(k, \frac{1}{a^w})$:

$$P(\eta) = \begin{pmatrix} k \\ \eta \end{pmatrix} \left(\frac{1}{a^w}\right)^{\eta} \left(1 - \frac{1}{a^w}\right)^{k-\eta} \tag{9.4}$$

Here $P(\eta)$ is the probability of the number of matches given the null hypothesis, $P(\eta|H_0)$. This probability in effect measures the relationship between different signatures and is relatively easy to understand and compute. The assumption that the occurrences of keys are equally likely in the realm of all allowable keys is unrealistic. For example, it is unrealistic for a window observation to consist of all ascenders. We therefore intend to derive realistic probabilities of key occurrences from experiments; this may revise the probability distribution function of the null hypothesis. The probability measure for the alternative hypothesis, however, is more involved.

To derive the probability distribution of the alternative hypothesis, we need to consider what realistic errors may be found between a candidate signature and its corresponding entry in the database. The discrepancies between observed and recorded signatures are in the form of insertions, deletions, or changes of shape code. In general it is more probable to have fewer errors than to have more, so we assume that the probability of observing i errors is a decaying exponential function, defined by a decay rate β . The general form of this function is $P(i) = P(0)e^{-\beta i}$, where $P(0)$ should be calculated by letting $\sum_{i=0}^{m} P(i) = 1$. Calculating $P(0)$ and formulating the distribution function we get

$$P(i) = P(0)e^{\beta i} = \left(\frac{1 - e^{-\beta}}{1 - e^{-\beta(m+1)}}\right) e^{-\beta i} \quad \text{for } i = 0, 1, ..., m. \tag{9.5}$$

This equation is not exact since this probability measure depends on the accuracy of identifying shape codes, the statistics of shape codes, and the similarity of duplicates found in the database. It is, however, impossible to characterize the statistical nature of these phenomena, so we have simplified $P(i)$ to an exponential function. In order to formulate the probability distribution of the alternate hypothesis we need to study the relationship between the number of errors and the matching score. In the case where only one error occurs, the error is propagated to w out of k index entries. This results in a matching score of $k - w$ out of a maximum of k. Although the errors might occur at either end of the signature, and insertions and deletions change the matching score (by 1) in contrast with shape changes, these occurrences are statistically insignificant and offset each other. For multiple errors, the worst-case scenario is the case where the errors occur w shapes away from each other and these errors propagate to the greatest possible number of allowable window observations. For this worst-case scenario, i errors will translate to a maximum matching score of

$$\eta_{worst} = \begin{cases} k - iw & \text{for } i = 0, 1, ..., \lfloor \frac{k}{w} \rfloor \\ 0 & \text{otherwise} \end{cases} \tag{9.6}$$

For the best-case scenario, the errors could occur next to each other, yielding a maximum matching score of

$$\eta_{best} = k - \left\lceil \frac{i}{w} \right\rceil w \approx k - i \text{ for } i = 0, 1, ..., k. \tag{9.7}$$

The probability that worst-case or best-case errors occur is clearly dependent on the number of errors. This is true since $i = 1$, by definition, is a worst-case scenario and $i = m$ it is a best-case scenario. It is also true that the score probabilities change as a power function with respect to the number of errors, and their analysis is extremely complex. Therefore, for the sake of simplification, and as will be seen in the following section we will assume that the matching score is an average of the worst- and best-case scenarios:

$$\eta(i) = \frac{\eta_{worst} + \eta_{best}}{2} = \begin{cases} k - \frac{i + i \times w}{2} & \text{for } i = 0, 1, ..., \left\lfloor \frac{2k}{w+1} \right\rfloor \\ \frac{k-i}{2} & \text{for } i = \left\lfloor \frac{2k}{w+1} \right\rfloor + 1, ..., m - w + i \end{cases} \tag{9.8}$$

Using (9.5) and (9.8), the probability distribution function of the alternative hypothesis is fully specified. We are now ready to calculate the probabilities of detection and false alarm.

Given a threshold η_T, we make a decision that any document with a higher score is identified as a duplicate, as shown in (9.2). We now have to apply this criterion to the probability distribution functions of the null and alternative hypothesis to get the probability of false alarm and probability of detection, respectively. The distribution function of the null hypothesis is given in (9.4), and summation over $\eta = \eta_T, ..., k$ gives the probability of false alarm:

$$P_{FA} = P(\eta \geq \eta_T | H_0) = \sum_{\eta=\eta_T}^{k} \binom{k}{\eta} \left(\frac{1}{a^w}\right)^{\eta} \left(1 - \frac{1}{a^w}\right)^{k-\eta} \tag{9.9}$$

In order to calculate the probability of detection we need to consider the inverse of (9.8) to be used in (9.5). Using η_T and solving for i in (9.8) gives the maximum number of allowable errors such that the effective score is the threshold score:

$$i_{max} = \begin{cases} \frac{2(k-\eta_T)}{1+w} & \text{for } \eta_T > \frac{k(w-1)}{2w} \\ k - \eta_T & \text{otherwise} \end{cases} \tag{9.10}$$

Then summing (9.5) from zero to i_{max} gives the probability of detection:

$$P_D = P(\eta \geq \eta_T | H_1) = \sum_{i=0}^{i_{max}} \left(\frac{1 - e^{-\beta}}{1 - e^{-\beta(m+1)}}\right) e^{-\beta i} \tag{9.11}$$

Given equations (9.9) and (9.11), it is possible to plot the probability of false alarm with respect to the probability of detection as a function of the threshold η_T. Traditionally, this plot is called the ROC (Receiver Operating Characteristic) curve; a typical curve is shown in Figure 9.4. Figure 9.4 also shows a diagonal line which signifies the minimum achievable performance. The diagonal line represents picking the null or the alternative hypothesis based on a coin toss; if an algorithm performs below this line, a coin toss would achieve better performance.

9.1.5 System design

Our model basically has two main profiles: performance and system resources. The performance can be characterized by specifying the error tolerance. The system resources can be thought of as constraints on the desired performance. It is hard to find a single formula that can give us optimum values of the model parameters (m, w, a) for given system resources and performance range. But we can at least give a recipe for finding "good" values for the parameters. Once the system resources are characterized, making tables (like Tables 9.1 and 9.2) for different combinations of the model parameters (m, w, a), and considering constraints for different values of these parameters, helps us understand the possible ranges of the model parameters. Then, by drawing ROC curves and looking for a desired detection probability range, we can shrink the ranges of the parameters further.

Let us comment on each parameter's effect. The increase in the value of m (the size of the signature) increases the search time and the size of the index table. However, the probability of duplicate detection becomes higher (Figure 9.5).

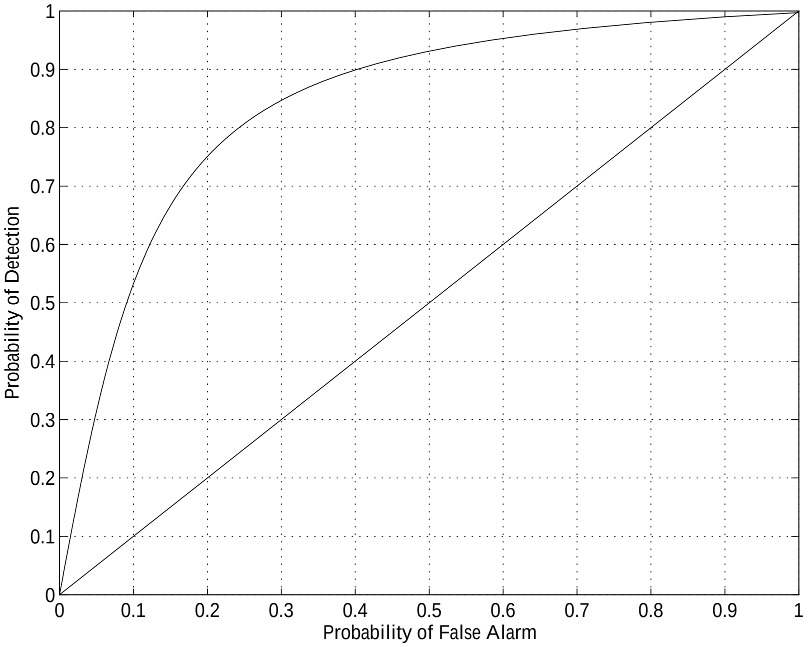


Figure 9.4: Typical ROC curve.

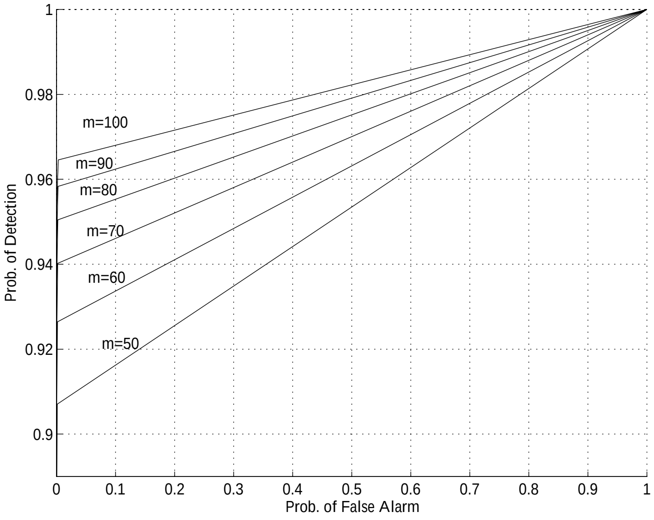


Figure 9.5: ROC curves for $\beta = 0.01$ $w = 5$, $a = 8$ and various values of m (50,...,100).

a	Number of buckets	Entries per bucket	Bucket size (MB)	Buckets per file	Bucket file size (MB)	Number of files	Expected search time (sec)
4	1024	2246094	8.6	1	8.6	1024	38.68
5	3125	736000	2.8	1	2.8	3125	12.82
6	7776	295782	1.1	1	1.1	7776	5.23
7	16807	136848	0.5	3	1.6	5603	2.64
8	32768	70191	0.3	7	1.9	4682	1.56

Table 9.1: Index table characteristics for $w = 5$, $m = 50$, $N = 50$M, $K = 2$MB.

w	Number of buckets	Entries per bucket	Bucket size (MB)	Buckets per file	Bucket file size (MB)	Number of files	Expected search time (sec)
3	512	4492188	17.136	1	17.1	512	76.94
4	4096	561524	2.142	1	2.1	4096	9.98
5	32768	70191	0.268	7	1.9	4682	1.56
6	262144	8774	0.033	59	2.0	4444	0.56
7	2097152	1097	0.004	477	2.0	4397	0.43
8	16777216	138	0.001	3799	2.0	4417	0.42

Table 9.2: Index table characteristics for $a = 8$, $m = 50$, $N = 50$M, $K = 2$MB.

