INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI®

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Université d'Ottawa • University of Ottawa

PATTERN COMPRESSION

By Pascal Blais

Thesis submitted to the School of Graduate Studies and Research of the University of Ottawa in Partial Fulfillment of the Requirements for the Degree of Masters in Computer Science*

School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario

* The Masters program in Computer Science is a joint program with Carleton University, administered by the Ottawa-Carleton Institute for Computer Science

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

0-612-38737-2

Canada

Acknowledgements

I would like to thank my supervisor, Dr. Robert Laganière for his advices and support. I would like also to thank Marcin Kolbuszewski, who was working at the National Research Council, for he is the initiator of this project. Finally, I would also like to extend my gratitude toward the Natural Sciences and Engineering Research Council of Canada for their support.

Table of Content

Table of Figures

Chapter 1. Introduction

The Canadian Institute for Scientific and Technical Information (CISTI is a service of the National Research Council (NRC)) is one of the world's major sources for information in all areas of science, technology, engineering and medicine. It has a huge library of scientific publications. The library receives everyday orders from public and private organizations, for copies of scientific articles. Since most of the publication is on printed document, the client will receive, depending on his preference, either a photocopy by mail, an electronic copy by fax machine or over the internet. This is where this project was initiated. Since it becomes more popular and more efficient to communicate information electronically, electronic network like the internet, becomes more and more loaded, and more electronic devices are needed to store all this information. So it is a good idea to look at better ways to efficiently handle electronic information. This project's goal was to improve one specific aspect of this problem: the compression of document imagery.

Because of the wide use of personal computers, documents are now mostly created electronically. Unfortunately it is not always possible to have access to the electronic version of a document; it is either not available or it does not exist. A simple way of having an electronic version of a printed document is to scan it: to "scan" a document is to take an electronic picture of it. The electronic picture can be electronically stored, communicated over networks and reprinted on paper or a computer screen. This kind of electronic picture made from printed document is called document imagery.

A problem with document imagery is the huge electronic size of such document. This is due to the fact that in order for the picture to be of acceptable quality, the resolution of the scan must be high. Fortunately, there are ways to transform a file (an electronic document) into a new file such that the later has a smaller size. It then becomes more efficient to store or communicate this compressed file. Of course the original file can be recreated from the compressed version. Various compression methods exist but only a few have been designed for document imagery. The best compression schemes use the properties of scanned documents.

The goal of this project was to study known compression methods and to design a new one that take better account of the properties of document imagery and would preferably be more efficient than the other known methods.

This document is a final report on the project. Standard and well-known compression algorithms will be presented as well as a new method, called Pattern Compression. This method takes advantage of two things: the kind of information a document imagery is composed of and that some data may be lost in the process as long as it remains visually imperceptible. It is a variation of vector quantization. The input document is broken down into numerous small 2 dimensional patterns, which correspond typically to the characters of a text. A codebook is created from these patterns. Patterns that are very similar are considered to be the same. A document is compressed by representing it with a sequence of codebook representatives. Experiments presented in this document, show that Pattern Compression generally compress around twice as much as the CCITT group 4 compression standard [3.2]. Other advantages of the method are that by altering the value of the algorithm parameters, a user can control the quality of the compression. The decompression process is very fast and the technique can also be applied to color images. Unfortunately the method have some problems: the compression process is rather long, it has a problem braking down big patterns and its compression ratio is not as good as another compression method called Cartesian Perceptual Compression [6]. Future work and ideas on how to improve Pattern Compression are suggested in the Conclusion.

This document can be viewed into two main sections; the first section only describes different aspects of the project, mainly the methods studied and used for compression. That section is represented by chapter 2 to 5. Chapter 2 introduces the concept of compression. Chapter 3 describes the specific type of document imagery used in this project. Chapter 4 describes various methods and other processes used for compression. That chapter describes only well known methods. Chapter 5 takes a different approach on the problem: vector compression. The new method Pattern Compression is described in details.

The second main section of this document is chapter 6: it contains results of experiments made in order to compare different methods and to gather statistics. The first experiment was simply to compress a common typical document with all the different methods and compare the results. The second experiment was to take the best methods and compress two groups of documents; the first group made of 1-page documents only and the second one made of 4-pages documents. These experiments show the various range of compression possible with the best methods and also show that some methods such as Pattern Compression, increase their effectiveness when compressing multi-pages documents.

The documents used in this project were stored using the TIFF file format [1]. Its great versatility makes it the perfect choice. Annexe 1 contains an overview of the specification of that standard file format.

In order to study and conduct experiments with the different compression methods, most of them had to be implemented. Almost every method described in chapter 4, the CCITT compression standards [3], predictive coding [2] and the morphological transformations [4] (except for the trimming operation), have been implemented by myself based on the respective description given by each reference. The morphological transformation trimming has been designed by me. I also completely designed and implemented the new method Pattern Compression. Cartesian Perceptual Compression [6] is a patented product

and for this project, the software "CPC Tool" was used to compress document with the CPC algorithm; this software is available from the Cartesian Company's web site [6.1]. Methods have been implemented with the programming language Java; it is a high level language and the implementation of all these schemes was fast and easy. Unfortunately, the use of Java proved to be less advantageous when running experiment with Pattern Compression (see 5.4.6 Advantages and Disadvantages of Pattern Compression).

Chapter 2. The Concept of Compression

The goal of compression is to reduce the time of communication and the storage space of information. The general concept of compression is that a data stream (d) can be mapped to an alternate data stream (d') using a mapping function (F) such that the resulting data stream is smaller (as small as possible) and that it can be reverted back to its original form using the inverse mapping function (F'). The effectiveness of compression is based on how predictable are the elements of a data stream: the more predictable are the elements (the more they are ordered, repeated etc...), the more compressed will be the resulting data stream. The inverse is also true: the more chaotic are the elements, the harder it is to compress.

Compression algorithms can be separated in two classes: loss-less and lossy. When compressed data streams can always be reverted back to their original form, the compression algorithm is called loss-less. When compressed data streams may not be reverted back to their original, the compression algorithm is called lossy. As it will be explained, it is not always necessary to be able to revert compressed data to its original form. By being less constraint, lossy algorithms usually have higher compression rate than loss-less.

Data stream usually have a type which is subject to different interpretation, for example: if the data stream is a text file, then the elements of that stream (each group of 8 bits) represent an ASCII character. If the data stream is an image, then elements represent

pixels. If the data stream is a sound, then the elements represent sound waves, etc... Groups of elements of a stream represent information. This information is interpreted by who ever can read it. In some forms, such as image and sound, it is possible to slightly change the data stream such that the interpreted information remain the same, for example:

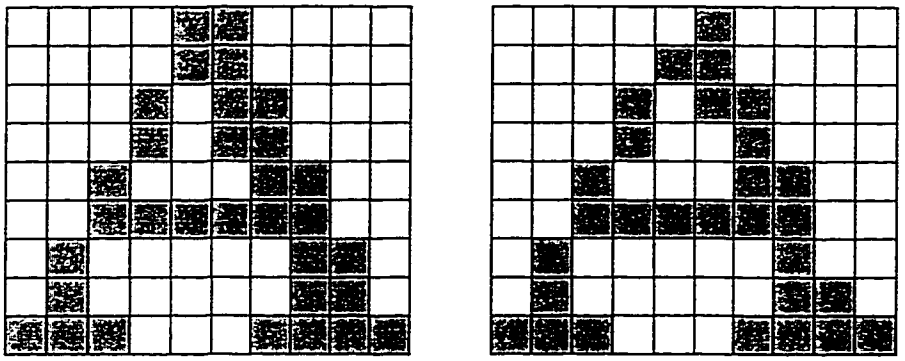


Figure 1. Although these two images are identical except for three pixels, it is still interpreted by us (human) as the same information; the letter "A".

The ability to take some loss without loosing information improves the effectiveness of compression. Lossy algorithms usually use generalization and approximation methods for compression. Generalization is a process where similar data are grouped and referenced as the same. Approximation is the process of mapping data to similar predefined data. The more generalization and approximation is applied, the more effective is the compression but the more data get lost as well.

The first two groups of compression method defined in the next chapter, the CCITT compression standards [3] and the Predictive Coding [2] are non-lossy algorithms. The morphological transformations [4], CPC [6] and the new method Pattern Compression are lossy algorithms. The morphological transformations are being used to try to rid a document of "noisy" data. CPC and Pattern Compression try to manipulate and generalize elements of a document. They allow the possibility of losing data in order to get a better compression ratio but at the same time, diligently enforcing a maximum

distortion level so that the original symbols on the input image are still interpreted as the same information after the compression process (see Chapter 5. Vector Compression).

Chapter 3. File Format

This section defines the kind of document imagery that has been used through out this project.

The type of data that we try to compress here is black and white faxed or scanned document. A document can have one or more pages. Each page is a 2-dimensional array of pixels. A pixel can only be one of two colors: white or black. Thus a pixel can be represented by only one bit. Conventionally the background color is represented by the bit 0 and the foreground color by the bit 1 (usually white is the background and black the foreground). Scanned documents have various sizes depending on the paper size, the region scanned and the resolution of the scan. Resolution usually varies from 100 DPI (Dot Per Inch) to 1200 DPI. In this report, all the tests have been done on 300 DPI scanned documents. Each document have one or more pages and each of them are 3300 pixels in length by 2544 pixels in width; each pages have a size of approximately 1 Megabyte. In our work, documents are stored using the TIFF file format [1]; it is described in Annexe A. Almost every documents used in the various experiments of this project are scanned scientific articles. The following figure is a very good example of a page of one such document; actually, this page has been used in this research to conduct common tests with various algorithms.

JPN. J. APPL. PHYS. Vol. 18 (1979), No. 11

A Nondestructive Method for Measuring the Spatial Distribution of Minority Carrier Lifetime in Silicon Wafer

Yoichi MADA

Musashino Electrical Communication Laboratory, Nippon Telegraph and Telephone Public Corporation, Musashino-shi, Tokyo 180

(Received May 24, 1979)

The electrical behaviour of silicon devices is generally affected by lattice defects present in the base material. The minority carrier lifetime is an appropriate measure of the degree of perfection of silicon crystal, because this parameter is more sensitive to crystal quality than other parameters such as resistivity or mobility. As the lifetime will change during polishing, annealing, or device-fabrication processes, a method for measuring silicon wafer lifetime will be useful. To measure wafer lifetime, the method must be free from ohmic contacts so as to avoid lifetime change. Further, it must have high spatial resolution to check electrical inhomogeneities in the wafer. No method satisfying the above requirements has been developed up to now.

A technique using microwave absorption[1] seems most promising, because no ohmic contacts are needed. In this technique, the change in photoconductivity after injection of minority carriers by a light pulse is detected by the reflected microwave. Two methods have been proposed for coupling the microwave to the charge carriers in the sample. One[2] is where the sample is placed inside a waveguide and electrical coupling is obtained through the fundamental mode of the waveguide. In this method, the sample size is limited by the dimension of the waveguide used. In the other method,[3] electrical coupling is obtained through a horn antenna. This method overcomes the inconvenience of size limitation, but great care must be taken with the signal on account of the open nature of the arrangement. The above two methods cannot determine the spatial distribution of the lifetime in the sample.

To measure the minority carrier lifetime of a small part of a wafer, two coupling methods can be considered. The first is to excite minority carriers locally, using a focused spot of pulsed light. The second is to couple the microwave locally with minority carriers. The second method is more accurate, because the signal detecting area used in this second method does not change during measurement. In this work, a nondestructive method will be shown that can measure the spatial distribution of the minority carrier lifetime in silicon wafer using an S-band microwave.

The experimental arrangement for measurement of the lifetime is shown in Fig. 1. The microwave oscillator provides 2–4 GHz (S-band) microwave output at one milliwatt. The electromagnetic waves travel through a variable attenuator and circulator, and then reach the antenna. Electrical coupling between the electromagnetic waves and the charge carriers in silicon is brought about by this antenna. For injection of minority carriers, a xenon flash lamp that provides a light pulse of 2 μs in duration is used. The light is collected by lenses to a spot size of 3 mm on the sample. The microwave reflected through the circulator is measured by a crystal detector after amplification by a Watkins Johnson 5004 amplifier.

The antenna structure that produces effective coupling between electromagnetic waves and charge carriers in the sample is shown in Fig. 2. It is made of two fine copper wires of diameter 0.1 mm. Their input ends are connected directly to the coaxial cable propagating the microwave. Their output ends, which radiate electromagnetic waves towards the sample, are fixed on a platform in a straight line with distance D between each other. The platform is made of polystyrene to reduce microwave electrical loss. The distance D has a great effect on the signal gain. The optimum distance for maximum

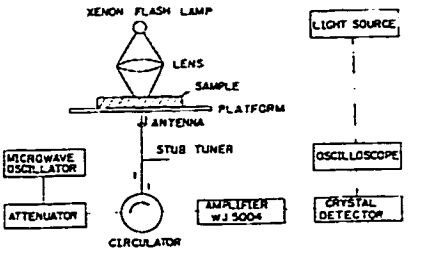


Fig. 1. Lifetime measurement apparatus block-diagram.

Chapter 4. Known Methods used for Compression

This chapter first defines two methods considered as industry standard for compressing document imagery: the CCITT compression standards [3]. It then presents a process called Predictive Coding [2] and some Morphological Transformations [4]: these processes are used in this project in conjunction with other compressing method in order to increase compression ratio. Finally, a method called Cartesian Perceptual Compression is presented.

