Journal of VLSI Signal Processing 20, 121–135 (1998)

Document Image Coding for Processing and Retrieval

OMID E. KIA
National Institute of Standards and Technology, Mathematical and Computational Sciences Division, Building 820, Room 365, Gaithersburg, MD 20899

DAVID S. DOERMANN
Language and Media Processing Laboratory, Center for Automation Research, University of Maryland, College Park, MD 20742

Abstract. Document images belong to a unique class of images where the information is embedded in the language represented by a series of symbols on the page rather than in the visual objects themselves. Since these symbols tend to appear repeatedly, a domain-specific image coding strategy can be designed to facilitate enhanced compression and retrieval. In this paper we describe a coding methodology that not only exploits component-level redundancy to reduce code length but also supports efficient data access. The approach identifies and organizes symbol patterns which appear repeatedly. Similar components are represented by a single prototype stored in a library and the location of each component instance is coded along with the residual between it and its prototype. A representation is built which provides a natural information index allowing access to individual components. Compression results are competitive and compressed-domain access is superior to competing methods. Applications to network-related problems have been considered, and show promising results.

1. Introduction

Recent advances in processing, storage, and transmission technology have supported the growth of image databases. The images contained in these databases can, however, strain available resources and in almost all cases some form of compression is required. In recent years, lossy compression, progressive transmission, and source channel coding have provided solutions to the immediate problem of reducing storage and transmission costs. In the document domain, document image databases are of special interest because of the high cost and low quality of automatic document image conversion (including OCR) and the fact that many archival documents can simply not be adequately represented in converted form.

The first document image coding techniques date back to the early 1970s with run-length encoding suggested by Huang [1] and symbol coding suggested by Ascher and Nagy [2]. With the increased use of facsimile machines, the ITU-T (then CCITT) organization finalized the G3 and G4 standards for use in digital transmission soon after [3–5]. The underlying mechanism which achieves compression in these standards is the run-length encoding scheme with some enhancements for vertical redundancy removal. These techniques remain the methods of choice for compressing binary images. Recently, some work has been done on more intelligent coders. The Joint Bi-level Image Experts Group (JBIG) developed an ISO/IEC international standard (also an ITU-T recommendation) [6] JBIG which uses context modeling [7, 8] and at the lowest levels uses a template model and adaptive arithmetic coding to encode predictions. While the standard can encode a hierarchical representation for progressive representation, it is based on resolution reduction and is rarely used.

For documents that will remain and be used in image form, it appears advantageous to consider the characteristics of the image as part of a comprehensive solution rather than simply relying on run-length redundancy. Pattern matching and substitution techniques have recently attracted research since document images contain a rich population of symbols that repeat

consistently. Witten et al. [9], Zhang and Danskin [10] and Howard [11] emphasizes symbol repetition and suggests methods that exploit such redundancy in developing image compression. While some algorithms have used these image characteristics to obtain desirable (lossy and lossless) compression results, their transmission and processing capabilities lag far behind the state of the art in document representation. An appropriate archival mechanism needs to integrate storage, transmission, and processing capabilities.

Similar to the methodology used by Witten et al. [9] and Zhang and Danskin [10], we use a pattern matching and substitution approach [5, 12–17] as was originally suggested by Ascher and Nagy [2]. This method first extracts components from the document image, represents instances of a component by prototypes and creates a prototype library. We define an image generated from only the prototype shapes as a *symbolic image* and the pixel difference between a rendering of the symbolic image and the original image as the *residual*. We code the symbolic portion by specifying the information about the prototype components and their locations within the image. By providing location information, we allow direct access to components and expedite later processing. We code the residual by ordering the residual bits in a way that not only provides access to them on a per component basis but also provides progressive and lossy modes in a rate-distortion sense.

In our representation we are first motivated by the need to create an informational hierarchy such that the most redundant and comprehensible information appears first followed by incremental portions as we step up in the hierarchy. Similar to concepts presented by Nohre [18], where a compression system is designed to be used in image interpretation, our hierarchy organizes information at varying scales. Tasks that can take advantage of this representation include variable lossy compression, progressive transmission, and general document image processing tasks, since each requires access to the document at the component level. Second, Most compression techniques try to minimize storage requirements by removing redundancy and in the case of lossy compression, this often results in removing redundancy along with detail. Competitive methods do not have a usable progressive transmission mode which preserves detail. Third, although some work has been done on the processing of document images in the compressed domain [19–21], the outlook for applying general processing tasks to standard compression methods does not appear promising. Finally, Our method can be used for variable resource allocation in terms of storage, transmission, and processing resources. In limited resource cases, such a representation will aid in all aspects of database operations.

In Section 2 we discuss some background items needed to express our processing and retrieval requirements. In Section 3 we describe our approach to document image processing tasks and in Section 4 we describe our approach to retrieval problems.

2. Previous Work

Before describing the details of our approach, it is useful to briefly review the general problem of image compression as it applies to document images. We then present the detailed methodology of our compression system with some implementation notes.

2.1. Document Characteristics

Document images are scans of documents which are in most cases adequately represented in binary and are rich in textual content. Informally, we can define document images as images that contain components that resemble the symbols of a language. Documents like those shown in Fig. 1 are often scanned at 300 dots/in., tend to be highly structured in terms of layout, and have significant numbers of symbols repeated within the image (i.e., a single or small number of fonts are used).

Since a document can be used in a large number of ways, one 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—something that may not be possible when using resolution reduction compression techniques. Symbol shape and reading order preservation should constitute the basis of any document image representation.

2.2. Symbolic Compression

In the following sections we describe details of our compression algorithm modules for prototype generation and residual coding.

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

Continuum Mechanics and Thermodynamics

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)

Figure 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 [22], cropped (automatically) to the main body of the text.

(*Continued on next page*).

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

(b)

Figure 1. (continued.)