The increase in the value of w (the window size) makes the detection probability lower (Figure 9.6). We conclude from Figures 9.6 and 9.5 that only $k = m - w + 1$ (the number of keys) matters for the detection probability, because increasing w means decreasing k, which lowers the detection probability.

The value of a (the alphabet size) actually has no significant effect on the detection probability (Figure 9.7). But increasing it improves the detection probability, since the false alarm rate drops.

The choice of a and w is crucial for the number of entries per bucket. The number of entries per bucket affects the search time, and as the search time increases, the number of entries increases. As w or a increases, the number of entries per bucket decreases, but on the other hand the errors increase. Smaller values of w are desirable for robustness.

9.1.6 Experiments

Two experiments were performed by Doermann et al. [23] with the results given below. The first experiment used a small database of 5000 text lines extracted from the Wall Street Journal's ASCII representation of news and reports. Each line was treated as an independent document so the database contained 5000 document im-

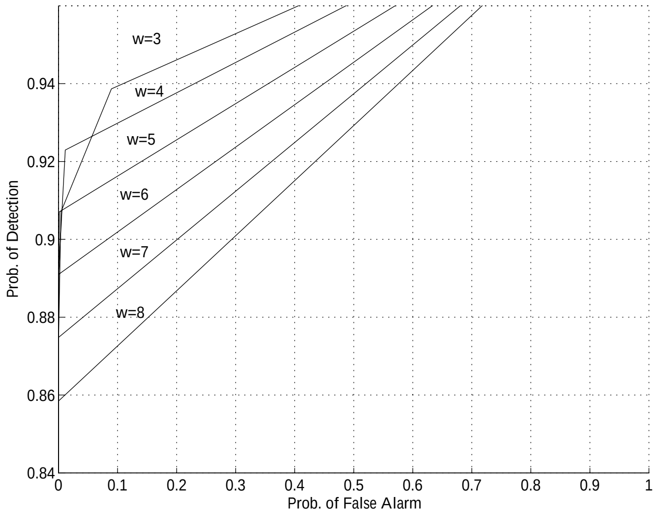


Figure 9.6: ROC curves for $\beta = 0.01$ $m = 50$, $a = 8$ and various values of w (3,...,8).

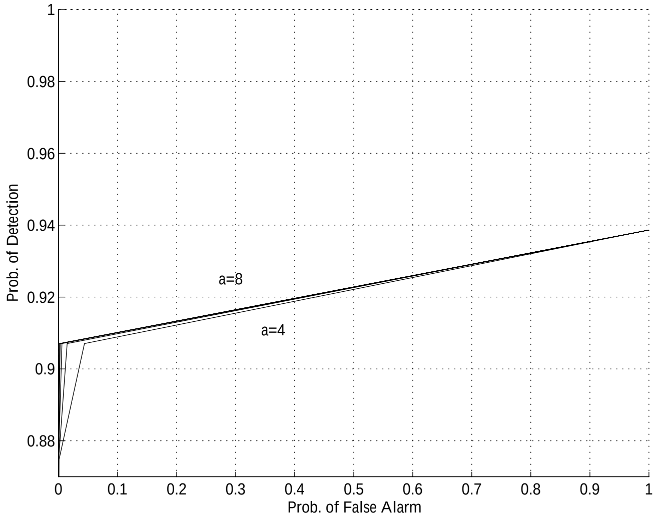
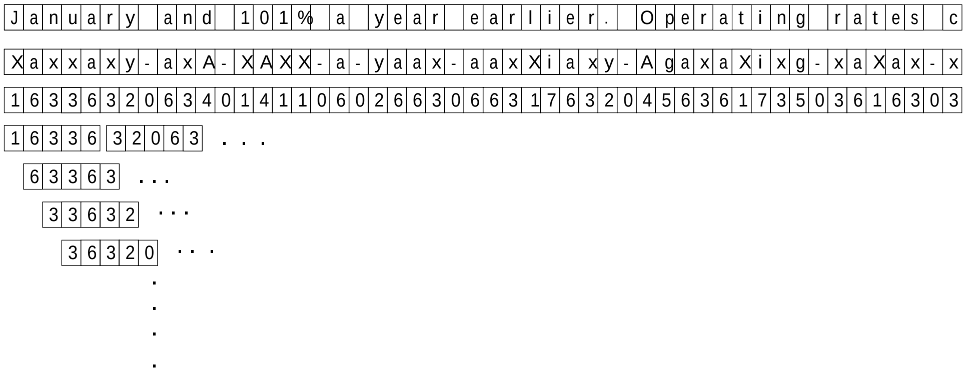


Figure 9.7: ROC curves for $\beta = 0.01$, $m = 50$, $w = 5$ and various values of a (4,...,8).

Index	Shape Code	Description	Members
0	-	space	*blank*
1	X	ascender	! % ( ) 1 2 3 5 7 < > ? C E F G H I J K L M N S T U V W X Y Z [ ] f h k l t
2	y	descender	, . _ y
3	e	xline	* + - : ; = c m n r s u v w x z
4	A	asc. with hole	# $ & 0 4 6 8 9 A B D O P Q R b d
5	g	des. with hole	g p q
6	a	xline with hole	a e o
7	i	ascender mark	i j ” ’ ˆ ‘

Table 9.3: Table of shape codes and symbols to which they apply.



```
J a n u a r y   a n d   1 0 1 %   a   y e a r   e a r l i e r .   O p e r a t i n g   r a t e s   c
X a x x a x y - a x A - X A X X - a - y a a x - a a x X i a x y - A g a x a X i x g - x a X a x - x
1 6 3 3 6 3 2 0 6 3 4 0 1 4 1 1 0 6 0 2 6 6 3 0 6 6 3 1 7 6 3 2 0 4 5 6 3 6 1 7 3 5 0 3 6 1 6 3 0 3
1 6 3 3 6   3 2 0 6 3  . . .
  6 3 3 6 3  . . .
    3 3 6 3 2  . . .
      3 6 3 2 0  . . .
```

Figure 9.8: Shape codes from sample signature **831** an

ages, each represented by a signature of length > 50. We c
of size 8 (shown in Table 9.3), and a key length of 5 (i.e., a
from each signature). An example of a text line is shown
its shape code signature and some of the index keys. The
database.

The experiment was divided into two parts. Part I exan
scores for known duplicates, corrupted duplicates and no
the database. Part II looks at the distribution of the ra
in the top 20 positions. To address the robustness of fe
and degradation model was built into the system. The mo
randomly perturbing a fixed number of shape codes in each
errors in shape coding.

In Part I of the experiment, a set of signatures known t
matched against the database. The results of matching a
(**831**) are shown in Table 9.4 for the cases where 0, 5, 10
introduced. Table 9.4 shows that although the match sco
there are only 5 errors, the correct document still has a si
alternative choices. The large difference between the top

Rank / *Error*	1	2	3	4	5	6	7	8
0	**831**(46)	3984(16)	839(14)	3734(14)	834(14)	3752(13)	828(13)	3990(13)
5	**831**(25)	708(10)	218(10)	1029(10)	2788(14)	2789(13)	4474(10)	4498(10)
10	**831**(25)	1990(7)	3827(7)	3984(7)	742(6)	3909(6)	2235(6)	4118(6)
15	**831**(9)	790(6)	839(6)	984(6)	1888(6)	2394(6)	2397(6)	2402(6)
20	387(3)	1421(3)	2066(3)	2952(3)	2955(3)	2959(3)	2990(3)	2994(3)

Table 9.4: Scores of line **831** in [candidate (score)] format.

match is not simply random, but rather well correlated with its candidate.

To test the robustness of the features, signatures of text lines that were not in the original database (i.e. documents known to be non-duplicates) were matched against the lines in the database. Table 9.5 shows the match scores for such a text line.

In Part II of the experiment, 100 signatures were randomly chosen from the 5000 in the database and perturbed to be matched against the 5000 signatures in the database. The correct match was recorded in the the top one, two, five, and ten positions. The results are shown in Table 9.6 when the candidates were corrupted with 0, 5, 10, 15 and 20 errors. In practice, the variation between two documents which are true duplicates is typically due to factors such as notes, photocopying and aging, and due to characteristics of the scanning process, including resolution, density and skew. It is expected that few “differences” exist in the shape codings of duplicate documents, so the introduction of 20 errors is more then sufficient.

In the second experiment a much larger database was considered for experimentation. The ASCII data was similar to the database of the first experiment but with millions of signatures. 2,500 non-duplicate and 2,500 randomly chosen duplicate signatures were used to simulate incoming documents.The first line of Table 9.7 shows the number of duplicates detected in the top 1, 2, 5, 10 and 20 positions[a].