4.1 CCITT Compression Standards

The CCITT (the Consultative Committee on International Telegraphy and Telephony) develop specifications for facsimile apparatus and communication. Namely they have defined two coding schemes for the transmission of document imagery; both schemes encode the source image on a horizontal scanline-by-scanline basis, corresponding to the way in which documents are scanned and printed on a facsimile machine. The difference lies in the way the two standards handle successive scanlines: in the first one, each scanline is encoded independently, whereas in the second scheme, scanlines are encoded with reference to the previous one, resulting in improved compression ratios. These two methods can be considered as standard since their simplicity and efficiency make them very popular.

4.1.1 1-Dimensional Compression

This compression scheme is also known as the Group 3 compression [3.1]. In this method, a scanline is encoded as a set of runs, each representing a number of consecutive white or black pixels. Every run is encoded using a predefined unique code word. These code words have been computed by a statistical study on typical document imagery such that frequently occurring lengths of run may be encoded very efficiently, at the expense of the infrequent ones. For example, a black run of 2 or 3 pixels is encoded using just 2 bits, whereas one of 1000 pixels is encoded with 25 bits.

This is an example of a scanline encoded using this scheme:

Runs:	7 Whites	3 Blacks	5 Whites	1 Black	4 Whites
Corresponding Code:	1111	10	1100	010	1011

Figure 3. 1-Dimensional Compression Example

As you can see, this 20 pixels line has been coded with 17 bits. Compression ratios achieved using Group 3 depend on the nature of the source image. Under certain circumstances, it will be obvious that the encoded version could be larger than the original (consider the case of single alternating black and white pixels), but in general compression ratios of 10:1 can be achieved on pages of typewritten text.

4.1.2 2-Dimensional Compression

This standard, known as CCITT Recommendation T.6 [3.2], is designed for error-free digital facsimile transmission mainly on public data networks. This specification has been

widely used in document imaging systems for compressing black and white imagery. It is also referred as the CCITT Group 4 compression standard.

This scheme is known as the Modified Modified Relative element address designate code (MMR): it uses a 2-dimensional line-by-line coding method. The position of each changing pixel (called element) on the current coding line is coded with the respect to the position of a corresponding reference element located on either the coding line or the reference line. The reference line is immediately above the coding line. After the coding line has been coded, it becomes the reference line for the next coding line. The reference for the first line of a page is an imaginary white line.

The coding is based on 5 changing picture elements:

- a0 : The reference element on the coding line.
- a1 : The next changing element to the right of a0 on the coding line.
- a2 : The next changing element to the right of a1 on the coding line.
- b1 : The next changing element on the reference line to the right of a0 and of opposite color of a0.
- b2 : The next changing element to the right of b1 on the reference line.

Example:

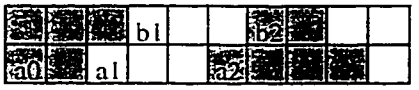


Figure 4. Picture Elements

They are three coding modes: Pass Mode, Vertical Mode and Horizontal Mode. Depending on the relative position of the changing element along the coding line, one of the three coding modes will be used.

Pass Mode:

This mode is used when the position of b2 lies to the left of a1; a Pass code word means that the run of pixels defined by b1 to b2 in the reference line does not repeat in the coding line.

Vertical Mode:

When the relative distance between a1 and b1 is less than or equal to 3 this mode is used. This is used to code runs of pixels that start or ends almost on top of the other.

Horizontal Mode:

When neither pass mode or vertical mode occur then horizontal mode is used; this mode is simply the standard 1-dimensional compression method mentioned earlier where run of pixels are encoded with predefined code words.

Example:

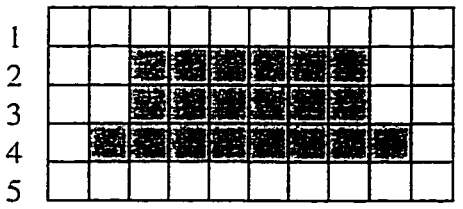


Figure 5. 2-Dimensional Compression Example

- Line 1 is coded vertically (if the previous line is the imaginary white line): since this line is made solely of white pixels, the a1 element will be placed at the end of the line. If the previous line were also a white line then the b1 element would also be placed at the end of the line, exactly on top of a1.

- Line 2 is mostly coded horizontally: this line will be coded into three parts: the first two runs (a run of 2 white pixels and a run of 6 black pixels) are coded horizontally. The last run of white pixel will be coded vertically. Since the previous line is a complete white line, the b1 element will always be situated at the end of the line. For the first two runs, the a1 element is too far from the b1 element and to its left, thus vertical or pass mode are not possible. But at the last run, the a1 element is within three pixels distance of b1, so that part is coded vertically.
- Line 3 is coded vertically (repeating the same line): here the three runs will be vertically coded because for each run, the b1 element is exactly on top of the a1 element.
- Line 4 is coded vertically: here the three runs will be vertically coded because for each run, the b1 element is always within three pixels distance from a1. At the first run, a1 is one pixel to the left of b1. At the second run, a1 is one pixel to the right of b1. At the last run, a1 is right under b1 both situated at the end of their line.
- Line 5 is coded with a pass code and a vertical code: since the line is made solely of white pixels, the a1 element is placed at the end of the line. The b1 element will be placed on the first pixel of the black strip and the b2 element on the last white pixel of the line 4 (the next changing element). Since b2 is to the left of a1, then a pass code is issued. The coding will continue after that passed run (at the last white pixel) and will coded vertically (b1 and a1 are both at the end of their line at that point).

Compression ratios achieved using this method is far better than the 1-dimensional compression scheme; The addition of the two modes provides more efficiency since the vertical mode is the most commonly occurring type of code. The typical compression ratio is from about 15:1 to 20:1 (or more depending on the resolution used).

4.2 Predictive Coding

This coding method is a transformation procedure that tries to reduce the complexity of an image in order to increase the compression effectiveness of other loss-less method

applied to the transformed image. The concept is simple: try to predict the next pixel based on its preceding neighbors. When the prediction is true, code the pixel as a success (usually as the background color); otherwise code it as an error (the foreground color). The more often the predictions are true, the easier the file will be to compress.

4.2.1 Differential Coding

This is the simplest form of predictive coding. The next pixel is predicted as being the same as the last one.

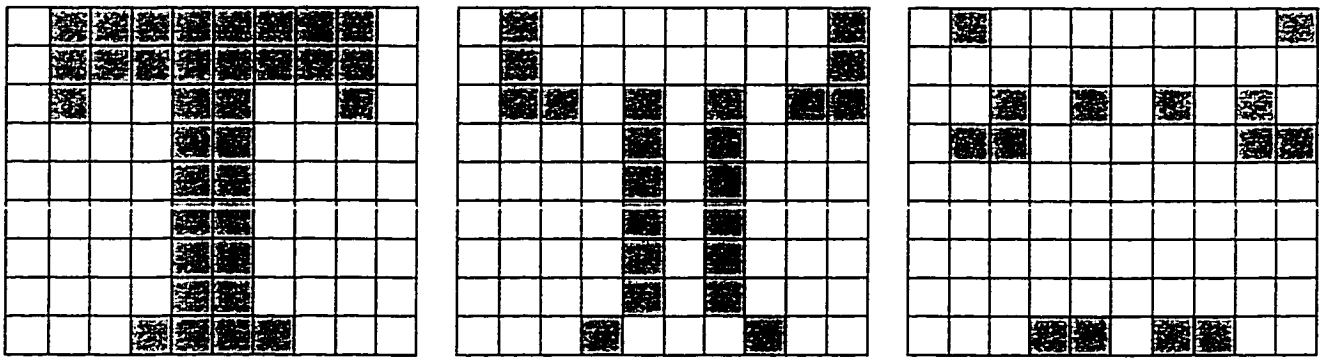


Figure 6. Differential Coding Example

In the above example, the original image on the left have been scanned from left to right and from top to bottom as image are normally scanned and stored. The image in the middle is the product of the differential coding. As you can see the black pixel denote the horizontal transition in the original image between black and white pixels. We can apply the same method on that image but now vertically; this result in the image on the right side. It should be easier to compress. Of course not every picture will become simpler by this transformation but it is generally the case with document imagery.

4.2.2 Standard Predictive Coding

In order to improve the transformation, we can take more pixels to predict the next one: the standard pattern is to use the last two pixels on the same line and five other on the previous line as shown in the next figure.

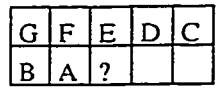


Figure 7. Standard Predictive Pattern

Use pixels A, B, C, D, E, F and G to predict the pixel '?'. The prediction rule is now more complex. We must use a predictor table that holds all the different possible value of A, B, C, D, E, F and G, and its corresponding prediction. The CCITT created a standard table based on statistics computed from 8 standard documents.

A	B	C	D	E	F	G	?	Accuracy
0	0	0	0	0	0	0	0	0.999
1	0	0	0	0	0	0	1	0.834
0	1	0	0	0	0	0	0	0.982
1	1	0	0	0	0	0	1	0.767
0	0	1	0	0	0	0	0	0.965
1	0	1	0	0	0	0	1	0.931
0	1	1	0	0	0	0	0	0.918
1	1	1	0	0	0	0	1	0.924
0	0	0	1	0	0	0	0	0.776
1	0	0	1	0	0	0	1	0.951
0	1	0	1	0	0	0	0	0.673
1	1	0	1	0	0	0	1	0.960
0	0	1	1	0	0	0	0	0.833
1	0	1	1	0	0	0	1	0.985
0	1	1	1	0	0	0	0	0.787
1	1	1	1	0	0	0	1	0.973
0	0	0	0	1	0	0	0	0.602
1	0	0	0	1	0	0	1	0.948
0	1	0	0	1	0	0	0	0.617
1	1	0	0	1	0	0	1	0.936
0	0	1	0	1	0	0	0	0.629
1	0	1	0	1	0	0	1	0.793
0	1	1	0	1	0	0	1	0.571
1	1	1	0	1	0	0	1	0.972
0	0	0	1	1	0	0	1	0.819
1	0	0	1	1	0	0	1	0.992
0	1	0	1	1	0	0	1	0.623
1	1	0	1	1	0	0	1	0.967
0	0	1	1	1	0	0	1	0.649
1	0	1	1	1	0	0	1	0.996
0	1	1	1	1	0	0	1	0.524
1	1	1	1	1	0	0	1	0.985
0	0	0	0	0	1	0	0	0.971
1	0	0	0	0	1	0	1	0.522
0	1	0	0	0	1	0	0	0.906
1	1	0	0	0	1	0	1	0.636
0	0	1	0	0	1	0	0	0.934
1	0	1	0	0	1	0	1	0.503
0	1	1	0	0	1	0	0	0.556
1	1	1	0	0	1	0	1	0.751
0	0	0	1	0	1	0	0	0.854
1	0	0	1	0	1	0	1	0.583
0	1	0	1	0	1	0	0	0.684
1	1	0	1	0	1	0	1	0.813
0	0	1	1	0	1	0	0	0.910
1	0	1	1	0	1	0	0	0.516
0	1	1	1	0	1	0	0	0.645
1	1	1	1	0	1	0	1	0.837
0	0	0	0	1	1	0	0	0.661
1	0	0	0	1	1	0	1	0.973
0	1	0	0	1	1	0	0	0.895
1	1	0	0	1	1	0	1	0.759
0	0	1	0	1	1	0	1	0.783
1	0	1	0	1	1	0	1	0.868
0	1	1	0	1	1	0	1	0.765
1	1	1	0	1	1	0	1	0.729
0	0	0	1	1	1	0	1	0.726
1	0	0	1	1	1	0	1	0.997
0	1	0	1	1	1	0	0	0.594
1	1	0	1	1	1	0	1	0.950
0	0	1	1	1	1	0	1	0.530
1	0	1	1	1	1	0	1	0.996
0	1	1	1	1	1	0	0	0.768
1	1	1	1	1	1	0	1	0.961