2.2.1. Prototype Generation. Prototype generation is accomplished by segmenting image components and clustering the resulting bitmaps For quality machine print binary document images (i.e., limited numbers of touching or broken characters), connected component analysis does an adequate job of segmentation. Since most digital library images are of this quality, connected components have been successful for segmentation. The analysis returns parameters describing the bounding box, upper left corner and its height and width, for each component.

For clustering, a pattern matching and substitution technique is used. Without fully specifying the matching functions for now, we match each observed component by comparing it to the existing prototypes and classifying the component as either a member of the cluster represented by a prototype or as the seed of a new cluster. Once new members occupy a prototype class, a new prototype is generated. The resulting prototypes represent a clustering of the observed components. Fine-tuning is possible in terms of frequency of prototype calculation, number of prototype calculations, and the threshold used for new prototype seed. Once clustering is complete, another pass through the prototypes removes under-populated clusters.

In our algorithm, we start with a small threshold value for a match using a distance-transform-based match function. After visiting 10% of the input components, we extrapolate a linear function for the expected number of clusters. If we extrapolate to create more than 500 clusters, we stop the prototype generation function and restart it using a higher threshold value. Otherwise, we only recalculate prototype centers (simple average of members) every time 10 new members have been added to the cluster. Furthermore, we calculate the cluster centers only 5 times. Once the clustering is done, we remove underpopulated clusters, determined by membership of less than 5 members, and assign the members to an either close cluster or to a NULL cluster.

Matching functions have been discussed by a number of researchers and have been summarized by Witten et al. [9] and Kia [23]. The most basic matching function is a Hamming distance, where the number of differing pixels measures the dissimilarity between two binary patterns. Most matching functions are derivatives of this method where weights are applied to the distribution of the unequal pixels with respect to some underlying condition. Blob-like distribution of the unequal pixels, complexity of the description of the unequal pixels, and the structural contribution of the unequal pixels with respect to the prototype and component are some conditions that have been considered. The overall consensus is that any matching function that measures some similarity in pattern will work for a wide range document images and that there will always exist ambiguous pairs. An appropriate algorithm should perform gracefully under such conditions.

The effectiveness of pattern extraction and clustering has a cumulative effect on our method, since it relies on connected component analysis and is sensitive to connected and broken characters, a phenomenon which is widely observed in degraded document images. Based on the level of degradation, the prototype images capture the degraded features to form a set of more ambiguous prototypes which in turn will affect the residual information.

Figure 2 shows a set of prototype patterns that belong to a document that has a large number of connected characters. It is obvious that in any document image analysis task a higher level parsing is needed to segment individual characters. Since these prototypes have constituent components spread throughout the rest of the prototypes, it is possible to decompose the bigger components into smaller, more repetitive components. By introducing segmentation cuts at strategic locations, it is possible to create a prototype set which is more indicative of the underlying pattern set [24]. A joint segmentation and clustering algorithm has been used to gather the prototype set shown in Fig. 3. While not perfect, it is more indicative of the underlying character set than the one in Fig. 2. The reduced set of prototype images allows for more effective compressed-domain processing to occur and yields improved compression ratios for degraded images.

2.2.2. Residual Coding. Given a set of prototype shapes, the difference between the prototype shape and the component shape is the residual map and must be represented to provide lossless rendering of the original document image. The ordering of the residual pixels is motivated by a degradation model [25–27], which hypothesizes that pixels close to edges are more likely to be corrupted than the pixels that are in the interior of the character or in the background. Edge effect degradation is the most common form which eventually leads to touching or broken characters. While most degraded images are still readable, it suggests that near-edge effect degradation does not contribute toward readability and may be either removed or be used as a low priority

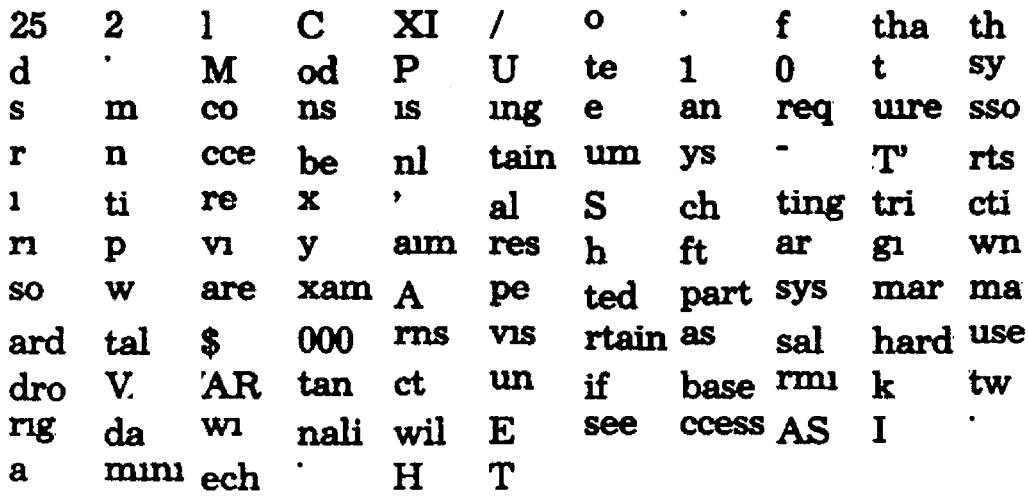


Figure 2. Prototype image set of a degraded document image.