The corrupted signatures of duplicate documents were used in the matching process, perturbing them by introducing fixed numbers of errors (5, 8 and 10). The numbers of duplicate documents detected in the top 1, 2, 5, 10, and 20 positions for these levels of errors are shown in the bottom three lines of Table 9.7. Figure 9.9 shows the percentage of detected duplicates as a function of the number of documents

[a] Some of the lines in the original database occur more than 20 times, so the “true” duplicate may not appear in the top 20. That is why even in the 0 error case, the 2500 documents were not all detected.

Rank Error	1	2	3	4	5	6	7	8
0	828(12)	834(10)	3390(10)	4598(10)	888(9)	2317(9)	2350(9)	223(9)
5	801(6)	822(6)	1872(6)	1882(6)	2208(6)	2635(6)	1959(5)	2159(5)
10	3390(8)	2684(6)	1823(6)	4530(6)	3373(5)	62(5)	4489(5)	1636(5)
15	2726(5)	2838(5)	397(4)	732(4)	941(4)	1593(4)	2027(4)	2097(4)
20	544(9)	2027(9)	4727(8)	1482(7)	489(7)	2274(7)	2620(7)	3383(7)

Table 9.5: Scores of line **5416**, which was not in our database, in [candidate (score)] format.

Added Errors	Top 1	Top 2	Top 5	Top 10	Top 20
0	100	100	100	100	100
5	100	100	100	100	100
10	100	100	100	100	100
15	51	58	69	77	100
20	17	21	24	30	100

Table 9.6: Top duplicate candidates in 100 queries.

Added Errors	Top 1	Top 2	Top 5	Top 10	Top 20
0	2416	2443	2458	2465	2467
5	2379	2419	2447	2457	2463
8	2059	2202	2327	2390	2431
10	1403	1678	1905	2039	2181

Table 9.7: Number of duplicate candidates detected in 2500 queries from a pool of one million documents.

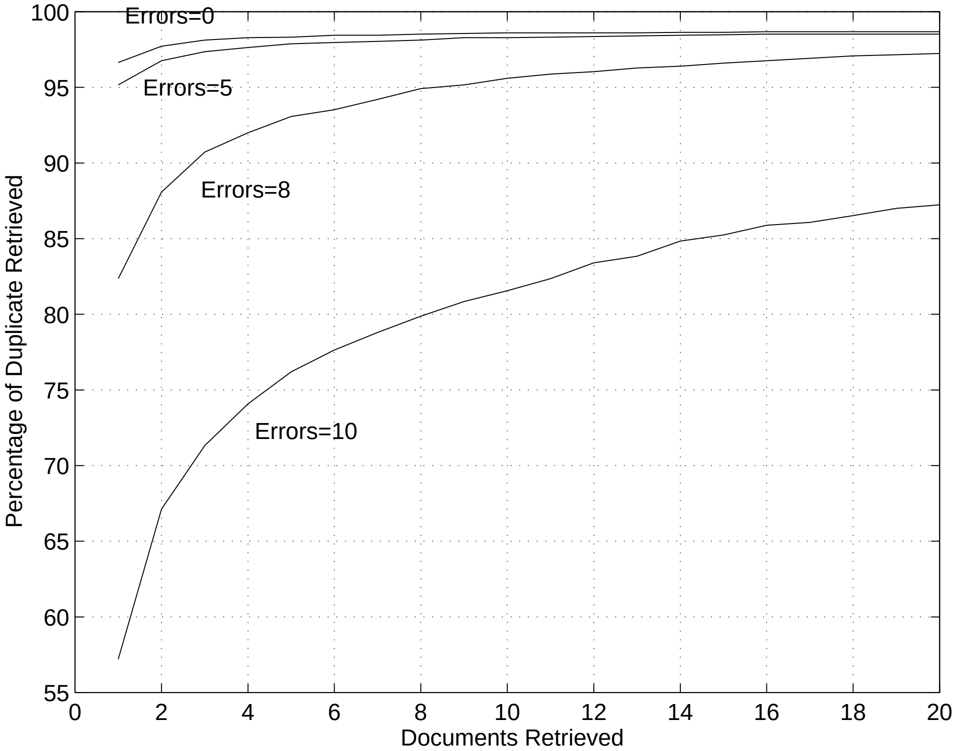


Figure 9.9: Percentage of duplicate documents identified in the top n candidate documents retrieved.

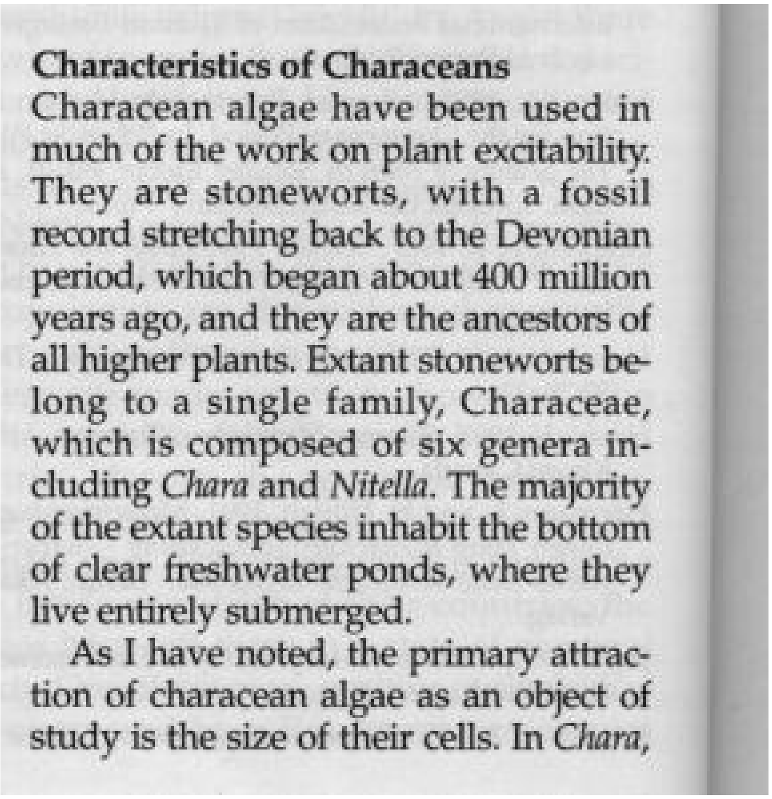
retrieved for each of the error levels.

9.2 Grayscale extension

The domain of document images is by no means limited to binary images. A document image, much like other images, can have pixels that span the entire color spectrum.

The majority of the work done with document images involves high-contrast black and white images which are ideal for bi-level image capturing. Our first extension from this baseline is to consider grayscale document images. While color document images need to eventually be considered, the grayscale extension is more widely applicable. Figure 9.10 shows a section of a grayscale document image, which exhibits a large amount of symbol-level redundancy and background texture. Given a successful application of component segmentation, clustering, background extraction, and residual coding, the application of symbolic compression is straightforward.

A survey of character segmentation was conducted by Casey and Lecolinet [19] and a method of grayscale segmentation of characters is given in [58]. In our preliminary implementations, we performed connected component analysis on a thresholded image. A successful grayscale segmenter is envisioned to be one that uses a combination of connected components based on ranges of thresholds and previously determined prototypes. The clustering algorithms are in general very similar to those used for



Characteristics of Characeans

Characean algae have been used in much of the work on plant excitability. They are stoneworts, with a fossil record stretching back to the Devonian period, which began about 400 million years ago, and they are the ancestors of all higher plants. Extant stoneworts belong to a single family, Characeae, which is composed of six genera including *Chara* and *Nitella*. The majority of the extant species inhabit the bottom of clear freshwater ponds, where they live entirely submerged.

As I have noted, the primary attraction of characean algae as an object of study is the size of their cells. In *Chara*,

Figure 9.10: Example of a grayscale document image.

clustering of binary components. The only difference is in the distance-based matching since distance is not strictly computable from a grayscale component. Other matching methods can address the problems of grayscale components. The two remaining tasks, background extraction and residual coding, need to be addressed for the successful implementation of grayscale symbolic compression.

9.2.1 Background extraction

Suppose that components have been segmented and removed from a grayscale document image. The resulting image is a "cut" version of the original image in which the pixel values have been set to 0 where components were detected and segmented. The remaining image can be considered to be mostly background and can be analyzed to determine its compressibility index. Comprehensive background removal was not attempted since it would involve background texture determination together with segmentation, clustering, and residual coding. However, once a background image is determined it will have the same characteristics as the cut image.

Figure 9.11 shows the distribution of the background pixel values. It shows that most pixels occur near the lower end of the grayscale spectrum. In this particular example, there exist no pixels above grayscale value 151 (on a scale of 0 to 255). The entropy of this background image comes to 5.5605 for a bit rate of 0.6951. Lossy representation of the background can be achieved in a number of ways, including

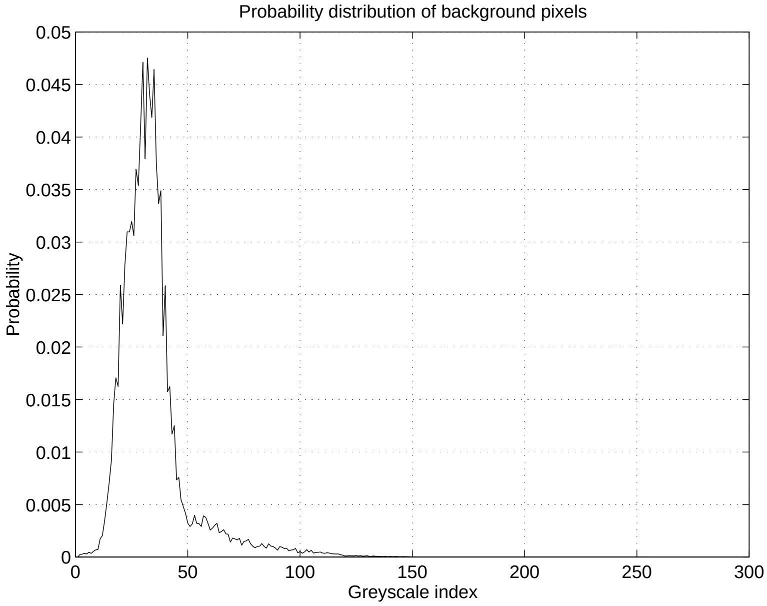


Figure 9.11: Probability distribution function of the background pixels.

resolution reduction, requantization of grayscale values, and multi-scale representation, among others. The background section can also tolerate a much higher lossy representation without affecting readability.