A	B	C	D	E	F	G	?	Accuracy
0	0	0	0	0	0	1	0	0.996
1	0	0	0	0	0	1	1	0.768
0	1	0	0	0	0	1	0	0.997
1	1	0	0	0	0	1	0	0.648
0	0	1	0	0	0	1	0	0.982
1	0	1	0	0	0	1	1	0.757
0	1	1	0	0	0	1	0	0.988
1	1	1	0	0	0	1	1	0.614
0	0	0	1	0	0	1	0	0.902
1	0	0	1	0	0	1	1	0.625
0	1	0	1	0	0	1	0	0.717
1	1	0	1	0	0	1	1	0.880
0	0	1	1	0	0	1	0	0.924
1	0	1	1	0	0	1	1	0.748
0	1	1	1	0	0	1	0	0.906
1	1	1	1	0	0	1	1	0.826
0	0	0	0	1	0	1	0	0.879
1	0	0	0	1	0	1	1	0.660
0	1	0	0	1	0	1	0	0.531
1	1	0	0	1	0	1	1	0.882
0	0	1	0	1	0	1	0	0.897
1	0	1	0	1	0	1	1	0.739
0	1	1	0	1	0	1	1	0.623
1	1	1	0	1	0	1	1	0.846
0	0	0	1	1	0	1	0	0.551
1	0	0	1	1	0	1	1	0.912
0	1	0	1	1	0	1	1	0.637
1	1	0	1	1	0	1	1	0.921
0	0	1	1	1	0	1	0	0.712
1	0	1	1	1	0	1	1	0.939
0	1	1	1	1	0	1	1	0.561
1	1	1	1	1	0	1	1	0.959
0	0	0	0	0	1	1	0	0.991
1	0	0	0	0	1	1	1	0.795
0	1	0	0	0	1	1	0	0.997
1	1	0	0	0	1	1	0	0.711
0	0	1	0	0	1	1	0	0.978
1	0	1	0	0	1	1	1	0.795
0	1	1	0	0	1	1	0	0.982
1	1	1	0	0	1	1	0	0.576
0	0	0	1	0	1	1	0	0.894
1	0	0	1	0	1	1	1	0.686
0	1	0	1	0	1	1	0	0.673
1	1	0	1	0	1	1	1	0.703
0	0	1	1	0	1	1	0	0.942
1	0	1	1	0	1	1	1	0.810
0	1	1	1	0	1	1	0	0.914
1	1	1	1	0	1	1	1	0.617
0	0	0	0	1	1	1	0	0.934
1	0	0	0	1	1	1	1	0.922
0	1	0	0	1	1	1	0	0.975
1	1	0	0	1	1	1	1	0.806
0	0	1	0	1	1	1	0	0.951
1	0	1	0	1	1	1	1	0.885
0	1	1	0	1	1	1	0	0.920
1	1	1	0	1	1	1	1	0.748
0	0	0	1	1	1	1	0	0.756
1	0	0	1	1	1	1	1	0.975
0	1	0	1	1	1	1	0	0.849
1	1	0	1	1	1	1	1	0.929
0	0	1	1	1	1	1	0	0.830
1	0	1	1	1	1	1	0	0.957
0	1	1	1	1	1	1	0	0.848
1	1	1	1	1	1	1	1	0.989

Figure 8. CCITT Standard Predictor Table ('0' represents background color (usually white) and '1' represent foreground color (usually black))

4.2.2.1 Goodness Threshold

Predictors have different level of success, for example: when A, B, C, D, E, F and G are all white, the next pixel is predicted to be white, and based on CCITT statistics, this will be true 99.9% when coding a typical document imagery. When B, D, E and G are white and the rest black, the next pixel is predicted to be black, but this will be true only 50.3%. A goodness threshold can be use to separate reliable predictors from the rest. With this distinction it is possible to group together the result of reliable predictors from the unreliable ones. Consider that the goodness threshold is 90%. Whenever a prediction has 90% or more chance to be right then the result is coded as normal. But when the prediction has less than 90%, then the result can be placed at a different part of the file (either at the end of the scanned line or at the end of the entire file). This insure that the first part of the coded image will be at least 90% all white (or coded as a success) and will be very compressible. The second part will look more like static and will be difficult to compress but usually the second part is a small portion of the entire image.

4.2.2.2 Adaptive Predictive Coding

In order to increase the effectiveness of predictive coding you can always build your own predictor table. This insures the best coding possible since the table is adapted to your documents. The inconvenience is that if you build a predictor table for each document, you have to include your adapted table with the compressed file, or else you will not be able to decode it. To create your own predictor table is a good idea when the documents you want to compress are not typical document imagery.

4.2.2.3 Augmented Predictive Coding

Another way to increase the effectiveness of predictive coding is by using even more neighboring pixels, for example:

	J	I	H	
G	F	E	D	C
B	A	?		

Figure 9. Augmented Predictive Pattern

Use pixels A, B, C, D, E, F, G, H, I and J to predict the pixel '?'. This slightly increase the effectiveness of the compression compared to the standard. It adds more complexity and a bigger predictor table.

4.3 Morphological Transformations

Unless it is generated by a word-processor software, the quality of a document imagery is not perfect: the page might be crooked, borders can be black, dirt, scratches, clips, staples, wrinkles and folds add noises to the image, and the overall quality drop when the source is a multiple generation photocopy. This decreases the effectiveness of compression. One way to filter out noise out of a document is by applying standard morphological transformations [4] to it.

A morphological transformation will be effective if it increases the compression effectiveness (by hopefully augmenting the quality of the prediction of information) and if the interpreted information is preserved.

4.3.1 Structuring Element

A structuring element is an entity used to probe an image. It is a pattern of pixel; one of the pixel represents the pixel of interest (it is usually the center of the pattern) and the rest are its neighbors. Here are some examples:

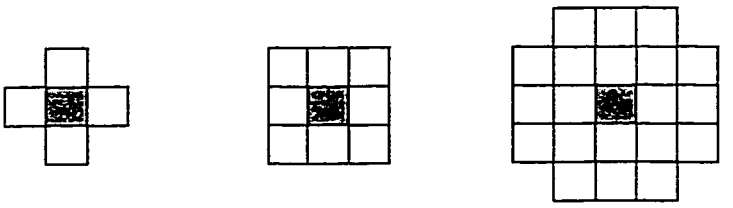


Figure 10. Examples of Structuring Elements

The black pixel represents the pixel of interest. These patterns are used with the various morphological operations.

In order to illustrate the effect of the following morphological operations, consider the next figure as a source image. It represents the capital letter 'L' in some fancy font followed by a period, and some noisy pixels are included. All the operations have been made with an 8-neighbor structuring element.

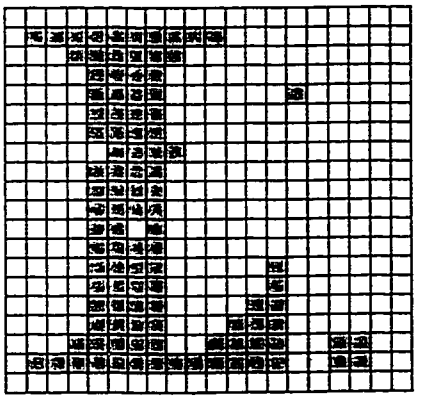


Figure 11. Source Image

4.3.2 Erosion

The algorithm of erosion is very simple: scan the entire image pixel by pixel. If the pixel is a background color (white) then it stays the same in the output image. If the pixel is a foreground color (black) then center the structuring element on the pixel. If one of its neighbor is of background color then change the pixel to a background, otherwise it stays as foreground. This transformation essentially peels off bordering pixels of foreground objects.

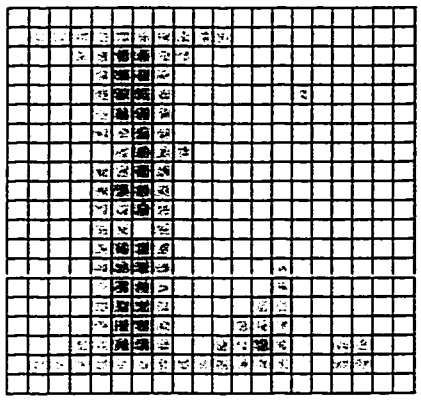


Figure 12. Source Image subjected to Erosion (the light gray pixels are those removed by the process)

4.3.3 Dilatation

This transformation is the opposite of erosion: scan the entire image pixel by pixel. If the pixel is a foreground color then it stays the same in the output image. If the pixel is a background color then center the structuring element on the pixel. If one of its neighbor is of foreground color then change the pixel to a foreground, otherwise it stays as a background. This transformation fattens up foreground objects by adding layers of bordering pixels.

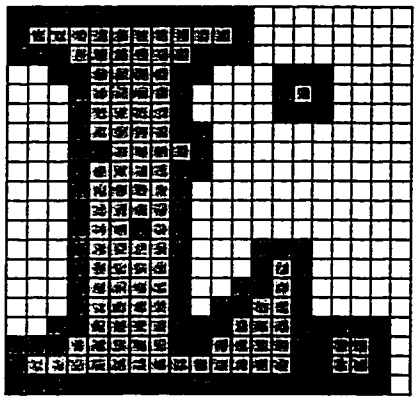


Figure 13. Source Image subjected to Dilatation (the dark gray pixels are those added by the process)

4.3.4 Closing

The closing operation is simply to dilate the image first and then erode the result. This operation is used to close small pockets of white pixel on black characters or objects. It also fills in small inward-dent. It has a side effect of connecting together very close objects.

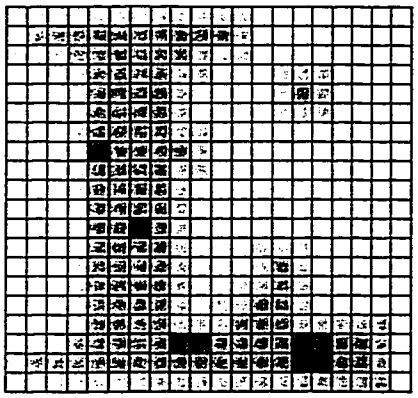


Figure 14. Source Image subjected to Closure (the dark and light gray pixels are those added and removed by the process respectively).

4.3.5 Opening

This operation is the opposite of closing: erode the image first and then dilate the result. This operation is used to get rid of small object (a single black pixel for example) and dent. It has a side effect of breaking up objects if they have too thin regions.

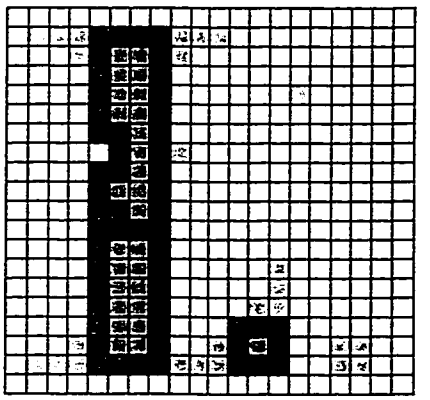


Figure 15. Source Image subjected to Openness (the light and dark gray pixels are those removed and added by the process respectively).

4.3.6 Trim

This operation is not a standard morphological transformation; I designed the operation to get the same result of the open transformation (with an 8-neighbors structuring element) but with milder side effects. The algorithm is the following: scan the entire image pixel per pixel. If the pixel is a background color then it stays the same on the output image. If the pixel is a foreground color then count how many neighbors are also of that color. If less than four are foreground than change the pixel for background. If more than four are foreground then the pixel stays the same. If exactly four neighbors are foreground and those are one of the two following figures then it change for background, otherwise it stays the same.

Figure 16. Trim Special Cases

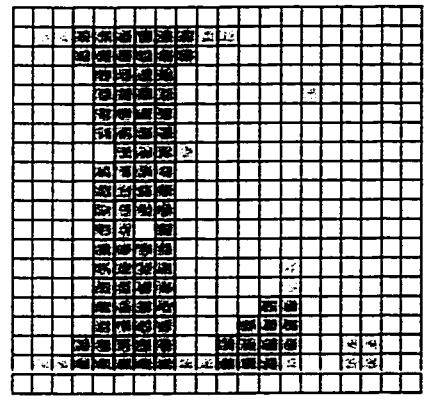


Figure 17. Source Image subjected to Trimming (the light gray pixels are those removed by the process).

4.4 Cartesian Perceptual Compression

Since CPC [6] is a patented product, not many details are available to the public on the description of the method except for the patent description and the product web site. In this research, CPC has been used more as a landmark, a goal to achieve: to create a method that compress as much or even more than CPC. Although the new compression method, Pattern Compression, does not compress as much as CPC, it does offer better compression than the standard and some advantages over CPC: several parameters of the algorithm can be set by the user and this give control over the effectiveness of the compression and the amount of data lost. They are both based on the same concept, which is the topic of the next section: Vector Compression. The following are extracts taken from the Cartesian web site [6.1]:

"Cartesian Perceptual Compression (CPC) is a novel patented image compression method specifically designed for document image storage and transmission systems. Designed by Cartesian Products, Inc., a high-technology research and development company based in Cambridge, Massachusetts, Cartesian's compression technology is the state of the art in document image compression methods, providing an order of magnitude better compression or more than other document image compression methods."

"The invention allows for the storage of documents in a fraction of the space that would be required when using today's standard technology. For instance, some documents have been compressed 20 to 30 times smaller than the industry-standard CCITT Group 4 compression method. Results across a broad spectrum of documents indicate that improvement of 10 to 1 over the Group 4 method is quite common."

"The Cartesian image model is based on novel pattern analysis technology. The model assumes that patterns in the image tend to be mutually similar and to recur. Thus, by predicting new patterns from old ones, strong expectations, and therefore better compression, can be achieved. The pattern analysis model is based on an underlying model of perceptual discriminability. Patterns that are indiscriminable may be treated as redundant, thereby achieving greater compression. Because the model takes into account perceptual factors, the reconstructed image is perceptually indistinguishable from the original."

Description of the CPC method based on the patent document [6.2]:

The compression system according to the invention will firstly segment the input image into symbols (equivalent to the concept of "pattern"). Each symbol is matched against a library of templates (equivalent to the concept of "codebook"). The template library is generated from scratch as new symbols are identified. Each identified symbol is matched against templates created so far, and it is added to a matching template group (equivalent to the concept of "cluster") if one exists. If it matches none of them, it forms a new template group and serves as the template for that group so that later symbols can be matched to it. Once all of the symbols are grouped into template groups, a representation of each template is composed, preferably based on features of many, if not all, of the members of the template group. Also, a representation of the location of each symbol in

the image and the symbol's template group is built. Thus, the compressed representation of the image will constitute in four streams of information: a template stream, a template code stream, and two streams for the location of symbols, one for the rows and one for the columns. The streams of information are each preferably further compressed.

Other Details of the method:

Small Symbols: Too small symbols (less than four pixels) are eliminated during the symbol identification process. Apparently, those are considered to be probably scanning or printing error and can disrupt the compressing process.