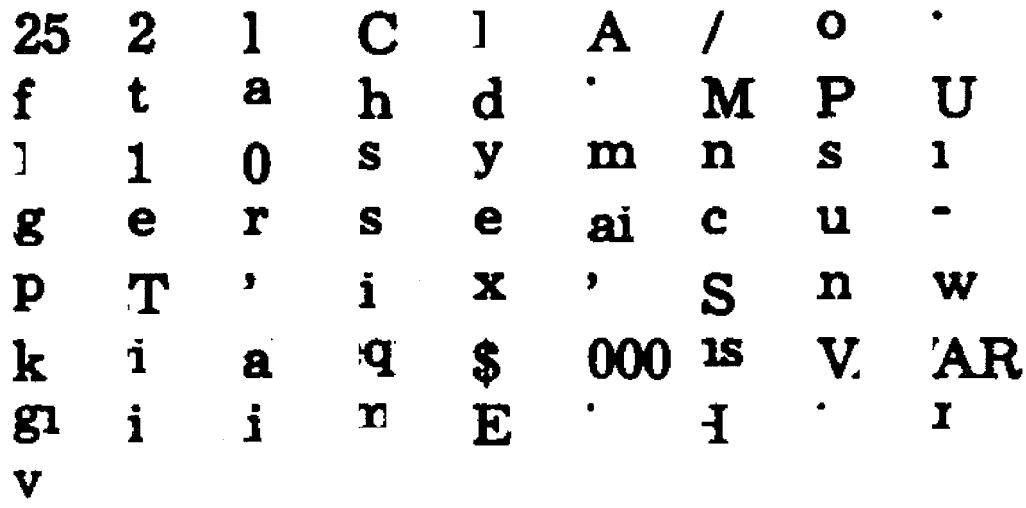


Figure 3. Prototype image set of a degraded document image by performing a joint segmentation-clustering algorithm.

information content. This leaves far-edge effects to be grouped together and be prioritized in an inverse order than the near-edge pixels.

To implement this concept we rely on a distance transform which records distance to the closest edge for each pixel in the image. By ordering the residual pixels by decreasing distance from an edge, pixels that are more likely to be caused by degradation will be presented last, and pixels which are most likely relevant to the structure of the symbol are presented first. It has been shown [23] that grouping of pixels in terms of distance to edge has a desired compressibility result since distribution of pixel values are similar as a function of their distance to edge. Section 4.2 will also discuss the effect of image quality in prioritization-based tasks such as progressive transmission. Together, they provide complete analysis in terms of rate-distortion trade-off.

Figure 4 shows a typical observed component, an assigned prototype, the residual image, and the coded residual stream using row order and distance order. The example portrays a situation where the component and the prototype are of the same recognizable symbol. The distance-ordered residual stream has a deterministic character where the distribution of the pixels varies monotonically. This is due to grouping pixels of the

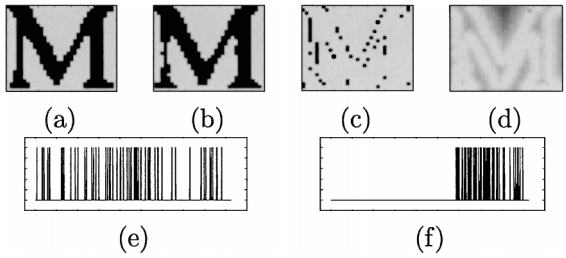


Figure 4. An example of an in-class component and prototype set where (a) is the prototype pattern, (b) is the observed component, (c) is the residual image, and (d) is the distance transform of the prototype. The residual stream is then shown in (e) row order and (f) distance-transformed order.

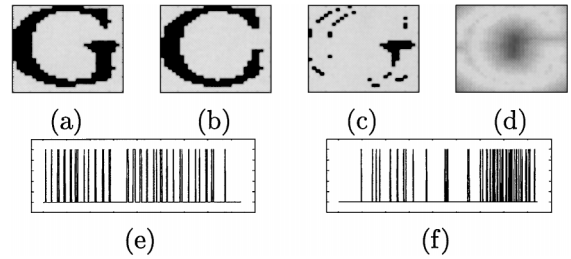


Figure 5. An example of an out-of-class component and prototype set where (a) is the prototype pattern, (b) is the observed component, (c) is the residual image, and (d) is the distance transform of the prototype. The residual stream is then shown in (e) row order and (f) distance-transformed order.

similar distance close to each other. Pixels towards the left of the residual code are far-from-edge pixels with higher probabilities to contain 0's. This shows the basis for higher compressibility. Context-based coding using raster-scan order can also achieve similar results but will not have desired ordering mechanism which can be used in progressive transmission and similar tasks. Figure 5 shows a typical set where the recognizable class differs between the component and the prototype. The distribution of the pixels in this case is not monotonic; however, partial specification of this stream has desirable distortion characteristics, as will be shown in Section 4.2. In addition, the pixels which represent the most significant difference between the two symbols, the lip on the G, will be represented first.

2.2.3. Indexable Representation. Our data representation is summarized in Fig. 6. Intuitively, the required information sources can be grouped into the prototype library, component layout, and residual information sections. Motivated by retrieval problems and the need for document analysis algorithms to access individual components, we propose a top-down hierarchy which starts with prototype image map specification as the top object. The layouts of components forms the next level in the hierarchy below the prototypes. The document image begins to convey the document's content with the use of the layout components and the prototypes. Below the layout information the residual information is provided to supply fine detail. Since the residuals often do not contribute features which facilitate