9.2.2 Residual coding

Figure 9.12 shows the set of prototypes assigned to the clusters of grayscale document image components. Instead of a distance transform, we use a gradient transform. This transform favors pixels whose neighborhoods are fairly flat over those that have higher gradients. We modify our hypothesis that informative pixels are farther from edges to a hypothesis that the informative pixels are in low-gradient areas.

Figure 9.13 shows examples of a segmented component, its assigned prototype, and its gradient transform. The gradient was calculated with a 3×3 window. It can be seen that the edges of characters, which exhibit the highest gradients, are shown as having darker pixels. The residual map and the ordered and unordered code of the example in Figure 9.13 are shown in Figure 9.14. It is easy to see that the degradation effects for binary images continue to hold for grayscale images.

9.2.3 Entropy analysis

To show this phenomenon more explicitly we can estimate the distribution of the residual pixels and the gradient values to arrive at an entropy value for the ordered and unordered code along with the mutual information between the unordered code and the distribution of the gradient map. This is analogous to the work done in

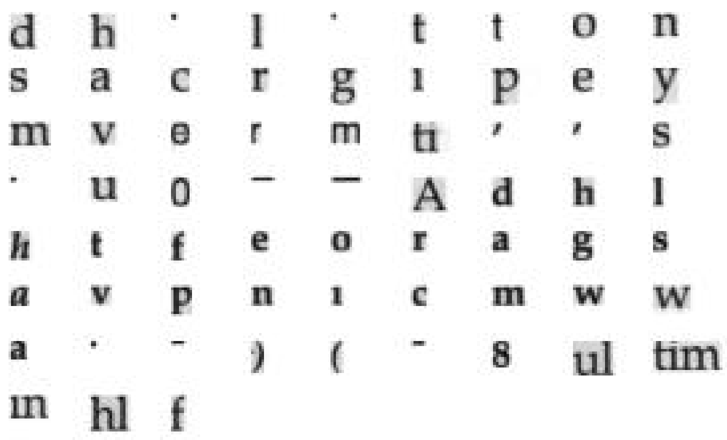
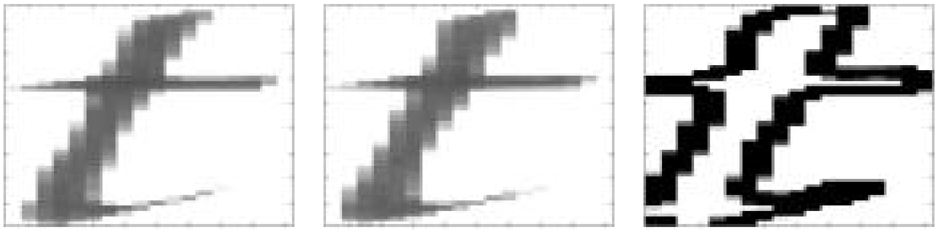
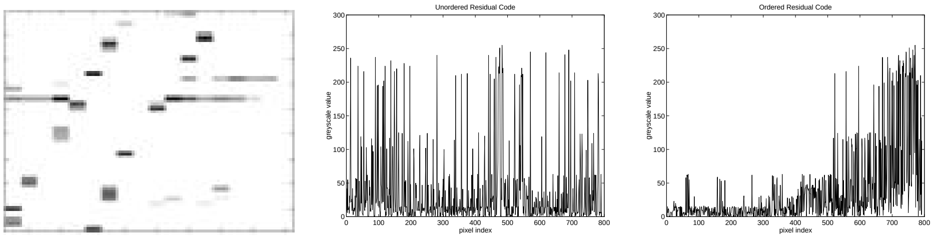


Figure 9.12: Prototype instances of a grayscale document image.



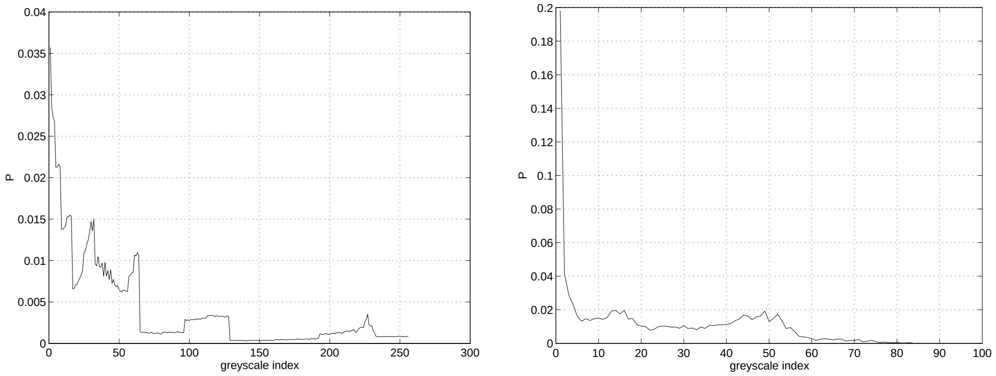
(a) (b) (c)

Figure 9.13: Example of a) grayscale component, b) assigned prototype, and c) gradient of the prototype.



(a) (b) (c)

Figure 9.14: Example of a) grayscale residual, b) unordered code, and c) ordered code.



(a) (b)

Figure 9.15: Distribution of a) the unordered pixels and b) the gradient values.

Chapter 5. Recall that the entropy of the unordered code is

$$H(U) = -\sum_{u=0}^{255} P_U(u) \times \log_2(P_U(u)) \tag{9.12}$$

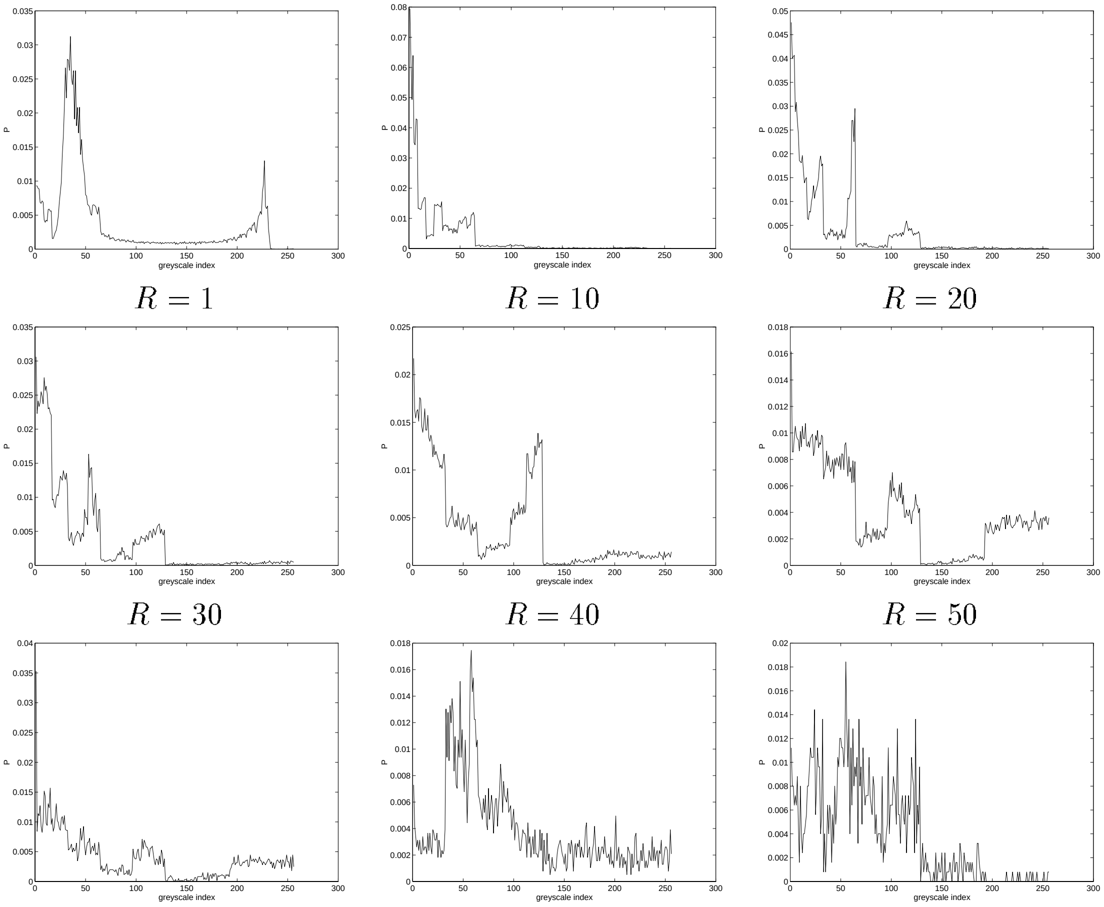
where $P_U(u)$ is the distribution of the residual pixels associated with a segmented component. The entropy of the ordered code is

$$\begin{aligned} H(O) &= H(U|R) \\ &= E_R[H(S_R)] \\ &= -\sum_{r=0}^{R_{max}} P_R(r) \sum_{s=0}^{255} P_{S_r}(s) \log_2(P_{S_r}(s)) \end{aligned} \tag{9.13}$$

and the mutual entropy which measures the improvement in coding is

$$\begin{aligned} I(U;R) &= H(U) - Y(U|R) \\ &= \sum_{s=0}^{255} \sum_{r=0}^{R_{max}} P_R(r) P_{S_r}(s) \log_2 \frac{P_{S_r}(s)}{P_U(s)} \end{aligned} \tag{9.14}$$

where R_{max} is the largest gradient, analogous to the largest distance in the case of binary images. Figure 9.15 shows the distribution of the unordered and the gradient values. The entropy of the unordered pixels comes to 0.8747 bits. The distribution of the unordered residual shows artifacts which could be the result of pixels occurring at different gradient levels. The gradient distribution of Figure 9.15 shows a non-constant distribution, which could provide better ordering and coupled with bit-allocation of residual pixels at different gradient levels could give desirable rate-distortion characteristics. Figure 9.16 show a set of plots which reflect the distribution of conditioned residual pixels. The distribution varies by a large amount and promises to provide a compact coding.