Symbol Matching: In order to match a symbol to a template, the method uses a voting scheme, where a plurality of tests is used for each symbol/template pair. Each test, which is passed (indicating similarity between the symbol and the template) contributes one vote to a sum of votes. In order for a match to be declared, a predetermined number of votes must be received. More generally, the tests can be grouped into one or more predetermined groups, and all tests within at least one of the groups must be passed for a match to be declared.

- Error Pixel: When overlapping a symbol on a template for matching, pixels that do not match are considered error pixels.

- Edge Test 1: All error pixels must be within a fixed distance of an edge.

- Edge Test 2: No block of error pixels may be larger than a defined size.

- Edge Test 3: The difference in heights or widths of the symbol and the template must be less than or equal to a fixed bound size.

- Density Test: The ratio of the number of black pixels in the symbol to the number of black pixels in the template must be within a defined range.

- Error Test 1: The ratio of error pixels must be less than some defined distortion bound.

- Error Test 2: The ratio of error pixels to black pixels occurring on an edge in the template or in the symbol must be less than a defined distortion bound.

- Shift Test: All inferred shifts of the symbol relative to the template must be consistent.

Representation of the Templates: When choosing the representative of a template group, the method is this: The first encountered symbol is used in the matching process, but other symbols added to the template group are allowed to vote and change any given pixel. The vote will succeed if more than two-thirds, of the other symbols disagree with the original symbol.

Frames and Holes: A symbol can be decomposed into a "frame" (the outer edge of a symbol) and some "holes" (inner edges in the symbol). Each frame and holes are considered as symbols and are compressed recursively. The original image of the templates can then be represented by a reference to a frame template and a list of hole templates with their associated positions in the frame.

Neighboring characters stuck together: Error in scanning or printing text can cause neighboring characters to run together. Those characters are divided. The splitting is performed on the templates after all symbols have been matched into template groups. To avoid splitting templates that happen to have a "bridge" in them, the two halves are preferably required to be of a sufficient size.

Representation of the Template Codes and Symbol Locations: An approximation of the reading order of the characters is reconstructed. This is done by carefully tracking the lines in the text. After obtaining this information, the symbols will be coded using the

reading order. Each symbol position is coded based from the previous character position. The displacement is from the lower right corner of the previous character to the lower left corner of the current character.

Chapter 5. Vector Compression

5.1 Vector Quantization

A vector is a group of one or more data stream elements (in our case, bits). A data stream can then be described as a list of vectors. This list can be compressed by assigning short codeword for frequent vectors and long codewords for infrequent vectors. In addition it can also be compressed by limiting the number of different vectors. Instead of allowing all the possible vectors, a certain number of representatives (or models) can be determined. The list can then be expressed using only this limited set of vectors; this set of vectors is called the codebook. The vectors that are not part of the codebook are represented by the closest (the most similar) vector of the codebook. The more restricted is the size of the codebook, the more efficient will be the compression but the more information will be lost.

A vector compression algorithm depends on three things: size and shape of a vector, the algorithm that construct the codebook and the distance function (used when mapping vectors to representatives).

Traditionally, algorithms that construct the codebook are limited by a maximum number of representatives and the problem is to find the best representatives possible such that distortion (loss of information) is minimal. This is required for coding systems that do not

create adaptive codebook. These systems use a fix codebook which have been typically created with some training example (see [5.1]).

Here is a general algorithm to create a codebook with as less distortion as possible based on a training sequence. The algorithm was developed for vector quantizers, training sequences, and general distortion measures by Linde, Buzo, and Gray and it is sometimes referred to as the LBG algorithm [5.2].

Step 0: Start the process with an initial codebook (which have a fix number of representatives) and a training sequence (some input data used to perfect the initial codebook).

Step 1: Encode the training sequence into vectors. Compute the average distortion (the information lost by coding the training sequence with the original codebook). If it is small enough, quit.

Step 2: Replace each codebook representative by the centroid of all training vectors that were mapped by that representative. Go to step 1.

What this algorithm does basically is to consider the position of a representative in relation to the vectors around it (that are mapped by it), and step by step the representative move in order to better represent its surrounding. At every iteration, the algorithm should either reduce the average distortion or leave it unchanged. The algorithm is usually stopped when the relative distortion decrease falls below some small threshold.

5.2 A Different Approach to the Problem

It is very important not to allow too much distortion, especially with document imagery, because if too much distortion is allowed the document can become unreadable and this

is not acceptable. A problem with the traditional vector quantization algorithms is that they do not guarantee a maximum level of distortion. So instead of playing with a fix number representatives until a minimum distortion is achieved, consider a different approach to the problem: let a maximum distortion allowed be fixed and try to find the least number of representatives such that no input vector is distorted more than that limit. The maximum distortion can be controlled by establishing a fix radius around representatives. The problem is then to create the least number of representatives such that all the input vectors are included inside the radius of one representative. This way, the minimum distortion can be achieved by finding the least number of representatives and imposing a maximum distortion control the lost of information; it is a better assurance that the output will be interpreted the same as the input.

The following methods use this approach; they build an adaptive codebook for each document, they are bounded by a maximum distortion and they are not limited by a maximum number of representatives.

5.3 Line Compression

The first method designed was a 1-dimensional vector compression. In this method an entire line of an image represents a vector. The algorithm to construct the codebook is the same as the Pattern Compression (see next). The distance function is simply counting the number of different pixel between two vectors. The resulting compression ratio generated by the method was very interesting but unfortunately changes in the resulting file, after decompression, were too visually perceptible.

5.4 Pattern Compression

So the next step was to create a special 2-dimensional vector compression method. The following is a general description of the method. A vector is a small 2-dimensional

portion of an image. More specifically, each character of a document imagery should be a vector. A vector have a limit size, and if a character is bigger than this limit, then it can be broken into many pieces or it could be compressed using a different technique. The algorithm to construct the codebook is explained in step 2 of the description of the method. As you will see in the "Test Result" section of this document, this method generates very good compression ratios and very good output.

5.4.1 Term Definitions

Before stepping into the description of the method, certain terms should be well defined.

Pattern: Group of black pixels connected horizontally and vertically.

Pixel Distance: considering the image as one stream of pixel, the distance between two pixels on a page is the number of pixel separating them. The distance between two patterns on a page is the pixel distance between the top left pixel of each pattern.

Vector Distance: the distance between two patterns in the vector space is defined by summing the number of different pixel when overlapping the two patterns. Since border pixels are perceptibly not as important as non-border pixels, only a fraction of a different border pixel can be counted. When a pattern is too big and broken down, the broken border pixels that were non-border pixels must not be considered as border pixel.

Border Pixel: A black pixel with at least one white neighboring pixel.

Cluster: A cluster is a group of close vectors (perceptibly closely similar patterns).

Representative: All the slightly different vectors in a cluster will be encoded using only one pattern, the representative. One of the vectors can be chosen (the closest one to the cluster center) or the center of the cluster can be used as the representative.

5.4.2 General Description of the Method

The following is general description of the method; each step can be implemented in different ways. Also some steps can be done simultaneously: for example, patterns can be extracted while building the codebook.

Step 1: Extract Patterns

Scan the input image and extract each pattern.

Pattern Size limit: A size limit can be enforced. If a pattern is bigger than the limit, it can be broken down (only the pixels within the limit are considered) or it can be entirely dismissed and be compressed using an alternative method.

Step 2: Create CodeBook

Using a vector distance rule and a fix cluster radius, regroup patterns in the vector space in such a way that each pattern is part of a cluster and that the number of cluster is minimal.

The following algorithm is an example of a simple algorithm to create the codebook. It does not however guarantee a minimal number of clusters.

- Consider the patterns P0 to Pn
- Set P0 as the center of cluster C0
- Consider Pi
  - Find the closest cluster Cc
  - Is Pi inside the cluster Cc? (Does the distance between Pi and the center of Cc is smaller than the cluster radius?)

- Yes: Add Pi to the cluster Cc. Adjust the center and the representative of the cluster.
- No: Pi becomes the center of a new cluster.

- Repeat for all patterns.

Step 3: Compression

The compressed output is simply made of these 3 parts:

- The codebook (description of the representatives)
- List of representatives representing the image. This should be a list of references to the codebook obviously, not a copy of each representatives.
- List of pixel distance between each pattern.

Each of these parts can be further independently compressed; see the implementation of the method for examples.

5.4.3 Implementation of the Method

In this section, the general method will be repeated in order to specify how the method was implemented.

Step 1: Extract Patterns.

A filling algorithm (pattern recognition technique) was used to extract each pattern. The size limit of a pattern is 50x50 pixels. Too big patterns were broken down into multiple patterns. The extraction was made simultaneously with the construction of the codebook.

Step 2: Create CodeBook

The simple algorithm mentioned in the previous section was used to build the codebook.

- Consider the patterns P0 to Pn
- Set P0 as the center of cluster C0
- Consider Pi
  - Find the closest cluster Cc:
    - Calculate the distance between Pi and every cluster (see next section about details on the implementation of the codebook)
    - To calculate the distance between pattern Pi and cluster Cj, use the Hamming distance: it can be easily computed by overlapping Pi on Cj and align them with their respective horizontal and vertical mass (see the alignment problem in the next section). Add the number of pixels that differ between the two. In addition, modify some pixel value if they are border pixel or broken border pixel (see next section).
  - Is Pi inside the cluster Cc? (Does the distance between Pi and the center of Cc is smaller than the cluster radius?). To find out, compare the distance between Pi and Cc to the following cluster radius (see next section for the explanation of the equation):Crf * ((Cnbp * Cweight + Pnbp)/(Cweight + 1))
    - Yes: Add Pi to the cluster Cc. Adjust the center and the representative of the cluster. This is done by incrementing the number of patterns Cc holds and by recalculating the representative, which is the average picture of all patterns of that cluster.
    - No: Pi becomes the center of a new cluster.
- Repeat for all patterns.

Step 3: Compression

Using the number of pattern in a cluster as its frequency, the Huffman algorithm is used to create a codeword for each representative. The compressed file is simply made of these parts:

- The codebook + their respective codewords: the codebook is compressed with the CCITT group 4 method [3.2].
- List of codewords representing the image.
- List of pixel distance between each pattern. This list is compressed by calculating the frequency of each distance and replacing each pixel distance by its Huffman codeword.

5.4.4 Other Implementation Details

Here are more specific aspects on how the method was implemented:

Modified Border Pixel Value: Border pixel value is modified by 0.9. A border pixel is a pixel lying on the outer edge of a pattern. Since border pixels are perceptibly not as important as non-border pixels (it is less perceptible to change a pixel lying on the outer edge of a pattern than one inside) the value of such pixels are modified when calculating a vectorial distance between two patterns. Only 90% of the value of border pixels was considered.

Modified Broken Border Pixel Value: Broken border pixel value is modified by 2. When a pattern is too big and is broken down, the broken border pixels that were non-border pixels before must not be considered as border pixel. So in order also to reduce the effects of the broken pattern problem (see section 5.4.6), the value of such pixels are increased by 200% when calculating a vectorial distance between two patterns.

Cluster Radius Function: To know if a pattern lies inside a cluster, the distance between the pattern and the middle of the cluster must be smaller than this number: Crf * ((Cnbp * Cweight + Pnbp)/(Cweight + 1))

- Crf: Cluster radius factor. This represents the proportion of patterns that can be different and still be considered in the same cluster. The cluster radius used was 0.30.
- Cnpb: The number of black pixel of the cluster representative.
- Cweight: The number of patterns already in the cluster.
- Pnbp: the number of black pixel in the pattern to compare.

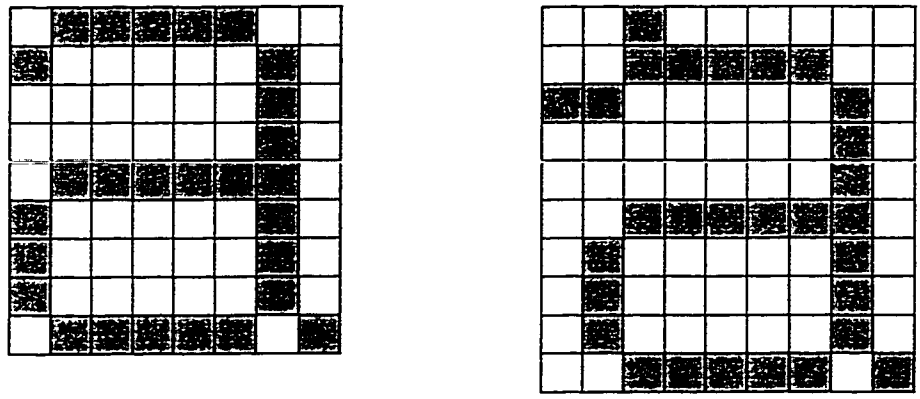


Figure 18. The Alignment Problem

The Alignment Problem: Patterns have different sizes and must be aligned properly to compute the right distance. If in the example above, the two patterns are aligned with the top and left sides, the distance between them will be unjustly high. The solution used here is to compute the horizontal and vertical mass center of each pattern and align them using this information.

CodeBook Implementation: The list of clusters is indexed by the number of black pixels in each representative. This allows us to restrain the matching search of a pattern to possible clusters. The search always begins with clusters with the same number of black

pixels as the pattern and then those with more or less black pixels alternatively until the best match is found, or until no possible match is possible.