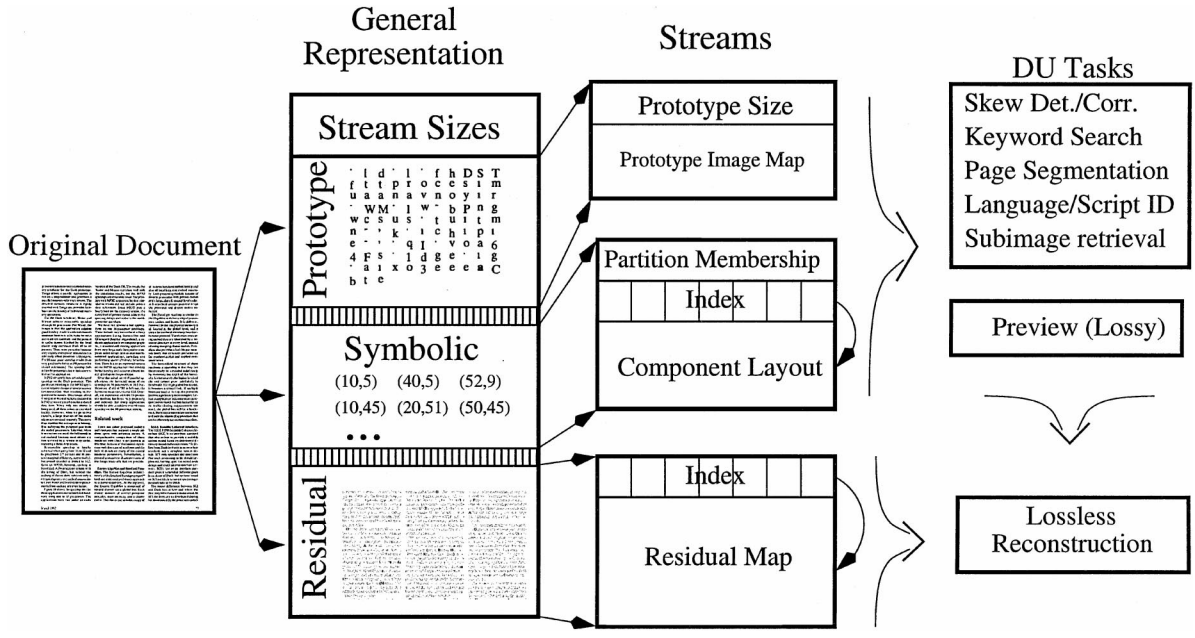


Figure 6. Data representation organization.

correct recognition they are placed at the bottom of the hierarchy.

The level of functionality that is designed into our representation affects storage, transmission, and processing tasks. By creating a prototype library and recording the locations of components within the image, we have in effect provided access to the image components. Tasks such as skew detection and layout analysis can process only the location information while other tasks such as keyword searching and major font detection can be performed using the prototype shape, size, and location. Since on average only 20% of the compressed representation is contained in the prototype shape and location specification, this organization allows access to important information quickly.

Residual coding occupies the rest of the representation, and by recording an index into this section we preserve the component access features. In essence we can extract residual information of a specific component without having to decompress the entire residual section. Furthermore, by ordering the residual pixels in distance-transformed order, we have incorporated desirable lossy compression and progressive transmission capabilities. Motivated by binary degradation models, the intermediate representation is designed using a distortion measure that facilitates recognition of components.

2.3. Implementation

The compression system has been implemented and tested as a set of UNIX executables. The basic routines decompose TIFF images into our representation and recompose our representation into TIFF images. The first step in decomposing an image into our representation is connected component analysis. There are a number of ways to determine the connected components; we use a method based on scanning rows of pixels. The results of this task are variables which specify a bounding box for each connected component. The second step is to cluster these components. We use the pattern matching and substitution method described earlier. In our implementation of the clustering algorithm, we use two null prototypes which are used to refer to small and large graphic components that did not create well populated clusters. We also limit the frequency and total number of prototype recalculations as specified in Section 2.2.1. Given the prototypes we calculate the distance transform by detecting the closest edge to every pixel. We then order the residual pixels in distance-transform order and index them on a per component basis. We populate the appropriate streams of information as mentioned earlier and compress the streams using a static Huffman coding technique. We are able to compress binary images of 2550×3300 pixels in 26 s using a workstation based on the Pentium processor operating at 200 MHz and running the Linux operating system. Roughly 21 s of this processing is attributed to clustering. We averaged a compression ratio of 13 for 122 scanned images and 23 for 20 synthetic images, using the University of Washington database [22]. Table 1 shows a summary of compression results and algorithmic features for our method (SYM), Managing Gigabytes [9] (MG), ITU-T standard Group 4 (G4), ITU-T standard Group 3 (G3), ISO/IEC JBIG standard (JBIG), Lempel-Ziv (LZW), and pack bits (PB). While the symbolic compression

Table 1. Comparison of coding techniques with respect to compression ratios and ability to perform tasks such as progressive retrieval, variable lossy compression, compressed-domain processing, and content-based analysis.

	SYM	G4	G3	JBIG	MG	LZW	PB
Synthetic images	23.2	18.2	12.8	26.9	39	9.5	5.3
Scanned images	12.2	17.8	9.7	22.3	27	8.7	5.5
Progressive retrieval	•			○			
Lossy compression	•			○	○		
Compressed-domain processing	•				○		
Content-based analysis	•				○		

•: Full functionality; ○: partial functionality; SYM: our symbolic compression; G3, G4: ITU-T's standard; JBIG: ISO/IEC's Joint Bi-Level Image Expert's Group standard; MG: method used in Managing Gigabytes; LZW: Lempel-Ziv algorithm; and PB: Packed Bits.

does not have the best compression ratios, it does have the best overall data organization facilitating several important concepts.

3. Document Image Processing

The purpose of our compression system is not only to compress images but also to make them usable. In this section we concentrate on describing processing tasks that exploit our representation scheme, which is in turn motivated by the characteristics of document images.

3.1. Class of Tasks

The tasks that are presented in the following sections are by no means the only possible tasks that can take advantage of our representation. It is obvious that document images contain information within their textual components (samples of rendered characters of an alphabet). Even though most documents contain graphics, images, and other annotated material, they typically contain primarily text. The class of problems which address these textual components can exploit our representation. These tasks, in general, access components, symbol shapes, and their layout to determine their intended target. By pre-processing these features, we can typically reduce their processing time.