$R = 1$ $R = 10$ $R = 20$

$R = 30$ $R = 40$ $R = 50$

$R = 60$ $R = 70$ $R = 80$

Figure 9.16: Distribution of the ordered pixels conditioned on their gradient values for six different gradient conditions.

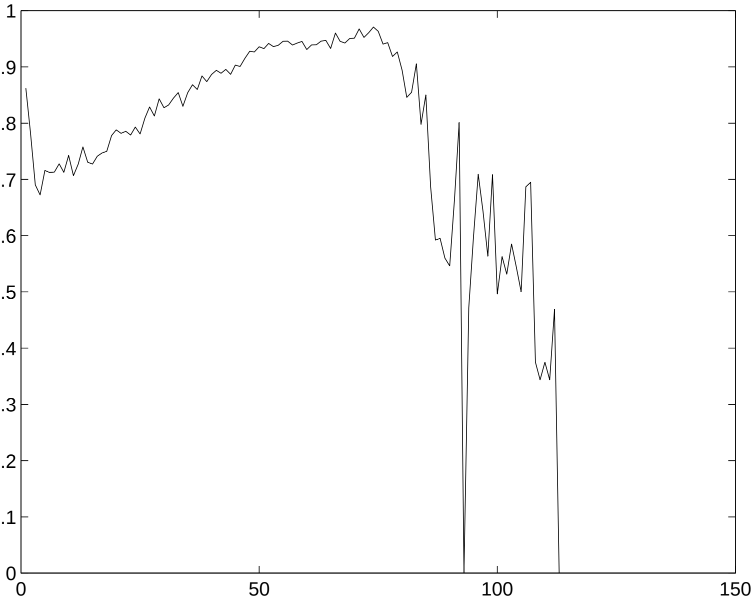


Figure 9.17: Entropy of U given $R = 1$ for $r = 1, ..., R_{max}$.

Figure 9.17 shows the entropy of pixels for each gradient value r. The trend of this plot is not as expected. We expected to observe lower entropy as r increases. This implies that we have not derived a sufficient condition R to apply to the unordered code U. This makes sense in that a 3×3 neighborhood does not provide an accurate measure of how residual pixels behave. The underlying principle should, however, hold in that edge pixels are more prone to degradation and pixels far from edges hold more information needed for correct recognition. In our example we measured the entropy of the unordered code to be 0.8747, and of the ordered code to be 0.8290, for an improvement of 0.0457 which translates into an improvement of 5 percent.

We are confident that the residual coding will improve. In any case, the developed infrastructure has a wide range of applicability to tasks that require a hierarchical representation. As was mentioned in Chapter 5, this infrastructure is immediately applicable to problems ranging from lossy compression and transmission to multimedia signal processing. The hierarchical representation can contribute in two ways. One is to the readability of the document, where an attempt is made to preserve the character structure, much as it was preserved for binary images. The second is to the fidelity of the image. While binarizing the document would renders it readable, it is much more esthetically pleasing to use shades of gray. The problem is to determine the best combination of fidelity and readability. For degraded documents an OCR engine might do better with a higher-fidelity image, and a balance can be reached that provides the best-recognizable document. However, at what point does a document's fidelity affect its readability? These problems still need to be addressed for a successful implementation of grayscale document image compression.

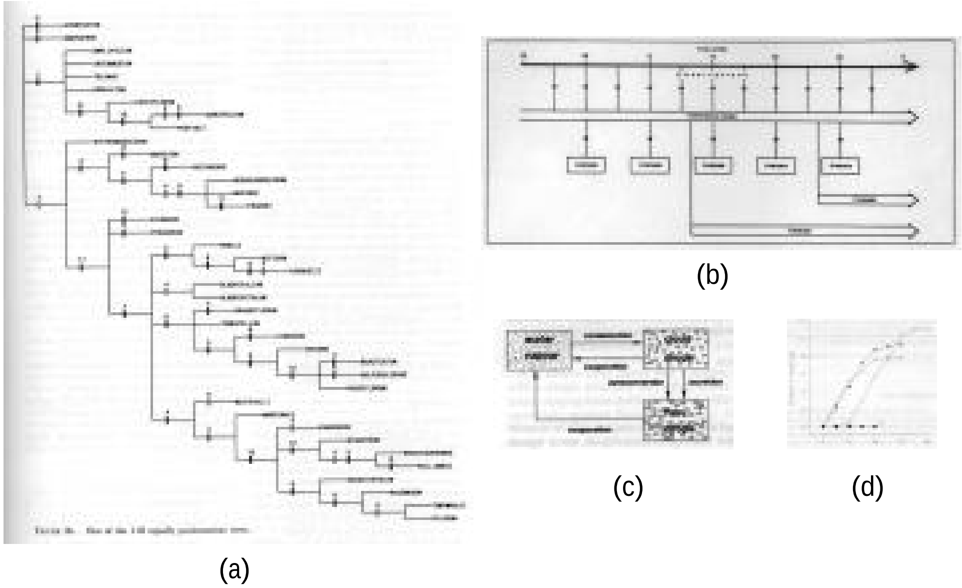


Figure 9.18: Examples of graphic images.

9.3 Graphic extension

It can be argued that graphical diagrams are specialized types of documents. These images do not fall into the category of natural images. They have a large amount of organization at several levels, and they are composed of mostly redundant elements. Examples of graphic images are shown in Figure 9.18.

The most redundant constituent in these images is a straight line segment. In determining a representative prototype, one needs to develop a segmentation method that produces sections of straight lines that can be indexed into the image and achieve compression. This may be hard to do since too-small lines will reduce compression by specifying too many location indexes. The effects of the residual may be minimal, but the application of residual coding should be addressed in determining an appropriate segmentation. Graphical analysis techniques (a review of some algorithms appears in [8]) can be applied toward graphic segmentation.

9.4 Networked databases

Network optimization is a task that can clearly benefit from our hierarchical organization. It is true that transmission capabilities have increased by a large amount, but it is also true that processing power has increased. This means that while we can deliver data faster, there exist processes that require larger amounts of data and in turn require higher data transfer rates. While there exist fast networks that do not need to be concerned about client-server information exchange, there exist remote client-server pairs that do not have optimized traffic and are constantly competing for network bandwidth. In the age of the world wide web, central database servers, and countless modem connections, many users are confronted with slow network speeds.

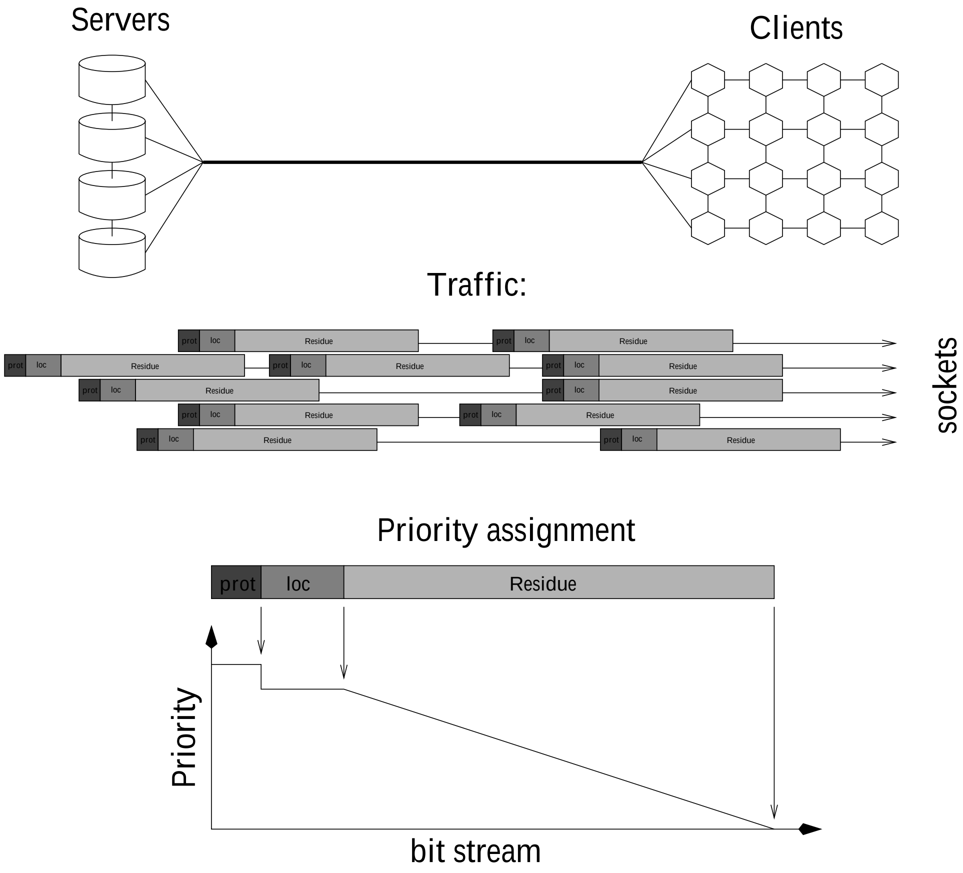


Figure 9.19: Network utilization for server-client transmissions.