Compressed CodeBook Format: The codebook is represented as a separate image file where each representative sits next to each other side by side horizontally. This file is compressed using CCITT group 4 [3.2].

5.4.5 Pattern Compression Example

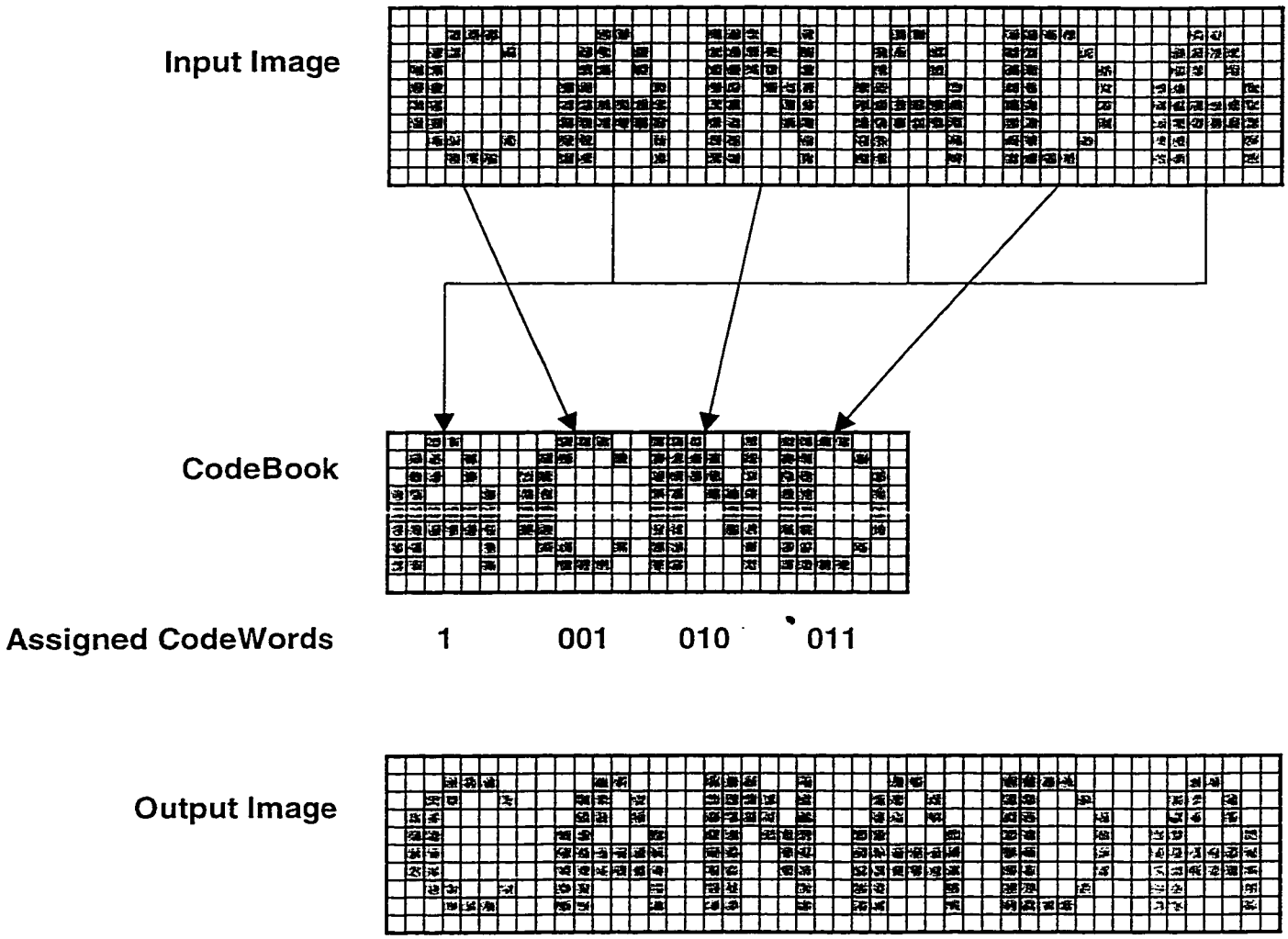


Figure 19. Pattern Compression Example

This section will illustrate the compression process with an example.

The input file represents a small document containing only one word "CANADA". Twice the process will scan this file. The first time to construct the codebook. The second time to construct the compressed file.

Each letter of the word "CANADA" represents a pattern (a distinct "blob" of black pixels connected horizontally and vertically). Each pattern will be matched against each other's during the construction of the codebook in order to create clusters. For example, the three "A", although each slightly different, are close enough to form a cluster. The other characters "C", "N" and "D" are too much different from each other and from the "A" cluster and so they each one form their own cluster.

Once the codebook created, every cluster is assigned a codeword. They are calculated by the Huffman algorithm, which make sure that heavy clusters (clusters formed with a lot of patterns) will receive the shortest codeword since they are the most frequent. In our example, the cluster "A" gets the smallest codeword and the rest are assigned codewords similar in length.

The final compressed file will consist of the codebook image, each respective representative codeword, a list of codeword describing the input image (for this example, it would be "001101010111") and a list of pixel distance between each pattern on the input image (for this example, each pattern is at 8 pixels distance from its predecessor except for the first one).

The output image represents the reconstructed image from the compressed information. As you can see, all the letters "A" are now similar, since they all come from the same representative. Some data have been lost from the original but not the information. The amount of lost of data can be controlled with the cluster radius factor.

5.4.6 Advantages and Disadvantages of Pattern Compression

Advantages:

- Control over the algorithm parameters: pattern size limit, border pixel modifier, broken border pixel modifier and cluster radius factor. These parameters affect the effectiveness of the compression and the quality of the resulting document. This is an advantage over CPC [6] (see section 4.4).
- Better compression than CCITT Group 4 [3.2] (see Test Results).
- Loss of information not perceptible (except for broken pattern).
- Color image can also be compressed using this method: simply compress each bilevel color plane separately.
- Decompression time is fast.

Disadvantages:

- Compression time is rather long. The process to create the codebook and creating the compressed file is an expensive process. The input file must be scanned twice; once to construct the codebook and once to construct the compressed file based on the codebook. For each scan, every pattern must be matched to a portion of the codebook in order to find the best match (closest cluster). Another factor that slowed the project is that all the methods were implemented in Java. Java is a high-level computer language. This was a very good idea at the beginning of the project since a lot of methods had to be implemented. Unfortunately, since Java is interpreted and not compiled, programs are very slow to run. One page could easily take one full day to compress on the faster machines.
- Compression effectiveness is not as good as CPC.
- This method has been developed for document imagery; other type of imagery may not be compressed as well.

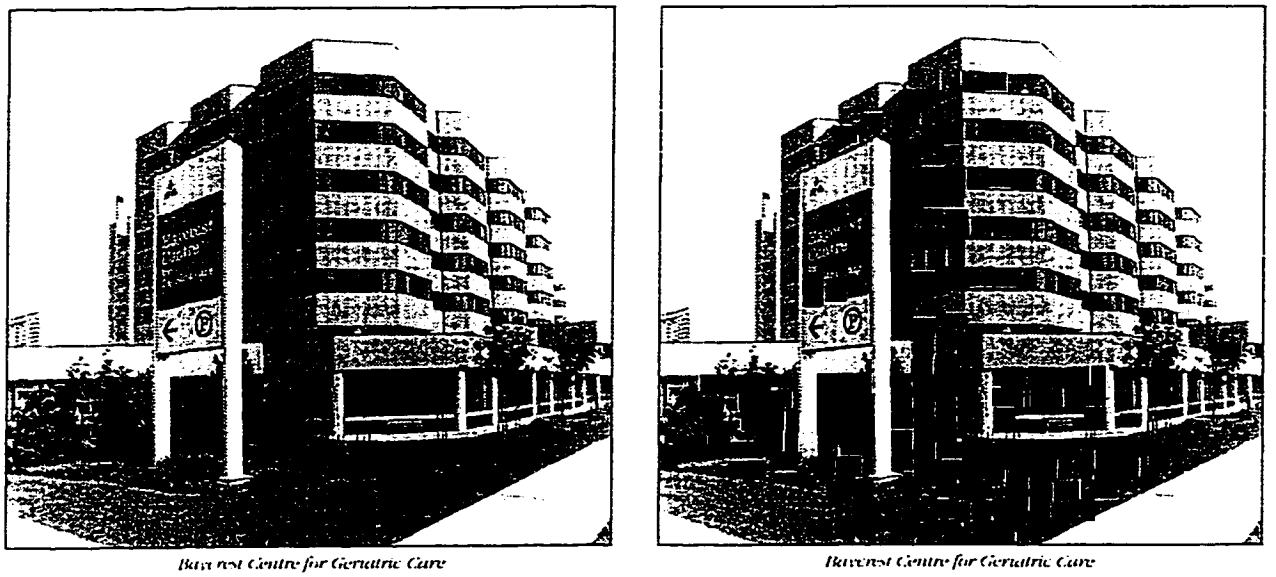


Figure 20. The Broken Pattern Problem

- The Broken Pattern Problem; unfortunately it is sometimes possible to notice the contour of broken patterns. This is especially evident when a region is very dense in black pixel. You can see this problem in the example above; the picture on the left is part of the original document and the picture on the right is the result after compression. Big patterns with a lot of black pixels can have more pixels changed and still remain visually the same. But this is a problem especially with broken pattern, when the region is almost all black. Too much change is allowed, and the result is not always visually be the same. Various solutions can be approached to rectify this problem, here are some:
  - One easy solution would consist in not breaking too big patterns but to compress them with a different algorithm (one more suited for picture for example)...
  - Another possible solution would be to deal with broken pattern with tougher constraint: for example, by reducing the cluster radius factor. This would create more accurate patterns.
  - The Inverse Density Correction: when the number of black pixel in a pattern exceed a certain level (like 50% of the density of the region) consider the white

pixels as the pattern to match instead of the black. This solves the problem by balancing the right amount of change allowed. This idea was tested; it does not solve the problem completely but does reduce it considerably. Unfortunately, the method was not implemented during the various experiments.

Chapter 6. Experiments

In this section, results of experiments and tests performed with the previously described algorithms are presented. Almost every method described in chapter 4, the CCITT compression standards [3], predictive coding [2] and the morphological transformations [4] (except for the trimming operation), have been implemented by myself based on the respective description given by each reference. The morphological transformation trimming has been designed by me. I also completely designed and implemented the new method Pattern Compression. Cartesian Perceptual Compression [6] is a patented product and for this project, the software "CPC Tool" was used to compress document with the CPC algorithm; this software is available from the Cartesian Company's web site [6.1]. The first experiment was to compress a common 1-page document with all the different methods and compare results. This 1-page document is a typical badly scanned article; it is filled with noises, it is slightly crooked and it has black contours... The whole page is displayed in chapter 3 as figure number 2.

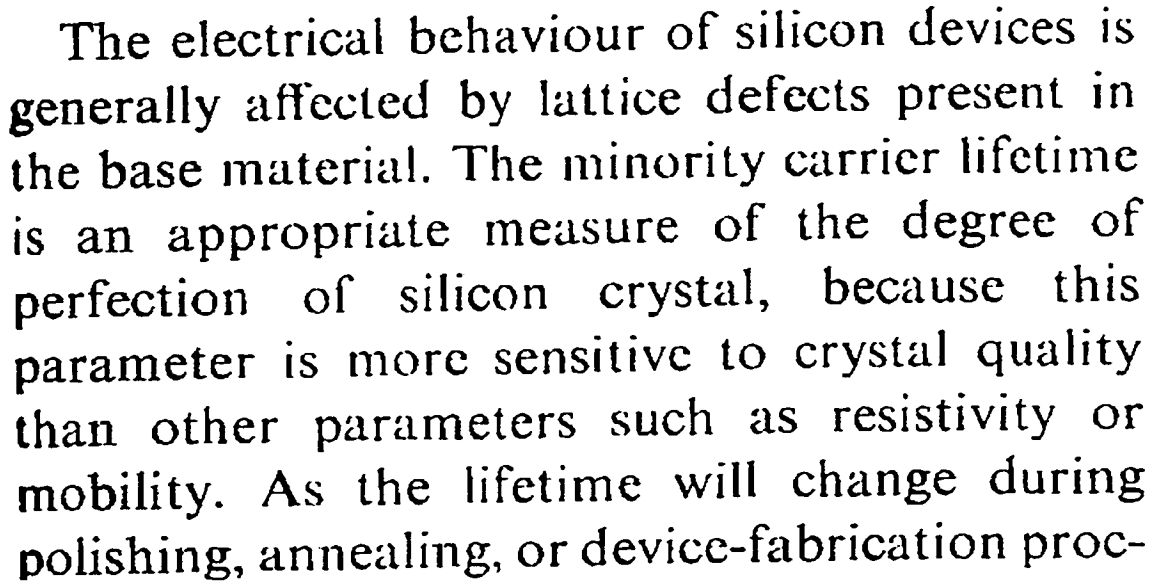


Figure 21. Specific section (the first half of the first paragraph) of the test document.

This document has an uncompressed size of 1.05 Meg approximately. It is reduced to 211K using the 1-dimensional CCITT standard [3] and to 82K using the 2-dimensional. A compression ratio of around 5:1 and 13:1 respectively.

6.1 Predictive Coding

Predictive coding [2] produces very peculiar result imagery. The CCITT standards [3] are ill suited to compress this type of imagery since it is not typical document imagery. Lossy scheme cannot be considered since no information can be lost in the prediction code in order to revert back to the original document. To get the best compression, the algorithm must be adapted to typical predictive coded imagery.