3.2. Skew Estimation and Correction

Skew estimation and correction are important tasks for most OCR systems. By using an algorithm based on the Hough transform, we are able to estimate skew and correct it using our representation, without fully decompressing. We use only the position and size of each component as shown in Fig. 7. We decompress the prototype size, block membership, and component layout streams and input the coordinates of the middle of the bottom of each component to a Hough transform to compute skew. For the 122 images in our database, it takes an average of 1.5 s and 9152 bytes/image to calculate skew to an accuracy of 1/640 vertical units. On the same set of test documents the average error was measured to be 0.1786° by taking the average difference between the measured and ground-truth skew angle (provided in the database). For skew correction, we ignore component level skew and modify the component layout and residual map streams to move components to their unskewed locations. We also reorder components if they move from one block to another. Figure 8 shows a sample of a deskewed image. Note

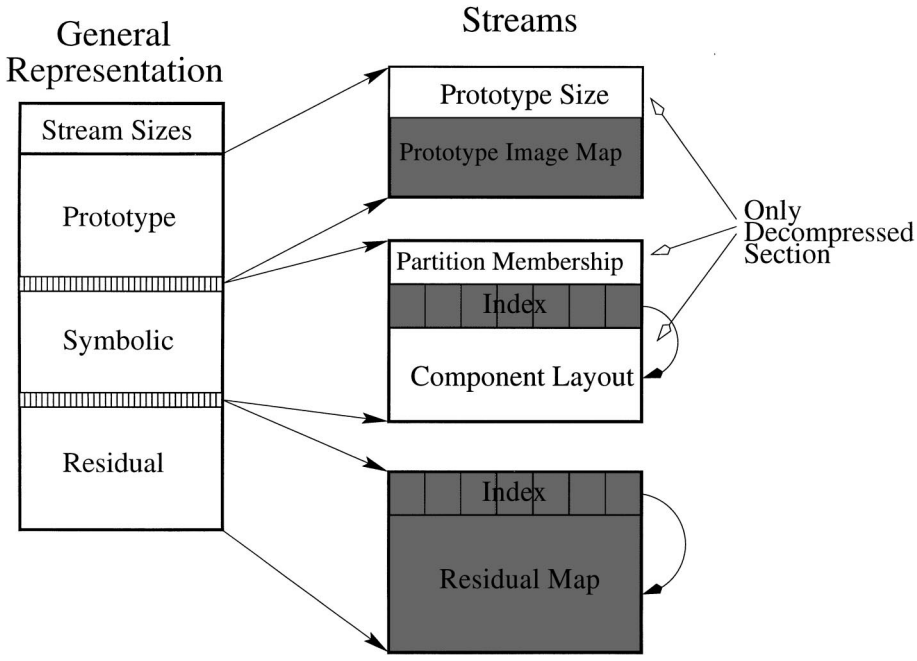
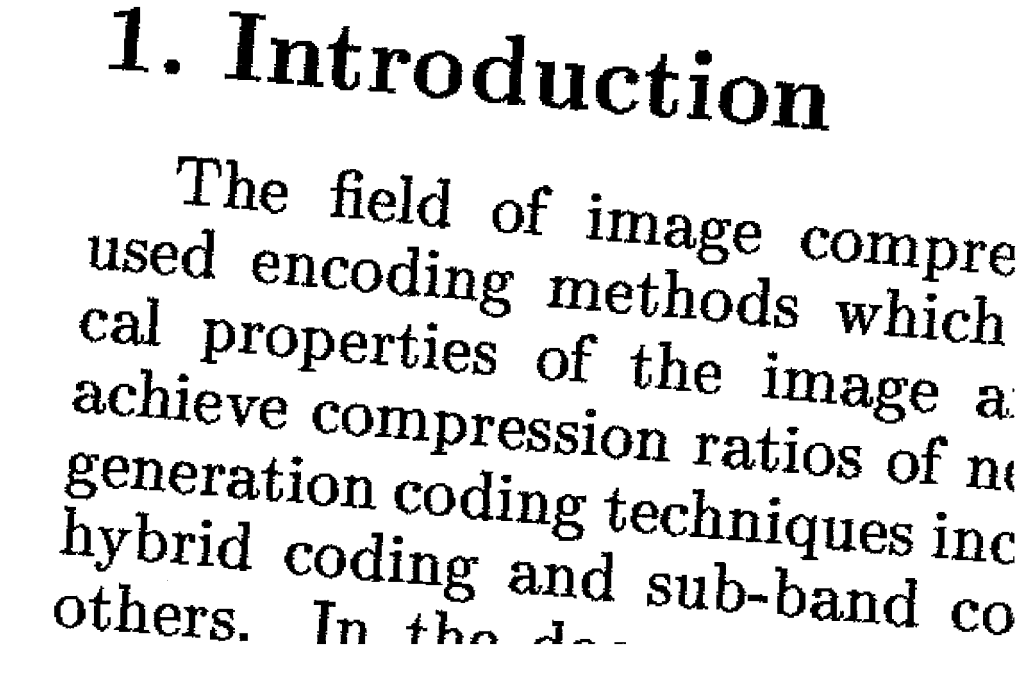
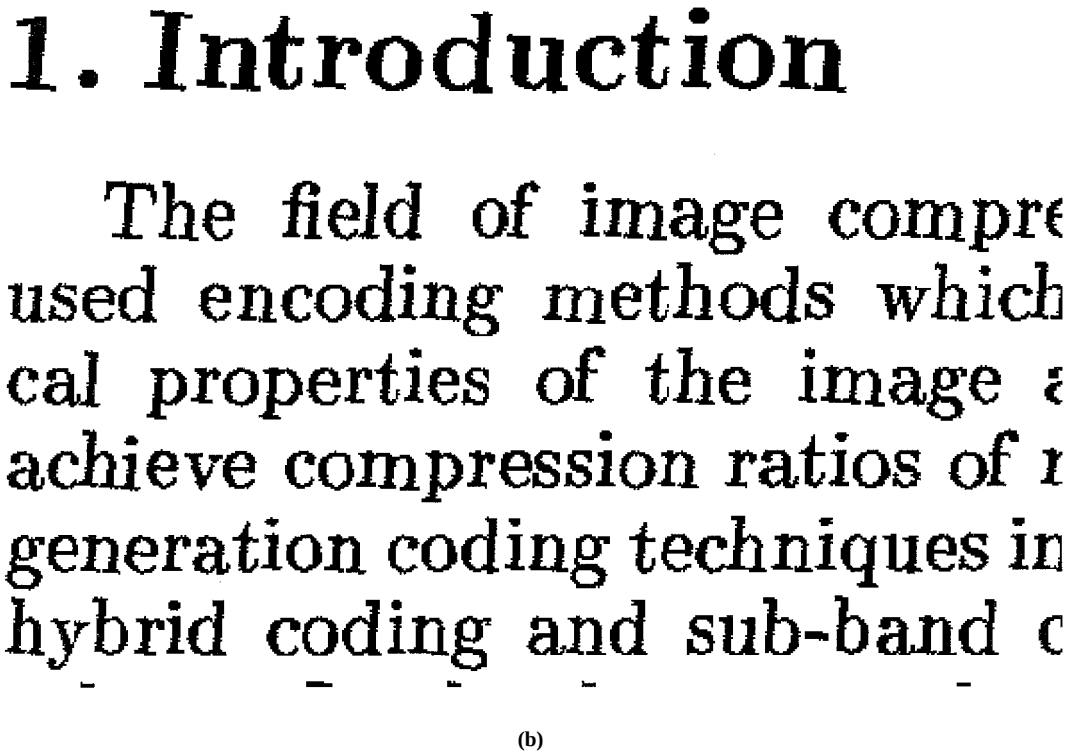