Figure 9.19 depicts a server-client network where at a particular time multiple sockets are busy transferring data. If uniform network resources are allocated, the informative prototype shapes and their locations are weighted equally with the residual pixels (if any), without any regard to total amount of service provided to a group of users. Service can be defined by the amount of readable information transferred and a priority assignment can easily be devised so that network traffic is optimized in the sense of providing the best service to the greatest number of users.

This approach can easily be extended to packet-transport networks and ATM networks that use quality-of-service indexes. A number of concerns must be addressed when faced with these issues. While textured images, video, and sound have traditionally been considered in connection with these problems, document images have not, and need to be, addressed [51].

9.5 Hyperdocument organization and transmission

The usage of documents has become very flexible in recent years. While it has been shown that documents hold information at various levels [24] that must be managed and retrieved, other media domains have also been integrated into their structure. Document processing with multimedia tools have been studied [102] but new characteristics, beyond synthetic languages like SGML or definitions like MHEG, need to be addressed [92].

Hyperdocuments offer a unique extension to the traditional functionality of a document by extending the semantic meaning of document objects to include multimedia objects. This is different from a synthetic markup language in that a physical document is used and referenced, while cross-media objects, from physical sources, are anchored to mark up the document.

The implications of this degree of freedom are great. For instance, consider a field manual, originally derived from text manuals having a high level of structure (title, paragraphs, graphs, annotated text and images). Field technicians can use a hyperdocument interface tool to annotate the document in various media (video, audio, and/or data such as "We have observed new cracks in the secondary valves ...") and design engineers can use a hyperdocument management and browsing tool to gain access to sections of the field manual.

Cooperative learning, reviewing, and project management could all benefit from this framework. The basic idea here is that there exist a single physical based document which is used by multiple sources. By annotating the document appropriately, conferencing occurs which allows for a fluent dissemination of information for the document readers. For media distribution, physical media types can easily be integrated, such as assembly procedures, to be delivered to customers or co-workers. The use of this concept provides a logical connection from a physical media domain to a multi-media markup domain. In achieving that goal, a large number of issues need to be addressed [92].

9.6 Summary

We have developed a methodology for detecting duplicate document images in a heterogeneous database. This can aid in maintaining document image databases. We are able to perform extremely fast duplicate detection on millions of documents. The first scan is able to reduce the possible duplicates from millions to tens or hundreds of documents. A finer, but more computationally intensive, duplicate detection algorithm was also suggested which can be used in reducing possible duplicates from hundreds to tens or lower by use of a symbolic approach. A final reduction can be done by comparing OCR results.

Several extensions to our base image compression effort have been suggested. The first natural extension of binary document images is grayscale images. Some preliminary results show that some amount of improvement in coding can be obtained; however, there exist fundamental concerns regarding the readability and fidelity of the image and how they interact. After these concerns are addressed, the performance of lossy compression and progressive transmission needs to be assessed along with efficiency of coding for document processing, e.g. keyword searching, skew estimation and correction, partial retrieval, etc. Once the grayscale extension is accomplished, the further extension to color images can be addressed. Another logical extension is to the compression of images which are composed of patterns that are connected. In our compression scheme we took a connected component approach since the patterns in the image appeared as disconnected components. However, graphic images, as well as documents in languages that are composed of connected characters, have constituent patterns that appear continuously. For a successful implementation of our approach for these image types, an alternative segmentation method needs to be used.

Representation and coding of images are only parts of a comprehensive solution to the class of information archiving and management problems. The transmission of information is an inherent part of information dissemination and needs to be addressed for efficient usage of limited resources. Often, resource allocation and usage are liberal, assuming unlimited resources. These practices fail to perform well in cases where the resources per user are limited. Technological advances have made optimized resource allocation possible and need to be addressed for network resources.

Finally, integration of other media into a document's structure was suggested. With technological advances bringing massive amounts of information to users, intelligent use and management of various media is desired. This goes beyond the confines of a markup language and widens the usage of classical documents. When a classical document image is viewed, a standard organization exists, such as title, author, paragraphs, images, footnotes, etc. This is easily extended to include other media domains. Organization of these components has been addressed to some extent [24] but fundamental problems in compression, transmission, and processing need to be addressed.

Chapter 10

Conclusion and future work

Computer systems have made significant contributions to the managagement and transmission of images in many forms. Transmission and processing remains an active research problem in many image domains. In an attempt to optimize retrieval of document images, it was observed that document images contain most of their meaningful information within their components. Since the components are samples of a specific language, redundancy in their occurrence can be exploited to provide a highly efficient representation. We developed a technique which extracted constituent components, clustered them, and represented them in a compressed form. While this technique was developed independently, it significantly extends the technique proposed by Witten et al. [110], and took its roots from work done by Ascher and Nagy [9].

Traditional coding, transmission, and lossy representation of images often made use of resolution-reduction practices. This was ideal for textured images but document images do not fit easily into that class. Document images lose their meaningful fine structure much faster than textured images when resolution is lowered. In the process of replacing an image component by the prototype of a representative cluster, we do not reduce resolution; instead, we reduce the allowable level of variability between image components. This effectively produces a lossy representation which is still readable and processable.

10.1 Summary

We first devised a data representation scheme that formulated the constituent parts of our algorithm. This representation was designed to not only provide compression, but also facilitate processing while in the compressed domain. Since the task of decompression is more computationally intensive than that of retrieval, we decomposed the representation into logical building blocks and introduced indexes for large blocks which benefit from partial decompression. The design of this representation helps in the identification and construction of various processing modules that enhance and augment the functionality of the compression. In addition to building the compression/decompression routines, we were able to conclude that component segmentation and clustering could benefit from joint processing and that eventually a global operations manager could optimize the overall representation.

We considered a number of clustering algorithms for incorporation into the above model. Due to the size and population variability of components, several well-known clustering algorithms were dismissed as having undesirable computational properties. We then considered a matching and classifying approach which had desirable properties and also provided the freedom of choosing a matching function that attempts to preserve some component properties. We considered the use of Hamming distance, weighted Hamming distance, and weighted AND-NOT matching functions for use in our clustering algorithm. We also considered entropy-based functions, and suggested a distance-based matching method that preserved component structures. This yielded a clustering algorithm that grouped components of similar structure. We argue that structural components contribute the most toward the correct recognition of characters. We also improved on the clustering methodology by considering a joint segmentation and clustering algorithm. This has proven to be useful for images which contain large numbers of connected characters. The method starts with the connected components of an image and derives a set of prototypes. The prototypes are then used to revisit all the components and determine an alternate segmentation strategy that attempts to segment characters which are joined together. A simple feedback construction allows for effective calculation of meaningful prototypes which can improve compressed-domain processing.

Residual coding constitutes a major portion of the image coding. We developed a novel approach to residual coding with desirable compression and transmission characteristics. Our original method was to provide a hierarchical representation in which the image representation based on only the prototypes defined the baseline. Any subsequent information was considered to be less important and farther down in the hierarchy. Since residual coding was necessary for effective representation of the image (without it, the image might be misinterpreted) we proposed a hierarchical ordering of residual information to allow for higher compression and desirable lossy representation. The basis of this code is the fact that all pixels are not created equal; while some pixels contribute to correct recognition of a component, other pixels do not. We pursued a distance-based ordering which favored more informative pixels and attempted to preserve character structure for lossy compression. Two other structural coding methods were also suggested which provided enhancements to the distance ordering by observing that structural components consist of groupings of pixels and that the groupings are most likely connected to the prototype. Favorable compression performance was shown with desirable lossy representation.

A rate-distortion analysis was performed with a measure of distortion based on OCR and distance-based matching performance. The OCR performance, with respect to various achievable rates, showed that distance ordering is clearly more favorable than row ordering. We experimented with various intermediate representations using fractional residual information and observed the performance of the OCR engine with respect to the available ground truth data. The performance of the OCR engine at the highest levels of lossy compression was very high (80%), guaranteeing that the OCR engine was operating under near-peak conditions, and the performance was observed to increase to its maximum (95%) when all of the residual information was represented. We also considered a rate-distortion tradeoff using the distance-based match as a measure of distortion. This allows for automatic generation of a lossy representation that gives optimally low distortion based on the achievable entropy rate.

A set of applications that utilize our data representation was then presented.

The basic compression and decompression routine formed the basis for these applications. We demonstrated successful implementation of lossy compression, progressive transmission, sub-document retrieval, skew estimation and correction, and keyword searching. Scalable processing was demonstrated for lossy compression, progressive transmission, and sub-document retrieval, and compressed-domain processing was exhibited for skew estimation and correction and keyword searching. We also used information about character shapes to perform duplicate detection of documents in large databases.

Due to the large number of immediate extensions to our work, a full section was devoted to this topic. Preliminary results on some concepts showed promise for the success of future developments. Our basic methodology appears to be applicable to other common image and networking problems.

10.2 Future work

Immediate work should emphasize concepts discussed under extensions (Chapter 9), especially those involving grayscale images and graphic images. It is envisioned that in the immediate future, these extensions will have the most impact. Extensions to network databases are also interesting and should be pursued, but only in conjunction with research on multi-media databases, and only if justified by market demand. In the long term, hypermedia extensions will prove to be a valuable research topic with a large number of market applications.