An adaptive predictive coding combined with an adaptive run-length algorithm (adaptive version of the 1-dimension CCITT standard) compress the test document down to 78K. Augmented predictors compress the document down to 77K.

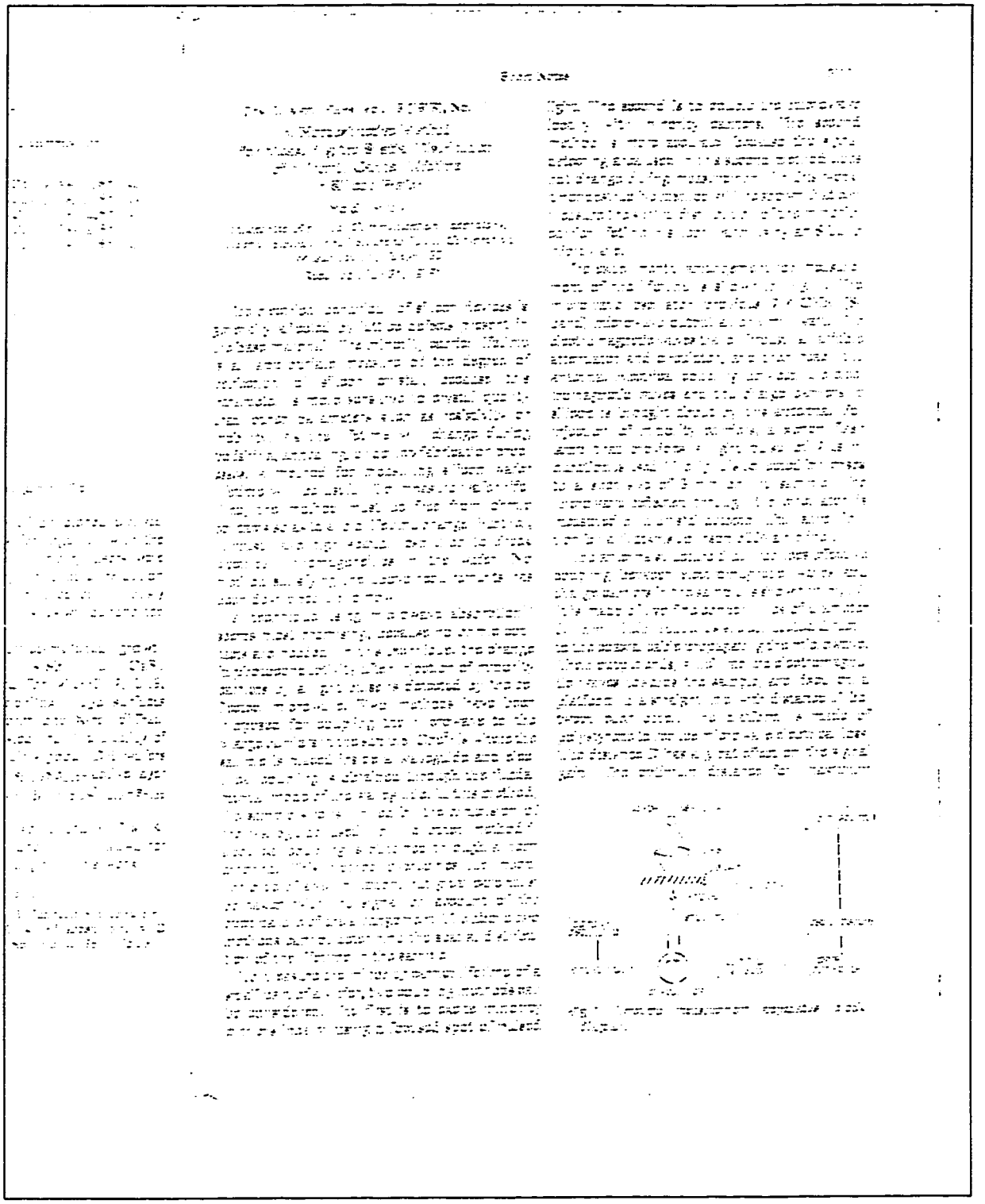


Figure 22. This is the result of standard predictive coding applied to the test document. Here no action has been made about the goodness of the predictor (no ordering of results).

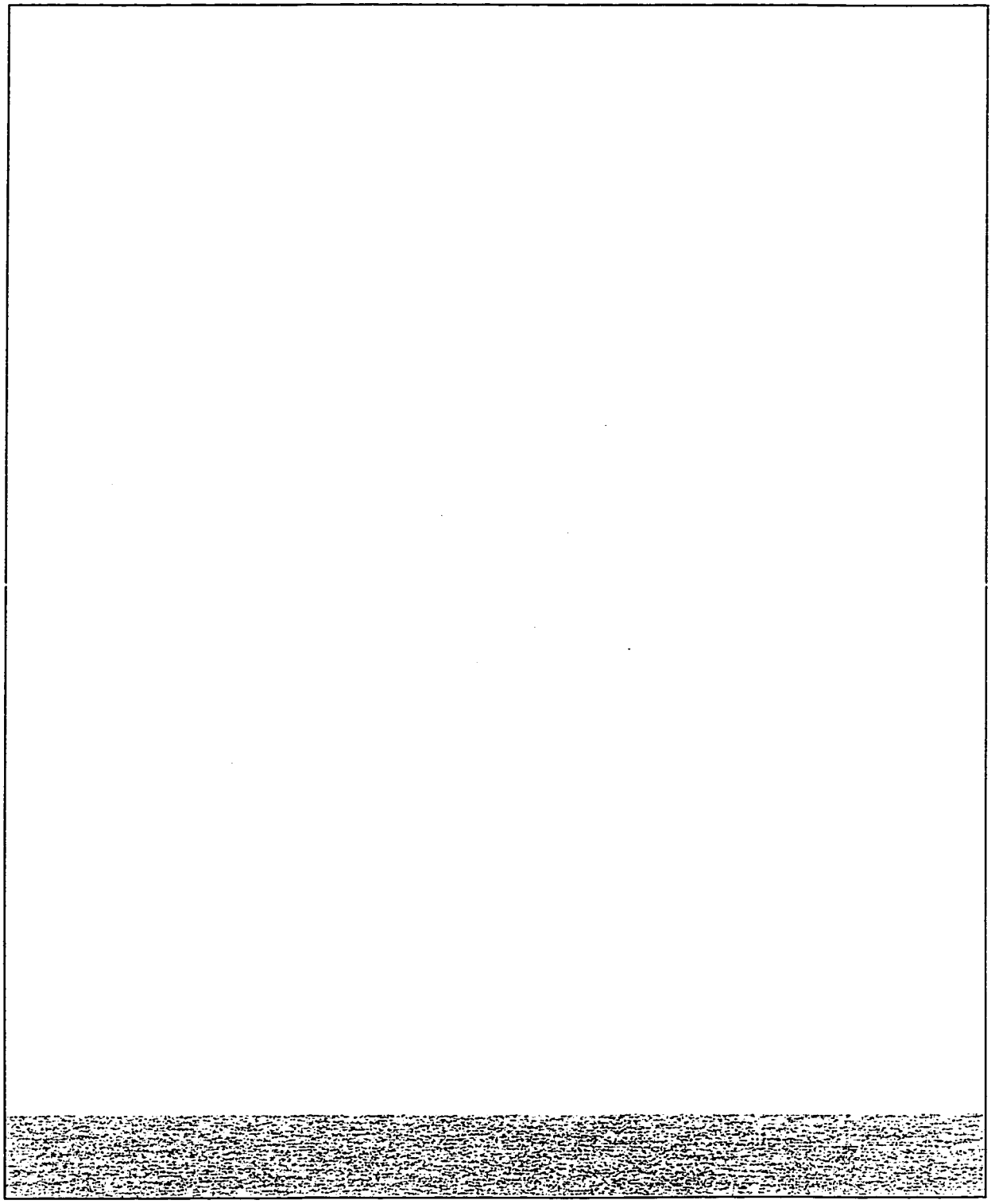


Figure 23. This is the result of adaptive predictive coding applied on the test document. Here a goodness threshold has been applied and the result of bad predictors has been put at the end of the file.

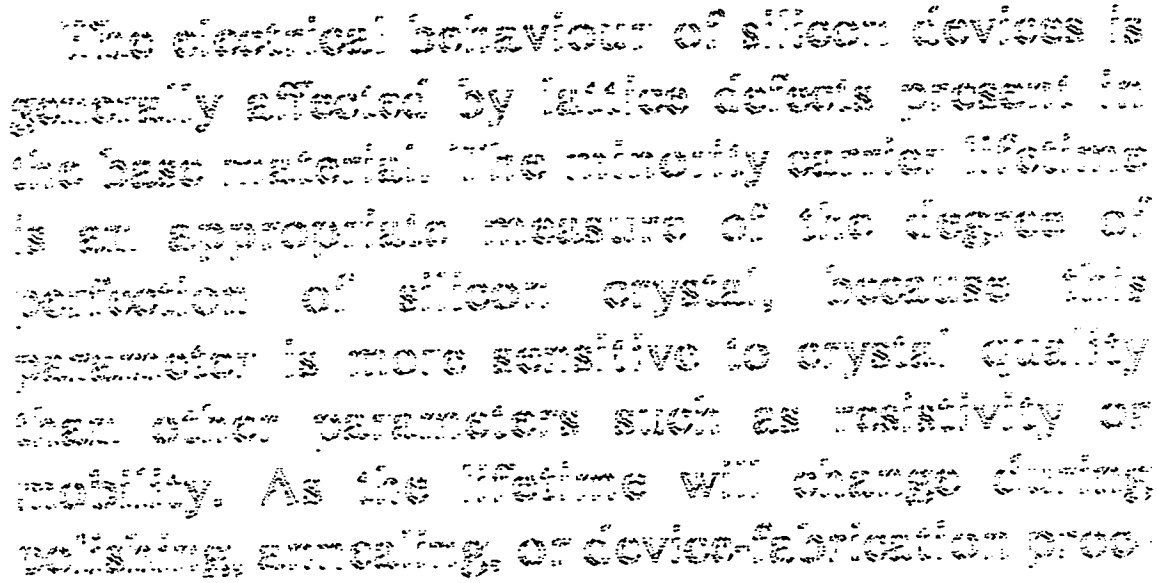


Figure 24. Here is a section of the result of standard predictive coding.

6.2 Morphological Transformations

Of all Morphological Transformations [4] presented previously, the use of the closing operation and of the trim produces the most effective compression. The test document transformed with the close operation compresses to 76K with the CCITT group 4 algorithm [3.2]. Trimmed, it compresses to 74K. And if the trimmed document is subject to closure, the result compress to 71K with the CCITT group 4 algorithm and down to 68K if compressed with adaptive predictor coding and adaptive run-length compression.

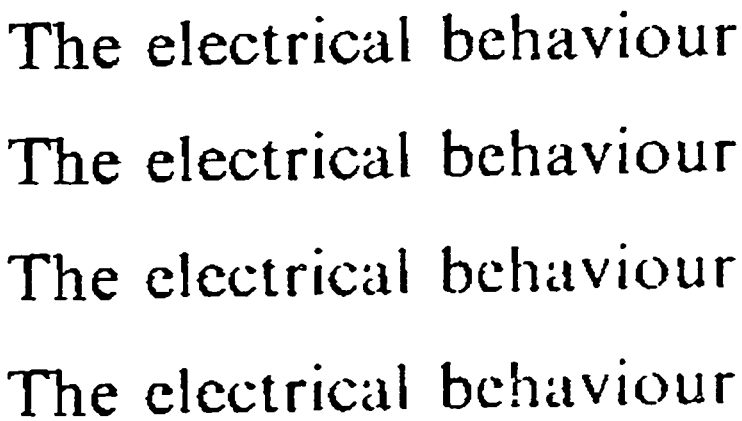


Figure 25. Morphological Transformations Comparison

This is a section of the test document that was subject to different operations: the top section represents the original document. The second section has been transformed with the close operation: some inward dents on the surface of some letters have been filled (look at the capital 'T', the right leg of the first 'h' and the top of the letters 'r'). The next section have been trimmed: very thin lines have been eliminated (look at the letters 'e', the 'a', the 'c' and the 'o') and some outward dents have been eliminated as well (look at the capital 'T' and its neighbor 'h'). The bottom section has been trimmed and then subjected to the closing operation: respective effects of both operations can be observed on the letters.