Figure 7. Portion of the code that requires decompression and processing.



(a)



(b)

Figure 8. Sample portion of an image (a) before and (b) after deskewed by only modifying prototype location.

that local skew is still present in the individual characters. For the 122 test images, it took an average of 1.88 s to deskew an image.

3.3. Keyword Search

Because of symbol and prototype access to the coded image, it is also possible to perform keyword searching in the compressed domain. A method, based on Spitz's work on shape coding [28], 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 as shown in Fig. 9. 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 in each line. This method picks 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. The result is a scan-line ordered code which represents an image component by several shapes. The shapes that we used in our experiments were ascenders, descenders, xheight, circular (existence of a whole in the component), and concave from right. The mapping from image domain to feature domain is relatively efficient since it only operates on the prototype set and not on every image component. The input ASCII query is then translated into shape codes and compared to the scan-line ordered shape codes as shown in Fig. 9; 90% of matching shapes are taken to be a match.

The query matching is scale independent due to the normalization effect of shape coding. Determination of the shapes mentioned above are solely dependent on determination of scan-lines and main body text. While these methods are prone to error in degraded text, it benefits from good component segmentation and clustering. Any improvement in these basis functions will improve accuracy and robustness of the image query.

Figure 10 shows some search results on the 122 database images for the query "approach". It took an

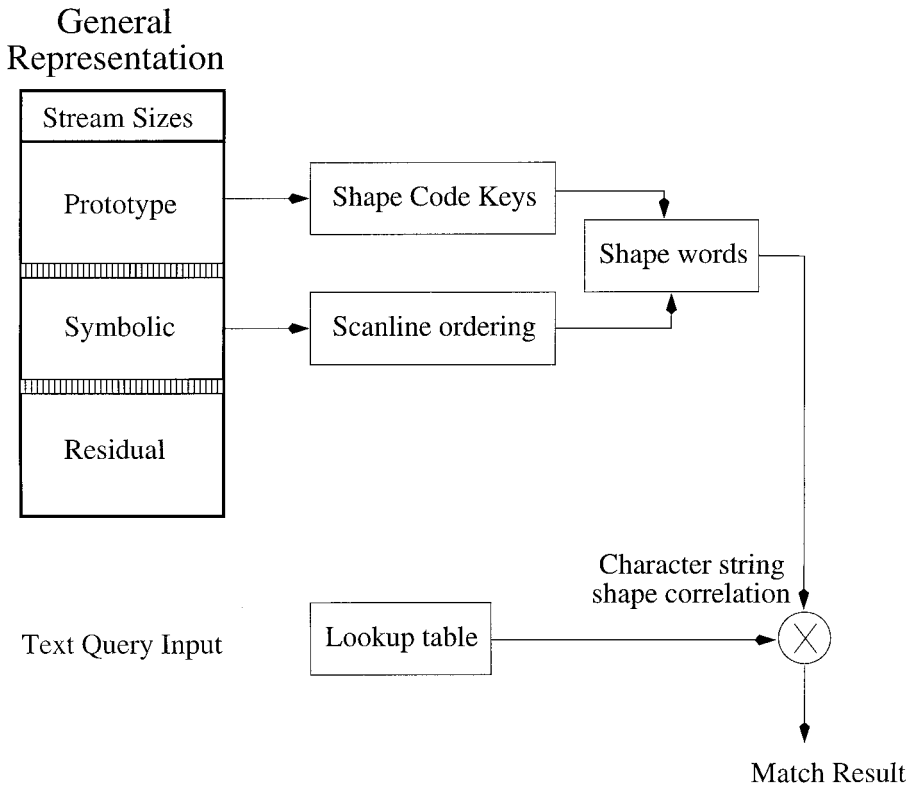


Figure 9. Accessing symbol shapes and components in the compressed domain and matching based on an input query.

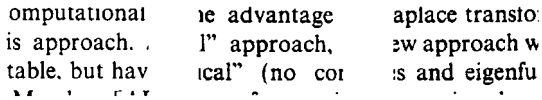


Figure 10. Query results for the string "approach" on selected images from the University of Washington Database [22].

average of 1.6 s/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.

3.4. Other Tasks

Some tasks that have been identified but are not yet implemented are major font detection, layout analysis, language identification, and document classification. Major font detection can prove useful to OCR engines and in similar tasks. This can be done by scanning the prototype library shapes and (if needed) some of the actual components (requiring residual recombination). Layout analysis is very generic and can have implications for a large number of tasks. Skew estimation and correction is such a related task. For such tasks, the layout of the document, the components, and their organization lies entirely in the component layout specification. This is a fundamental part of our representation and will expedite any layout analysis task. Some other examples of these tasks are column detection, reading order determination, and zone classification. Language identification is desirable for a number of reasons which will not be discussed here. It can be done by analyzing the prototype shapes and their locations. The preprocessing of these portions of the data can expedite the processing task. Document classification is useful for some database problems. Location of components within an image can provide a rich source of information on how to classify the document (journal article, memo, newspaper, ...).