In other areas not yet mentioned, integration of wavelet decomposition will prove to be valuable. Research on binary wavelet decomposition has given good results, but in the realm of display technology rather than compression, transmission, and processing technologies. The idea is that if a screen of low resolution is used to render an image, it does not seem efficient to transmit a high-resolution image. The problem is then to decompose the image into desired scales and transmit based on those scales. Since binary wavelets are rare and their bit allocation is a hard problem, their use in binary image compression was not pursued. However, for grayscale images, there may exist sets of wavelets that are tunable based on prototypes to create the best scalable decomposition through a resolution path. The idea of wavelet decomposition should be reexamined for the case of grayscale images.

Integration with MG (Managing Gigabyte), and creation of a code that can achieve low entropy will prove to be extremely beneficial. Integration with MG should provide higher compression; it will also add scalable lossy and progressive representation, which MG currently lacks. It will also promote compressed-domain processing using its representation. Performing compressed-domain processing relies on performing indexable compression. A coding method needs to be pursued that can achieve close to the entropies mentioned in Chapter 5 while allowing for partial decompression. It is also desirable to devise coding methods that will allow for scalable error coding, for eventual use in source-channel coding.

After studying the symbolic approach and its rate-distortion tradeoff, we can predict that distance ordering and structural decomposition may provide an approach to document image enhancement or to determining image quality prior to performing any processing. Since we have used a degradation model which relates well to correct recognition (OCR-based rate-distortion), we may extend this model to either alter the image for better recognition or calculate a quality index. While the use of compressed-domain processing in OCR applications is logical, it is also logical to have built-in OCR correction mechanisms based on our representation. This will augment the standard word dictionary or other lexicon-based checking.

The final extension of our method is envisioned to address compression in other image domains such as satellite or aerial images, biomedical images, and so on. The concepts of segmentation, clustering, and residual coding need to be addressed for these domains; this may lead to significant progress in the compression and processing of general images.

Bibliography

[1] J. Abate. Linear adaptive delta modulation. *Proceedings of the IEEE*, 55:298–308, 1967.

[2] N. Ahmed, T. Natarajan, and K. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, 23:90–93, 1974.

[3] S. Alexander and S. Rajala. Image compression results using LMS adaptive algorithm. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 33:712–717, 1985.

[4] M.R. Anderberg. *Cluster Analysis for Applications.* Academic Press, 1973.

[5] G. Anderson and T. Huang. Piecewise Fourier transformation for picture bandwidth compression. *IEEE Transactions on Communications*, 19:133–140, 1971.

[6] H. Andrews. *Computer Techniques in Image Processing.* Academic Press, 1970.

[7] A. Antonacopoulos and R. Ritchings. Flexible page segmentation using the background. In *Proceedings of the International Conference on Pattern Recognition*, pages 339–344, 1994.

[8] J. Arias, R. Kasturi, and A. Chhabra. Evaluating the performance of techniques for the extraction of primitives from line drawings composed of horizontal and vertical lines. In *Proceedings of the Document Analysis Systems Workshop*, pages 191–204, 1996.

[9] R. Ascher and G. Nagy. A means for achieving a high degree of compaction on scan-digitized printed text. *IEEE Transactions on Computers*, 23:1174–1179, 1974.

[10] M. Atallah, Y. Genin, and W. Szpakowski. Pattern matching image compression: Algorithmic and empirical results. Technical Report CSD TR-95-083, Computer Science Department, Purdue University, 1995.

[11] H.S. Baird. The skew angle of printed documents. In *Proceedings of the SPSE 40th Annual Conference and Symposium on Hybrid Imaging Systems*, pages 21–24, 1987.

[12] M.F. Barnsley and L.P. Hurd. *Fractal Image Compression.* A.K. Peters, 1993.

[13] G. Bessho, K. Ejiri, and J.F. Cullen. Fast and accurate skew detection algorithm for a text document or a document with straight lines. In *Proceedings of the SPIE - Document Recognition*, volume 2181, pages 133–140, 1994.

[14] D. Bloomberg and L. Vincent. Blur hit-miss transform and its use in document image pattern detection. In *Procceding of the SPIE- Document Recognition II*, volume 2422, pages 278–292, 1995.

[15] D. Bodson, S. Urban, A. Deutermann, and C. Clarke. Measurement of data compression in advanced Group 4 facsimile system. *Proceedings of the IEEE*, 73:731–739, 1985.

[16] T. Boult. Dynamic digital distance maps in two dimensions. *IEEE Transactions on Robotics and Automation*, 6:590–597, 1990.

[17] R. Boyer and J. Moore. A fast string matching algorithm. *Communications of the ACM*, 20:762–772, 1977.

[18] H. Cai and G. Mirchandani. Wavelet transform and bit-plane encoding. In *Proceedings of the International Conference on Image Processing*, volume I, pages 578–581, 1995.

[19] R. Casey and E. Lecolinet. Strategies in character segmentation: A survey. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 1028–1033, 1995.

[20] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms.* MIT Press, 1990.

[21] T. Cover and J. Thomas. *Elements of Information Theory.* John Wiley and Sons, 1991.

[22] C. Cutler. Differential quantization of communication signals. U.S. Patent 2 605 361, 1952.

[23] D. Doermann, H. Li, and O. Kia. Duplicate document image detection. Technical Report 3739, Center for Automation Research, University of Maryland, 1997.

[24] D. Doermann, C. Shin, A. Rosenfeld, H. Kauniskangas, J. Sauvola, and M. Pietikainen. The development of a general framework for intelligent document image retrieval. In *Proceedings of the Document Analysis Systems Workshop*, pages 605–632, 1996.

[25] K. Etemad, D. Doermann, and R. Chellappa. Multiscale document page segmentation using soft decision integration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:92–96, 1997.

[26] N. Faller. An adaptive system for data compression. In *Proceedings of the Asilomar Conference on Circuits, Systems and Computers*, pages 593–597, 1973.

[27] B. Fino. Relations between Haar and Walsh/Hadamard transforms. *Proceedings of the IEEE*, 60:647–648, 1972.

[28] L. Fletcher and R. Kasturi. A robust algorithm for text string separation from mixed text/graphics images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10:910–918, 1988.

[29] R. Gallager. Variations on a theme by Huffman. *IEEE Transactions on Information Theory*, 24:668–674, 1978.

[30] A. Gersho and R. Gray. *Vector Quantization and Signal Compression.* Kluwer Academic Publishers, 1992.

[31] A. Habibi. Survey of adaptive image coding techniques. *IEEE Transactions on Communications*, 25:1275–1284, 1977.

[32] R. Haralick. UW English document image database I: A database of document images for OCR research. CDROM.

[33] T. Hirata. A unified linear-time algorithm for computing distance maps. *Information Processing Letters*, 58:129–133, 1996.

[34] J. Hobby and H. Baird. Degraded character image restoration. In *Proceedings of the Symposium on Document Analysis and Information Retrieval*, pages 233–245, 1996.

[35] M. Holt. A fast binary template matching algorithm for document image data compression. In J. Kittler, editor, *Pattern Recognition*, pages 230–239. Springer Verlag, 1988.

[36] M. Holt and C. Xydeas. Recent developments in image data compression for digital facsimile. *ICL Technical Journal*, pages 123–146, 1986.

[37] P. Howard. Lossless and lossy compression of text images by soft pattern matching. In *Proceedings of the IEEE Data Compression Conference*, pages 210–219, 1996.

[38] T. Huang. Run length coding and its extensions. In T. Huang and O. Tretiak, editors, *Picture Bandwidth Compression*, pages 231–264. Gordon and Breach, 1972.

[39] J.J. Hull. Document image matching and retrieval with multiple distortion- invariant descriptors. In *Proceedings of the International Workshop on Document Analysis Systems*, pages 383–400, 1994.

[40] J.J. Hull. Presented at SPIE - *Document Recognition IV*, 1997.

[41] R. Hunter and A. Robinson. International digital facsimile coding standards. *Proceedings of the IEEE*, 68:854–867, 1980.

[42] D. Huttenlocher, G. Klanderman, and W. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:850–863, 1993.

[43] S. Inglis and I. Witten. Compression-based template matching. In *Proceedings of the IEEE Data Compression Conference*, pages 106–115, 1994.

[44] A. Jain. *Fundamentals of Digital Image Processing.* Prentice-Hall, 1989.

[45] A. Jain and S. Bhattacharjee. Text segmentation using Gabor filters for automatic document processing. *Machine Vision and Applications*, 5:169–184, 1992.

[46] N. Jayant. Adaptive delta modulation with a one-bit memory. *Bell Systems Technical Journal*, 49:321–343, 1970.

[47] O. Johnsen, J. Segen, and G. Cash. Coding of two-level pictures by pattern matching and substitution. *Bell System Technical Journal*, 62:2513–2545, 1983.

[48] T. Kanungo, R.M. Haralick, and I.T. Phillips. Global and local document degradation models. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 730–734, 1993.

[49] O. Kia and D. Doermann. Structure-preserving image compression and transmission. In *Proceedings of the International Conference on Image Processing*, volume I, pages 193–196, 1996.

[50] O. Kia and D. Doermann. Symbolic compression for document analysis. In *Proceedings of the International Conference on Pattern Recognition*, volume III, pages 664–668, 1996.

[51] O. Kia and D. Doermann. Document image coding for processing and retrieval. In *Proceedings of the IEEE Workshop on Multimedia Signal Processing*, 1997. To appear.

[52] O. Kia and D. Doermann. Residual coding in document image compression. Technical report, Center for Automation Research, University of Maryland, 1997. To appear.

[53] O. Kia, D. Doermann, and R. Chellappa. Compressed-domain document retrieval and analysis. In *Proceedings of the SPIE - Multimedia Storage and Archiving Systems*, volume 2916, pages 176–187, 1996.