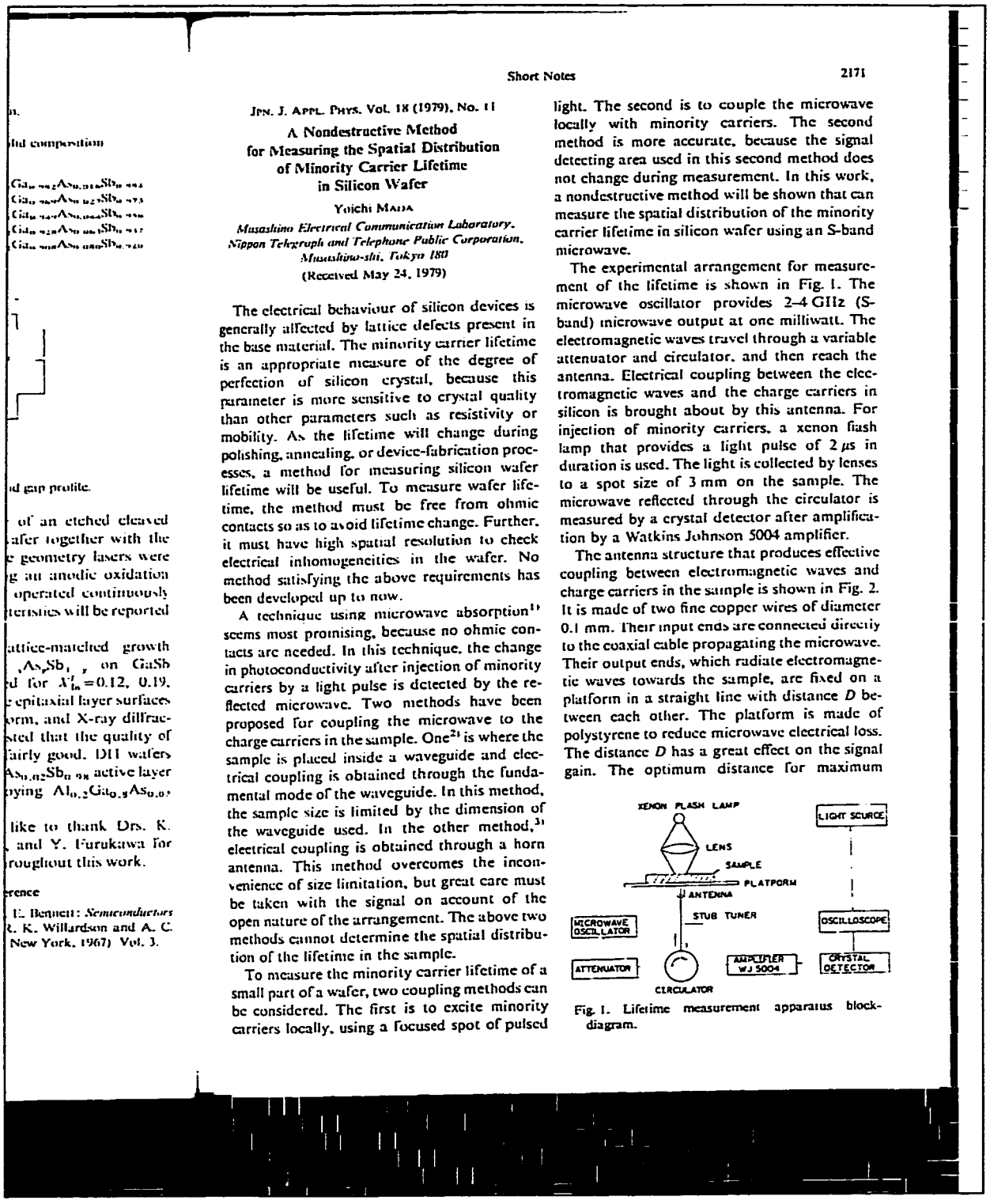
Although these morphological transformations help in the effectiveness of compression, the improvement is not phenomenal; the best result gets a ratio of 15:1. We would expect a little more from a lossy process...

6.3 Pattern Compression

Pattern Compression compresses the test document down to 33k, a ratio of 30:1. That's twice the ratio of the CCITT standard 2D compression [3.2].

The result page after compression is presented in the next figure. The various white lines at the bottom of the page are created by the problem of broken pattern (discussed in section 5.4.6). Apart from that, the end result is quite acceptable. Except for a few characters that have been misreconstructed (like an "e" that turned into a "c"), no information have been lost.

During this experiment, several tries have been made in order to find the best Cluster Radius Factor (see section 5.4.4). The higher is this factor, the better compression will be achieves in term of ratios, but the more information is likely to be lost. A high factor means bigger clusters and different patterns are more likely to cluster together. The opposite is also true; the smaller is the factor, the less efficient is the compression but less information will be lost. For this experiment, the Cluster Radius Factor have been pushed as much as possible in order to get the best compression ratio and no perceivable loss of information (except for the broken pattern problem and the very occasional misreconstructed pattern). The Cluster Radius Factor used in the experiments is .30, but using a factor of .28 insures no loss of perceivable information (no misreconstructed pattern) and the test document get compressed down to 35k.



Short Notes 2171

JPN. J. APPL. PHYS. Vol. 18 (1979), No. 11

A Nondestructive Method for Measuring the Spatial Distribution of Minority Carrier Lifetime in Silicon Wafer

Yoichi MADA

Musashino Electrical Communication Laboratory, Nippon Telegraph and Telephone Public Corporation, Musashino-shi, Tokyo 180

(Received May 24, 1979)

The electrical behaviour of silicon devices is generally affected by lattice defects present in the base material. The minority carrier lifetime is an appropriate measure of the degree of perfection of silicon crystal, because this parameter is more sensitive to crystal quality than other parameters such as resistivity or mobility. As the lifetime will change during polishing, annealing, or device-fabrication processes, a method for measuring silicon wafer lifetime will be useful. To measure wafer lifetime, the method must be free from ohmic contacts so as to avoid lifetime change. Further, it must have high spatial resolution to check electrical inhomogeneities in the wafer. No method satisfying the above requirements has been developed up to now.

A technique using microwave absorption[1] seems most promising, because no ohmic contacts are needed. In this technique, the change in photoconductivity after injection of minority carriers by a light pulse is detected by the reflected microwave. Two methods have been proposed for coupling the microwave to the charge carriers in the sample. One[2] is where the sample is placed inside a waveguide and electrical coupling is obtained through the fundamental mode of the waveguide. In this method, the sample size is limited by the dimension of the waveguide used. In the other method,[3] electrical coupling is obtained through a horn antenna. This method overcomes the inconvenience of size limitation, but great care must be taken with the signal on account of the open nature of the arrangement. The above two methods cannot determine the spatial distribution of the lifetime in the sample.

To measure the minority carrier lifetime of a small part of a wafer, two coupling methods can be considered. The first is to excite minority carriers locally, using a focused spot of pulsed light. The second is to couple the microwave locally with minority carriers. The second method is more accurate, because the signal detecting area used in this second method does not change during measurement. In this work, a nondestructive method will be shown that can measure the spatial distribution of the minority carrier lifetime in silicon wafer using an S-band microwave.

The experimental arrangement for measurement of the lifetime is shown in Fig. 1. The microwave oscillator provides 2–4 GHz (S-band) microwave output at one milliwatt. The electromagnetic waves travel through a variable attenuator and circulator, and then reach the antenna. Electrical coupling between the electromagnetic waves and the charge carriers in silicon is brought about by this antenna. For injection of minority carriers, a xenon flash lamp that provides a light pulse of 2 μs in duration is used. The light is collected by lenses to a spot size of 3 mm on the sample. The microwave reflected through the circulator is measured by a crystal detector after amplification by a Watkins Johnson 5004 amplifier.

The antenna structure that produces effective coupling between electromagnetic waves and charge carriers in the sample is shown in Fig. 2. It is made of two fine copper wires of diameter 0.1 mm. Their input ends are connected directly to the coaxial cable propagating the microwave. Their output ends, which radiate electromagnetic waves towards the sample, are fixed on a platform in a straight line with distance D between each other. The platform is made of polystyrene to reduce microwave electrical loss. The distance D has a great effect on the signal gain. The optimum distance for maximum

Fig. 1. Lifetime measurement apparatus block-diagram.

Figure 26. Reconstructed Test Document after being compressed by Pattern Compression.

6.4 Cartesian Perceptual Compression

The Cartesian Perceptual Compression gives the best compression result: it compresses the test image down to 19K, a 55:1 ratio...

Jpn. J. Appl. Phys. Vol. 18 (1979), No. 11

A Nondestructive Method for Measuring the Spatial Distribution of Minority Carrier Lifetime in Silicon Wafer

Yoichi Mada

Musashino Electrical Communication Laboratory, Nippon Telegraph and Telephone Public Corporation, Musashino-shi, Tokyo 180

(Received May 24, 1979)

The electrical behaviour of silicon devices is generally affected by lattice defects present in the base material. The minority carrier lifetime is an appropriate measure of the degree of perfection of silicon crystal, because this parameter is more sensitive to crystal quality than other parameters such as resistivity or mobility. As the lifetime will change during polishing, annealing, or device-fabrication processes, a method for measuring silicon wafer lifetime will be useful. To measure wafer lifetime, the method must be free from ohmic contacts so as to avoid lifetime change. Further, it must have high spatial resolution to check electrical inhomogeneities in the wafer. No method satisfying the above requirements has been developed up to now.

A technique using microwave absorption[1] seems most promising, because no ohmic contacts are needed. In this technique, the change in photoconductivity after injection of minority carriers by a light pulse is detected by the reflected microwave. Two methods have been proposed for coupling the microwave to the charge carriers in the sample. One[2] is where the sample is placed inside a waveguide and electrical coupling is obtained through the fundamental mode of the waveguide. In this method, the sample size is limited by the dimension of the waveguide used. In the other method,[3] electrical coupling is obtained through a horn antenna. This method overcomes the inconvenience of size limitation, but great care must be taken with the signal on account of the open nature of the arrangement. The above two methods cannot determine the spatial distribution of the lifetime in the sample.

To measure the minority carrier lifetime of a small part of a wafer, two coupling methods can be considered. The first is to excite minority carriers locally, using a focused spot of pulsed light. The second is to couple the microwave locally with minority carriers. The second method is more accurate, because the signal detecting area used in this second method does not change during measurement. In this work, a nondestructive method will be shown that can measure the spatial distribution of the minority carrier lifetime in silicon wafer using an S-band microwave.

The experimental arrangement for measurement of the lifetime is shown in Fig. 1. The microwave oscillator provides 2-4 GHz (S-band) microwave output at one milliwatt. The electromagnetic waves travel through a variable attenuator and circulator, and then reach the antenna. Electrical coupling between the electromagnetic waves and the charge carriers in silicon is brought about by this antenna. For injection of minority carriers, a xenon flash lamp that provides a light pulse of 2 μs in duration is used. The light is collected by lenses to a spot size of 3 mm on the sample. The microwave reflected through the circulator is measured by a crystal detector after amplification by a Watkins Johnson 5004 amplifier.

The antenna structure that produces effective coupling between electromagnetic waves and charge carriers in the sample is shown in Fig. 2. It is made of two fine copper wires of diameter 0.1 mm. Their input ends are connected directly to the coaxial cable propagating the microwave. Their output ends, which radiate electromagnetic waves towards the sample, are fixed on a platform in a straight line with distance D between each other. The platform is made of polystyrene to reduce microwave electrical loss. The distance D has a great effect on the signal gain. The optimum distance for maximum

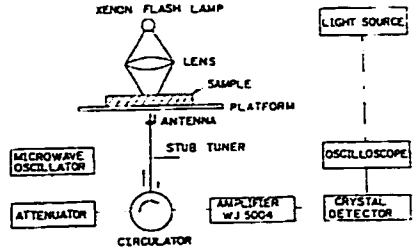


Fig. 1. Lifetime measurement apparatus block-diagram.

Figure 27. Reconstructed Test document after compression with CPC.

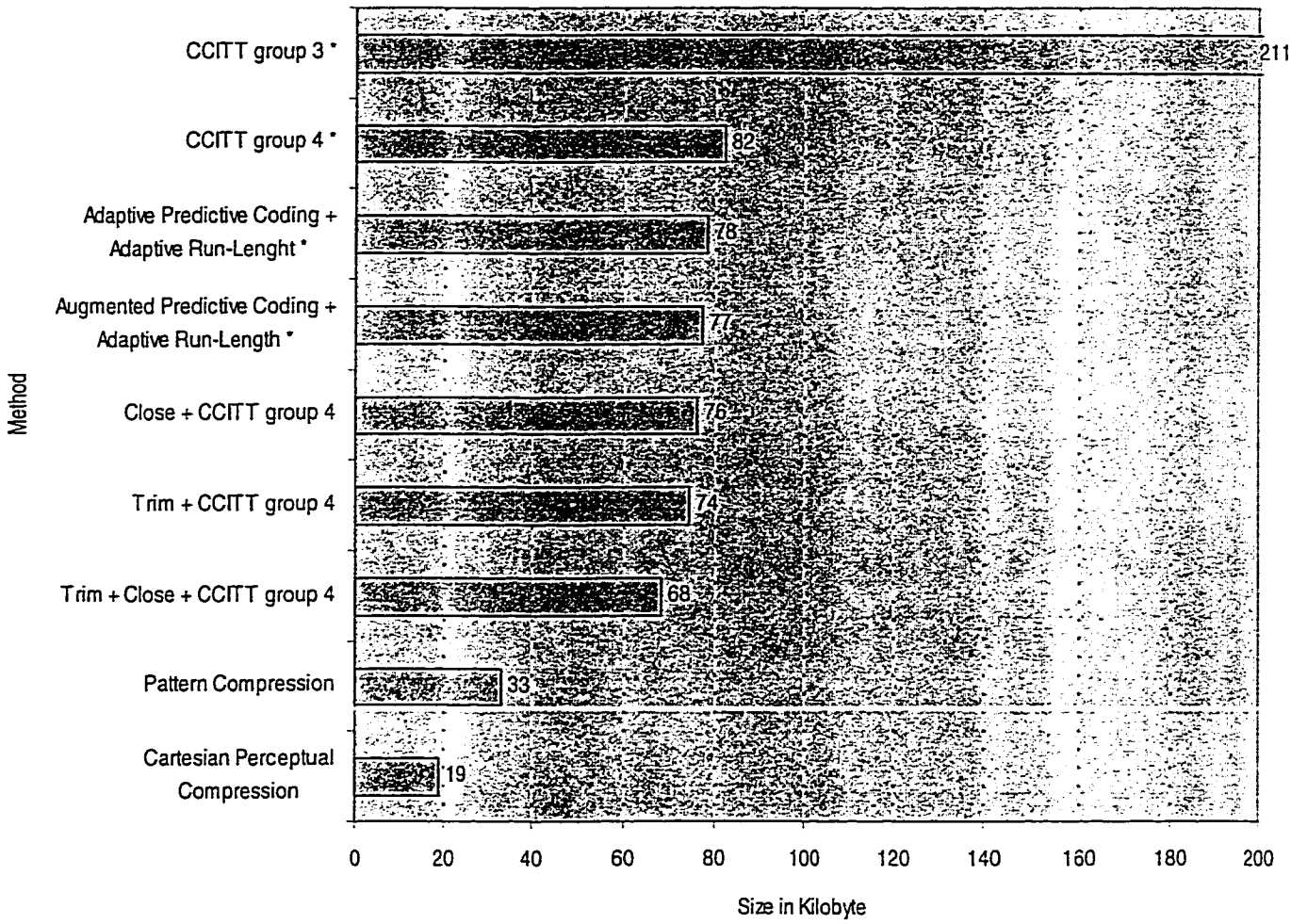


Figure 28. This figure resume the compression results obtained using the different methods (* non-lossy methods).