4. Retrieval

Using our coding method we can demonstrate a scalable lossy representation and by the induced hierarchy we are able to show a desirable progressive transmission mechanism. We may also assign priority to the

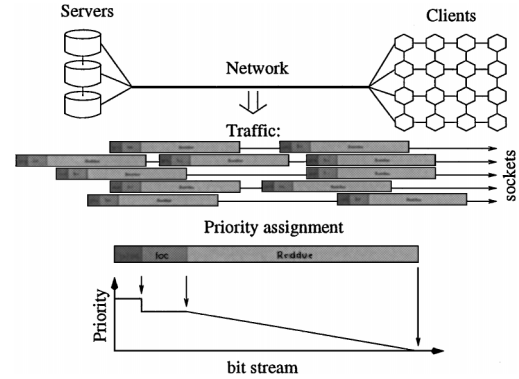


Figure 11. Network utilization for server-client transmissions.

hierarchy as shown in Fig. 11 for network traffic, a variable measure of error correction in channel coding, and a mechanism to achieve rate control. While rate control can be accomplished by manipulating the prototype maps and component layout with their representation resolution, more freedom is available by the additional coding of the residue due to its size and granularity.

4.1. Class of Retrieval Problems

Similar to the class of applicable processing tasks, there exists a class of retrieval problems, such as content transmission and delivery, that can exploit our representation. The basis of our argument for retrieval is the same as before; symbol access expedites retrieval, and furthermore the implied hierarchy of prototypes over residuals and the hierarchy of the distance-ordered residuals enhances retrieval. The underlying emphasis in retrieval problems is the organization of useful data by relative importance. We have definitely achieved this in our representation where not only was compression achieved at various levels but the importance of the information was also ordered.

4.2. 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 block-wise 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.

Lossy compression can be easily achieved by terminating a progressive transmission code. The quality of image representation needs to be addressed in terms of an appropriate rate-distortion relationship. It is hard to define a suitable distortion measure; the traditional signal-to-noise ratio and its derivatives are not acceptable. This is because a constant power level might represent images that have a varying amount of image quality, in terms of readability, recognizability, or other document-related measures. An appropriate distortion function should reflect the recognition rate of these lossy representations. Using OCR engine and an OCR evaluation software, we are able to associate recognition rate with entropy and get a rate-distortion relationship.

We are able to step through rates by recording a fractional amount of the residual information, since the majority of the data (about 80%) is in the residual section. We compare the distortion between a number of residual ordering mechanisms. Figure 12 shows the trade off between forward and backward row and distance ordering. The dashed line is an estimate of the rate-distortion trade-off if a predictive structural ordering were used [23] rather than distance ordering. An important comment is needed to clarify the rate aspect of these trade-offs. The value of rate which is used in Fig. 12 is the first-order binary entropy value. If the binary code does not have memory (correlation across binary samples) the first-order entropy will be the true entropy rate. While the row-ordered codes do not have strong memory content, the distance order does. Therefore, the rate calculations for the distance-ordered code should be lower than those shown in Fig. 12.

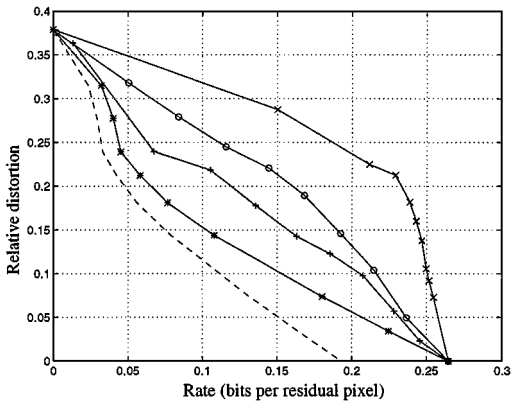


Figure 12. Rate-distortion trade-off of residual coding image by performing a joint segmentation-clustering algorithm. $*$ is forward distance ordering and $\times$ is reverse distance ordering. $+$ is forward row ordering and o is reverse row ordering. Dashed line is an expected predictive ordering.

4.3. Subdocument 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 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.

5. Conclusion

We have presented a document image representation which achieves compression along with access to components to allow for compressed-domain processing. In achieving this representation we attempted to improve retrieval of document images. It was found that useful information, in the case of documents, is in the form of components and that better component representation is the key to better retrieval. The ultimate image representation would integrate an ASCII representation within the coded components. Residual coding is still necessary to allow lossy representation to varying degrees.

We have demonstrated the effectiveness of our representation for compressed-domain processing. Tasks such as skew estimation, skew correction, and keyword searching are easily implemented. There is also sufficient evidence to support the claim that other document

image analysis tasks such as major font detection, layout analysis, and language identification can use our representation to perform their operations in the compressed domain. This is because most existing algorithms depend on symbol shape and location, which is sufficiently captured by our representation. It is very seldom that residual information is needed for these tasks, but even in such cases, our residual code supports indexing based on image components and is ordered hierarchically.

Hierarchical representation forms the basis of applying our approach to various transmission problems. An implied hierarchy first orders the prototype symbols, then their locations, and finally their residual information. This hierarchy is not sufficient since the residue constitutes the majority of the code. A distance-based hierarchy is imposed on the residue which not only improves overall residual compression but also contributes to information transmission. A simple ordering of residues contributed to lossy compression and progressive transmission of the code. The remaining task is to define an appropriate cost function so that a network broker can bid for an appropriate transmission channel.

Acknowledgments

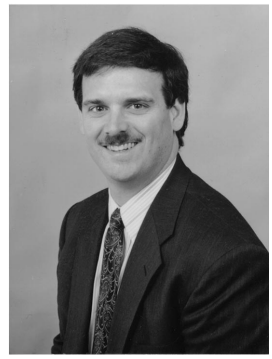
We would like to thank Professor Azriel Rosenfeld for an unending supply of ideas, comments, and improvements in regard to this project. It has been an honor and a privilege to have worked with him. The support provided for this research by the Advanced Research Projects Agency under contract MDA 9049-6C-1250, is gratefully acknowledged.