[54] O. Kia, D. Doermann, A. Rosenfeld, and R. Chellappa. Symbolic compression and processing of document images. Technical Report CAR TR-849, Center for Automation Research, University of Maryland, 1997.

[55] D. Knuth. Optimal binary search trees. *Acta Informatica*, 1:14–25, 1971.

[56] D. Knuth, J. Morris, and B. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6:323–350, 1977.

[57] A. Kolmogorov. Interpolation and extrapolation of stationary random series. *Journal of the Soviet Academy of Science*, pages 3–14, 1941.

[58] S.W. Lee, D.J. Lee, and H.S. Park. A new methodology for gray-scale character segmentation and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18:1045–1050, 1996.

[59] T. Lei, N. Scheinberg, and D. Schilling. Adaptive delta modulation system for video encoding. *IEEE Transactions on Communications*, 25:1302–1314, 1977.

[60] F. Leymarie and M. Levine. Fast raster scan distance propagation on the discrete rectangular lattice. *CVGIP: Image Understanding*, 55:84–94, 1992.

[61] J. Lim. *Two-Dimensional Signal and Image Processing.* Prentice-Hall, 1990.

[62] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28:84–95, 1980.

[63] J. Liu, C.M. Lee, and R.B. Shu. An efficient method for the skew normalization of a document image. In *Proceedings of the International Conference on Pattern Recognition*, volume III, pages 122–125, 1992.

[64] T. Luczak and W. Szpakowski. A lossy data compression based on an approximate pattern matching. Technical Report CSD TR-94-072, Computer Science Department, Purdue University, 1995.

[65] C. Maa. Identifying the existence of bar codes in compressed images. *CVGIP: Graphical Models and Image Processing*, 56:352–356, 1994.

[66] J. Makhoul, S. Roucos, and H. Gish. Vector quantization in speech coding. *Proceedings of the IEEE*, 73:1551–1558, 1985.

[67] U. Manber. Finding similar files in a large file system. In *Proceedings of the USENIX Conference*, pages 1–10, 1994.

[68] A. Mazzarri and R. Leonardi. Perceptual embedded image coding using wavelet transforms. In *Proceedings of the International Conference on Image Processing*, volume I, pages 586–587, 1995.

[69] K. Mohiuddin. *Pattern Matching with Application to Binary Image Compression.* PhD thesis, Stanford University, 1982.

[70] K. Mohiuddin, J. Rissanen, and R. Arps. Lossless binary image compression based on pattern matching. In *Proceedings of the International Conference on Computers, Systems, and Signal Processing*, pages 447–451, 1984.

[71] S. Mori, C.Y. Suen, and K. Yamamoto. Historical review of OCR research and development. *Proceedings of the IEEE*, 80:1029–1058, 1992.

[72] G. Nagy, P. Sarkar, D. Lopresti, and J. Zhou. Spatial sampling of printed patterns. Draft, 1996.

[73] Y. Nakano, Y. Shima, H. Fujisawa, J. Higashino, and M. Fujinawa. An algorithm for the skew normalization of document image. In *Proceedings of the International Conference on Pattern Recognition*, pages 8–13, 1990.

[74] L. O'Gorman. The document spectrum for page layout analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:1162–1173, 1993.

[75] L. O'Gorman and R. Kasturi. *Document Image Analysis.* IEEE Computer Society Press, 1995.

[76] J. O'Neal. Predictive quantization system (differential pulse code modulation) for the transmission of television signals. *Bell Systems Technical Journal*, 45:689–721, 1966.

[77] D. O'Shaughnessy. *Speech Communication: Human and Machine.* Addison-Wesley, 1987.

[78] A. Parker and J. Hamblen. Computer algorithms for plagiarism detection. *IEEE Transactions on Education*, 32:94–99, 1989.

[79] T. Pavlidis. Page segmentation by white streams. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 945–953, 1991.

[80] T. Pavlidis and J. Zhou. Page segmentation and classification. *CVGIP: Graphical Models and Image Processing*, 54:484–496, 1992.

[81] W. Pennebaker and J. Mitchell. Probability estimation for the Q-Coder. *IBM Journal of Research and Development*, 32:737–752, 1988.

[82] W. Pennebaker, J. Mitchell, G. Langdon, and R. Arps. An overview of the basic principles of the Q-Coder adaptive binary arithmetic coder. *IBM Journal of Research and Development*, 32:717–726, 1988.

[83] P. Pirsch. Adaptive intra/interframe DPCM coder. *Bell Systems Technical Journal*, 61:747–764, 1982.

[84] W. Postl. Detection of linear oblique structures and skew scan in digitized documents. In *Proceedings of the International Conference on Pattern Recognition*, pages 687–689, 1986.

[85] W. Pratt, P. Capitant, W. Chen, E. Hamilton, and R. Willis. Combining symbol matching facsimile data compression system. *Proceedings of the IEEE*, 68:786–796, 1980.

[86] K. Rose. *Deterministic Annealing, Clustering and Optimization.* PhD thesis, California Institute of Technology, 1991.

[87] A. Rosenfeld and J. Pfaltz. Distance functions on digital pictures. *Pattern Recognition*, 1:33–61, 1968.

[88] W. Rucklidge. Efficient computation of the minimum Hausdorff distance for visual recognition. Technical Report TR94-1454, Computer Science Department, Cornell University, 1994.

[89] H. Garcia-Molina S. Brin, J. Davis. Copy detection mechanisms for digital documents. In *Proceedings of the ACM SIGMOD Annual Conference*, 1995.

[90] H. Samet. Distance transform for images represented by quadtrees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4:298–303, 1982.

[91] P. Sarkar. Random phase spatial sampling effects in digitized patterns. Master's thesis, Rensselaer Polytechnic Institute, 1994.

[92] J. Sauvola and O. Kia. Hyperdocument management for compression, transmission, and processing. In *Proceedings of the IEEE Workshop on Multimedia Signal Processing*, 1997. To appear.

[93] J. Sauvola and M. Pietikainen. A document management interface utilizing page decomposition and content-based compression. In *Proceedings of the International Conference on Pattern Recognition*, volume III, pages 752–757, 1996.

[94] R. Sennhauser and K. Ohnesorge. Document image compression using document analysis and block-class-specific data compression methods. In *Proceedings of the SPIE - Image and Video Compression*, volume 2186, pages 146–155, 1994.

[95] C. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27:623–656, 1948.

[96] N. Shivakumar and H. Garcia-Molina. Scam: A copy detection mechanism for digital documents. In *Proceedings of the Second International Conference on Theory and Practice of Digital Libraries*, 1995.

[97] C. Song, J. Garondick, and D. Schilling. A variable-step-size robust delta modulator. *IEEE Transactions on Communications*, 19:1033–1044, 1971.

[98] A. Spitz. An OCR based on character shape codes and lexical information. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 723–728, 1995.

[99] A. Spitz. Using character shape codes for word spotting in document images. In *Shape, Structure, and Pattern Recognition*, pages 382–389. World Scientific, Singapore, 1995.

[100] A.L. Spitz. Skew determination in CCITT Group 4 compressed document images. In *Proceedings of the First Symposium on Document Analysis and Information Retrieval*, pages 11–25, 1992.

[101] A.L. Spitz. Logotype detection in compressed images using alignment signatures. In *Proceedings of the Fifth Symposium on Document Analysis and Information Retrieval*, pages 303–310, 1996.

[102] R. Steinmetz and K. Nahrstedt. *Multimedia: Computing, Communications and Applications*. Prentice-Hall, 1995.

[103] H. Tanaka and A. Kogawara. High-speed string edit methods using hierarchical files and hashing technique. In *Proceedings of the International Conference on Pattern Recognition*, pages 334–336, 1988.

[104] B. Verwer, P. Verbeek, and S. Dekker. An efficient uniform cost algorithm applied to distance transforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:425–429, 1989.

[105] M. Vetterli. Multi-dimensional sub-band coding: Some theory and algorithms. *Signal Processing*, 6:97–112, 1984.

[106] J. Vitter. Design and analysis of dynamic Huffman codes. *Journal of the Association for Computing Machinary*, 34:825–845, 1987.

[107] F. Wahl. A new distance mapping and its use for shape measurement on binary patterns. *Computer Vision, Graphics and Image Processing*, 23:218–226, 1983.

[108] F. Wahl, K. Wong, and R. Casey. Block segmentation and text extraction in mixed text/image documents. *Computer Graphics and Image Processing*, 20:375–390, 1982.

[109] R. Williams. *Adaptive Data Compression*. Kluwer Academic Publishers, 1991.

[110] I. Witten, T. Bell, H. Emberson, S. Inglis, and A. Moffat. Textual image compression: Two-stage lossy/lossless encoding of textual images. *Proceedings of the IEEE*, 82:878–888, 1994.

[111] I. Witten, T. Bell, M. Harrison, M. James, and A. Moffat. Textual image compression. In *Proceedings of the IEEE Data Compression Conference*, pages 42–51, 1992.

[112] I. Witten, A. Moffat, and T. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Van Nostrand Reinhold, 1994.

[113] K. Wong, R. Casey, and F. Wahl. Document analysis system. *IBM Journal of Research and Development*, 26:647–656, 1982.

[114] J. Woods and S. O'Niel. Subband coding of images. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 34:1278–1288, 1986.

[115] T. Yan and H. Garcia-Molina. Duplicate detection in information dissemination. In *Proceedings of the Very Large Database Conference*, 1995.

[116] Q. Zhang and J. Danskin. Entropy-based pattern matching for document image compression. In *Proceedings of the International Conference on Image Processing*, pages 221–224, 1996.

[117] Y. Zhang and L. Po. Fractal color image compression using vector distortion measure. In *Proceedings of the International Conference on Image Processing*, volume III, pages 276–279, 1995.

[118] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23:337–343, 1977.

[119] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24:530–536, 1978.