References

1. T. Huang, "Run length coding and its extensions," in *Picture Bandwidth Compression*, T. Huang and O. Tretiak (Eds.), Gordon and Breach, pp. 231–264, 1972.
2. R. Ascher and G. Nagy, "A means for achieving a high degree of compaction on scan-digitized printed text," *IEEE Transactions on Computers*, Vol. 23, pp. 1174–1179, 1974.
3. D. Bodson, S. Urban, A. Deutermann, and C. Clarke, "Measurement of data compression in advanced group 4 facsimile system," *Proceedings of the IEEE*, Vol. 73, pp. 731–739, 1985.
4. M. Holt and C. Xydeas, "Recent developments in image data compression for digital facsimile," *ICL Technical Journal*, pp. 123–146, 1986.
5. W. Pratt, P. Capitant, W. Chen, E. Hamilton, and R. Willis, "Combining symbol matching facsimile data compression system," *Proceedings of the IEEE*, Vol. 68, pp. 786–796, 1980.
6. JBIG, Progressive bi-level image compression, 1993.
7. W. Pennebaker and J. Mitchell, "Probability estimation for the Q-Coder," *IBM Journal of Research and Development*, Vol. 32, pp. 737–752, 1988.
8. 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*, Vol. 32, pp. 717–726, 1988.
9. I. Witten, A. Moffat, and T. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images*, Van Nostrand Reinhold, 1994.
10. Q. Zhang and J. Danskin, "Entropy-based pattern matching for document image compression," *International Conference on Image Processing*, pp. 221–224, 1996.
11. P. Howard, "Lossless and lossy compression of text images by soft pattern matching," *Proceedings of the IEEE Data Compression Conference*, pp. 210–219, 1996.
12. 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.
13. O. Johnsen, J. Segen, and G. Cash, "Coding of two-level pictures by pattern matching and substitution," *Bell System Technical Journal*, Vol. 62, pp. 2513–2545, 1983.
14. 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.
15. K. Mohiuddin, "Pattern matching with application to binary image compression," Ph.D. thesis, Stanford University, 1982.
16. K. Mohiuddin, J. Rissanen, and R. Arps, "Lossless binary image compression based on pattern matching," *Proceedings of the International Conference on Computers, Systems, and Signal Processing*, pp. 447–451, 1984.
17. K. Wong, R. Casey, and F. Wahl, "Document analysis system," *IBM Journal of Research and Development*, Vol. 26, pp. 647–656, 1982.
18. R. Nohre, "Computer vision: Compress to comprehend," *Pattern Recognition Letters*, Vol. 16, pp. 711–717, 1995.
19. C. Maa, "Identifying the existence of bar codes in compressed images," *CVGIP: Graphical Models and Image Processing*, Vol. 56, pp. 352–356, 1994.
20. A.L. Spitz, "Skew determination in CCITT Group 4 compressed document images," *Proceedings of the First Symposium on Document Analysis and Information Retrieval*, pp. 11–25, 1992.
21. A.L. Spitz, "Logotype detection in compressed images using alignment signatures," *Proceedings of the Fifth Symposium on Document Analysis and Information Retrieval*, pp. 303–310, 1996.
22. R. Haralick, "UW English document image database I: A database of document images for OCR research," CDROM.
23. O. Kia, "Document image compression and analysis," Ph.D. thesis, University of Maryland, College Park, 1997.
24. O. Kia and D. Doermann, "Integrated segmentation and clustering for enhanced compression of document images," *International Conference on Document Analysis and Recognition*, Vol. 1, 1997.
25. T. Kanungo, R.M. Haralick, and I.T. Phillips, "Global and local document degradation models," *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 730–734, 1993.

26. G. Nagy, P. Sarkar, D. Lopresti, and J. Zhou, "Spatial sampling of printed patterns," Draft, 1996.
27. P. Sarkar, "Random phase spatial sampling effects in digitized patterns," Master's thesis, Rensselaer Polytechnic Institute, 1994.
28. A.L. Spitz, "An OCR based on character shape codes and lexical information," *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 723–728, 1995.

Omid E. Kia received the Bachelor's degree in Electrical Engineering from the Catholic University of America, Washington DC, 1987. He received the M.Sc. degree in Electrical Engineering from the University of Illinois at Chicago in 1989. He received his Ph.D. degree in Electrical Engineering at the University of Maryland at College Park in 1997 with a thesis entitled "Document Image Compression and Analysis." He worked in industry during his pursuit of the Ph.D. degree and worked in various areas such as estimation/detection, pattern recognition, and statistical analysis for radar and sonar related projects. He has been affiliated with the Center for Automation Research and the Language and Media Processing Laboratory of the University of Maryland since 1994 and is now with the National Institute of Standards and Technology. His research interests are in areas relating to compression. His current interests are in document image compression and processing, large scale image browsing, video image database indexing and processing, and multimedia databases.



David S. Doermann received the B.Sc. degree in Computer Science and Mathematics from Bloomsburg University in 1987. He earned the M.Sc. degree in 1989 in Computer Science at the University of Maryland. He continued his studies in the Computer Vision Laboratory where he earned the Ph.D. in 1993 with a thesis entitled "Document Image Understanding: Integrating Recovery and Interpretation." He is currently Co-Director of the Laboratory for Language and Media Processing in the University of Maryland's Institute for Advanced Computer Studies. His research centers widely around the topics of document image analysis and multimedia information processing. Recent intelligent document image analysis projects include page decomposition and structural analysis, page segmentation, logo recognition, document image compression, duplicate document image detection, image-based retrieval, character recognition, generation of synthetic OCR data, signature verification. In video processing, projects have centered around the segmentation of compressed domain video sequences, structural representation and classification of video and the performance evaluation of automated video analysis algorithms. Dr. Doermann has served on numerous board and program committees and is an Editor of the newly formed International Journal on Document Analysis and Recognition.