6.5 Statistics

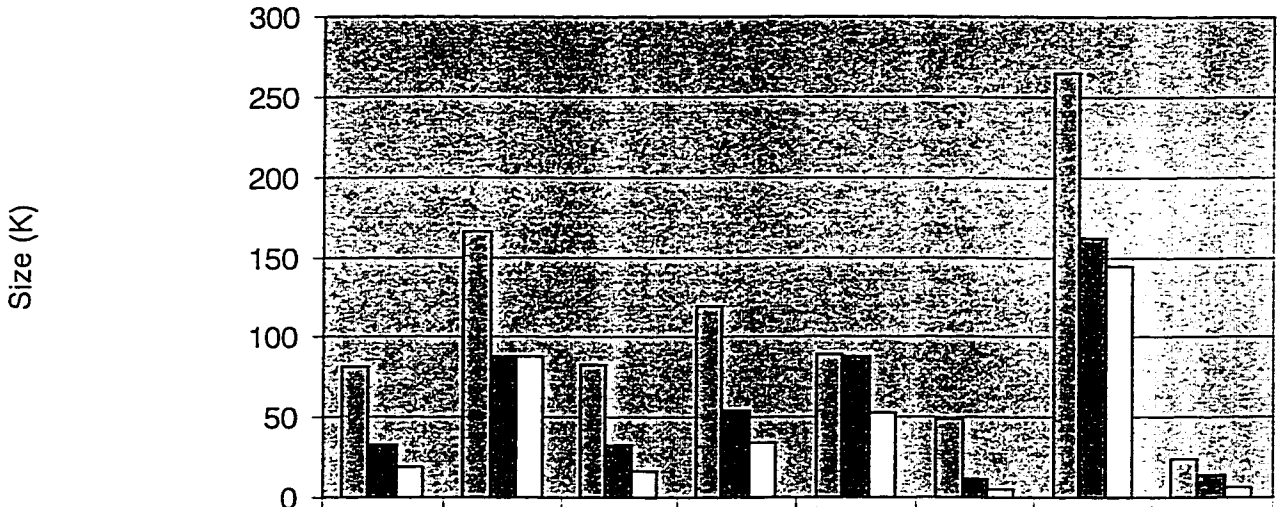
Two groups of testing have been done in order to compare the following three schemes: CCITT 2-Dimensional Compression [3.2], Pattern Compression and CPC [6]. The first group is made of 8 documents of 1 page only. The second group is made of 8 documents of 4 pages each. The goal of the second group is to show that Pattern Compression and CPC have even better compression ratio when a document has many pages. Both

schemes apply repetitive patterns on all the pages; they are not restricted to each individual page like the CCITT standard.

6.5.1 1-Page Documents

Here is a short description of each document:

- Test Document: The first document is the test document mentioned earlier in this chapter and in section 2.1.
- Doc1p01: The text is divided in three columns. A paragraph is boxed in a light gray background. The color of the text is not completely black since every letter is spotted with white pixels. A simple diagram lies at bottom of the page. A light contour of the page is visible.
- Doc1p02: This is a text only document. The text is divided in two columns. A contour of the page is present at the right and bottom of the page only.
- Doc1p03: The direction of the text is landscape. The text is divided in two columns. The first column represents the abstract of the article and is written in bold characters. The color of the text is not completely black. A light contour is present at the top and left side of the page.
- Doc1p04: This is a Japanese document. The text is divided in two columns. A few references on the bottom of the page are written in roman characters. A light gray contour is visible.
- Doc1p05: This is a text only document. The font is typewriter style. A small contour is present at the top and right side of the page.
- Doc1p06: This is a dithered picture.
- Doc1p07: This is an invoice type document. It was created by a software, not scanned from a paper.



	TestDoc	doc1p01	doc1p02	doc1p03	doc1p04	doc1p05	doc1p06	doc1p07
CCITT	82.024	165.682	83.066	119.872	89.634	49.298	264.29	24.057
Pattern Compression	33.222	88.231	32.747	54.179	87.61	11.304	161.809	14.107
CPC	19.419	87.827	16.359	33.299	52.243	4.797	144.137	6.83

Figure 29. 1-Page Document Compression Results

The average compression ratio for CCITT is 10:1, for Pattern Compression it's 17:1 and for CPC it's 23:1.

6.5.2 4-Pages Documents

Here is a small description of each document:

- Doc4p08: This is a medical article (magazine style) laid out in three columns. It has some pictures. The letters are not quite black.
- Doc4p09: This is an article on chemical textile. It has various figures and pictures. The letters are not quite black and some paragraphs are boxed in a light gray background.

- Doc4p10: The direction of this article is landscape. This article on biology has a text in two columns. It also have some figures and a page size table.
- Doc4p11: This is a chemical article with various figures and tables.
- Doc4p12: This is a biophysic article. It is mostly text with some figures.
- Doc4p13: This is a chemical article with various figures and tables.
- Doc4p14: This is an astro-physic article. The text font is typewriter style. The first page has been scanned crooked. The second page contains two big charts. Some formulas are in slimmer fonts and are really badly scanned.
- Doc4p15: This is an article on biology. The text is divided in two columns. It has a few figures.

	doc4p08	doc4p09	doc4p10	doc4p11	doc4p12	doc4p13	doc4p14	doc4p15
CCITT	562.984	721.937	416.614	208.495	322.151	216.586	261.652	269.04
Pattern Compression	282.8	317	145.3	79.7	105.5	86.7	137.4	79.8
CPC	229.252	357.794	99.513	48.909	54.562	45.456	108.439	42.962

Figure 30. 4-Pages Document Compression Results

The average compression ratio for CCITT is 10:1, the same as the 1-page document group. The average compression ratio for Pattern Compression is 23:1 and 30:1 for CPC.

When compared to the ratios obtained with the group of single page documents, CPC have increased its ratio by approximately 30%, and Pattern Compression by around 37%. This increase is due to two facts:

- Only one codebook needs to be created and applied to all the pages.
- In typical document imagery, the same set of characters will be used throughout the document.

It is more efficient to represent similar patterns that appear in different pages by only one codebook instead of repeating their representation in a codebook for each page. The only kind of multi pages document that would not make a difference in compression ratio if one codebook is applied to the whole document or one codebook is applied for each page is if every patterns in a page do not repeat in any other pages of the document.

The CCITT method does not benefit an increase of its ratio since its efficiency is independent to the size of the document.

Chapter 7. Conclusion

This project mainly contributes two things: a study of different compression techniques applied to document imagery that contrasts their effectiveness and the introduction of a new method that use a special approach to the problem. Definitely vector based compression proves to be the most effective approach, providing order of magnitude better compression ratio than any other methods.

The success of Pattern Compression can be attributed to mainly two factors: the method takes advantage of the kind of information document imagery is composed of; small patterns (i.e. characters). Secondly, some data may be lost in the process as long as it remains visually imperceptible. Vector compression makes this possible. The amount of lost information is very important and it is reflected in the Pattern Compression method; contrary to Vector Quantization [5.1], the average distortion is better controlled by viewing the problem not as finding the best representatives, but by creating the least number of representatives.

The Pattern Compression also has the advantage over other methods like CPC [6], of giving control over the algorithm parameters, which in turn have a direct impact on its effectiveness and the maximum distortion allowed. Other features contributing to the method success include the distinction made about border pixels, broken border pixel and non-border pixels; a change on a border pixel is not as visually perceptible than a change of a non-border one, so this is considered in the process. The alignment of patterns

according to their horizontal and vertical mass centers provides a quick and efficient way to match them properly.

7.1 Future Work

Unfortunately, the Pattern Compression method still need some more work. Here are some suggestions that would improve the method's efficiency:

- Create an algorithm that guarantees a minimal number of clusters when constructing the codebook.
  - Idea to study: an algorithm that uses a convex hull of the entire vectorial space in order to assign the first clusters to the vectors lying on the hull surface. Move the clusters in order to also encompass the maximum number of vectors inside the hull. Discard all the clusters and their vectors aside and repeat the same operation recursively until there is no more vector left in the space.
- The method could easily be improved on the time complexity, starting by implementing the method in a computer language a bit faster than Java.
- The implementation of the codebook and the input document could be changed in order to compress the document in one scan only, by building references between the patterns on the page and the clusters. Here is how this would work:
  - The codebook would be implemented as an array of clusters.
  - Each cluster would be made of one pattern (the representative) and a list of references (pointers) to the input page pointing to the patterns that are part of the cluster.
  - When you extract patterns you must also build a list of references to the associated clusters such that when the codebook is done, the list can be used directly to create the set of codeword representing the image. This prevents the need to rescan the image and finding the best representative for each pattern.
- Further study of the inverse density correction: this idea mentioned in section 5.4.6 do not solve the broken pattern problem completely but reduce it considerably.

- Ideas on compressing further the different parts of the compressed file:
  - CCITT group 4 [3.2] has been used to compress the codebook, but perhaps another method would compress it more.
  - The list of codewords representing the image (and perhaps the list of pixel distance) could perhaps be compressed further by using the LZW algorithm [7.1].

Vector Compression deserves to be researched more.

References

[1] The TIFF specification:

[1.1] *TIFF™ Revision 6.0 (Final – June 3, 1992)* by Aldus Corporation.

[1.2] Murray, James D., vanRyper, William, "*Encyclopedia of Graphics File Format*", 2nd edition, O'Reilly, pp. 880-908, April 1996

[2] Predictive Coding:

[2.1] Netravali, Arun N., Haskell, Barry G., "*Digital Pictures Representation & Compression*", New York, London: Plenum Press, pp. 164-167, 1988

[3] The CCITT specification:

[3.1] CCITT, Blue Book Volume VII - Fascicle VII.3, Recommendation T.4, "*Standardization of Group 3 facsimile apparatus for document transmission*", pp.21-39, 1989

[3.2] CCITT, Blue Book Volume VII - Fascicle VII.3, Recommendation T.6, "*FACSIMILE CODING SCHEMES AND CODING CONTROL FUNCTIONS FOR GROUP 4 FACSIMILE APPARATUS*", pp.48-57, 1988

[4] Morphological Transformation:

[4.1] Gonzalez, Rafael C., Woods, Richard E., "*Digital Image Processing*", Addison Wesley, pp. 518-569, 1993

[5] Vector Compression:

[5.1] Gray, Robert M., "*Vector Quantization*", IEEE ASSP Magazine, pp. 4-28, April 1984

[5.2] Linde, Y., Buzo, A., Gray, Robert M., "*An algorithm for vector quantizer design*", IEEE Transactions on Communications, COM-28, pp. 84-95, January 1980.

[6] Cartesian Perceptual Compression:

[6.1] The Cartesian web site: http://www.cartesianinc.com/

[6.2] Mark, Peter B., Shieber, Stuart M. (Inventors), "*Method and Apparatus for Compression of Images*", US Patent No. 5303313, Issued April 12, 1994.

[6.3] US Patent and Trademark web site: http://www.uspto.gov/

[7] Other References:

[7.1] Welch, Terry, *"A Technique for High-Performance Data Compression"*, Computer, June 1984

[7.2] Wallace, Gregory K. *"The JPEG Still Picture Compression Standard"*, Communications of the ACM, April 1991 (vol. 34 no.4), pp. 30-44.

Annexe A

A.1 TIFF

The documents used in this project were stored using the TIFF file format [1]. This format is presented in this section in some detail and should demonstrate that it is the file format of choice for this project.

The TIFF specification was originally designed by Aldus Corporation [1.1] as a standard method of storing black-and-white images created by scanners and desktop publishing applications. The TIFF format is perhaps the most versatile and diverse bitmap format in existence: multiple images can be stored in one file, various compression scheme are supported (uncompressed, Run-Length Encoding [1.1], LZW [1.1, 7.1], CCITT Group 3 and 4 [1.1, 3] and JPEG [1.1, 7.2]) and it is multi-platform. The following figure illustrates the logical organization of a TIFF file.

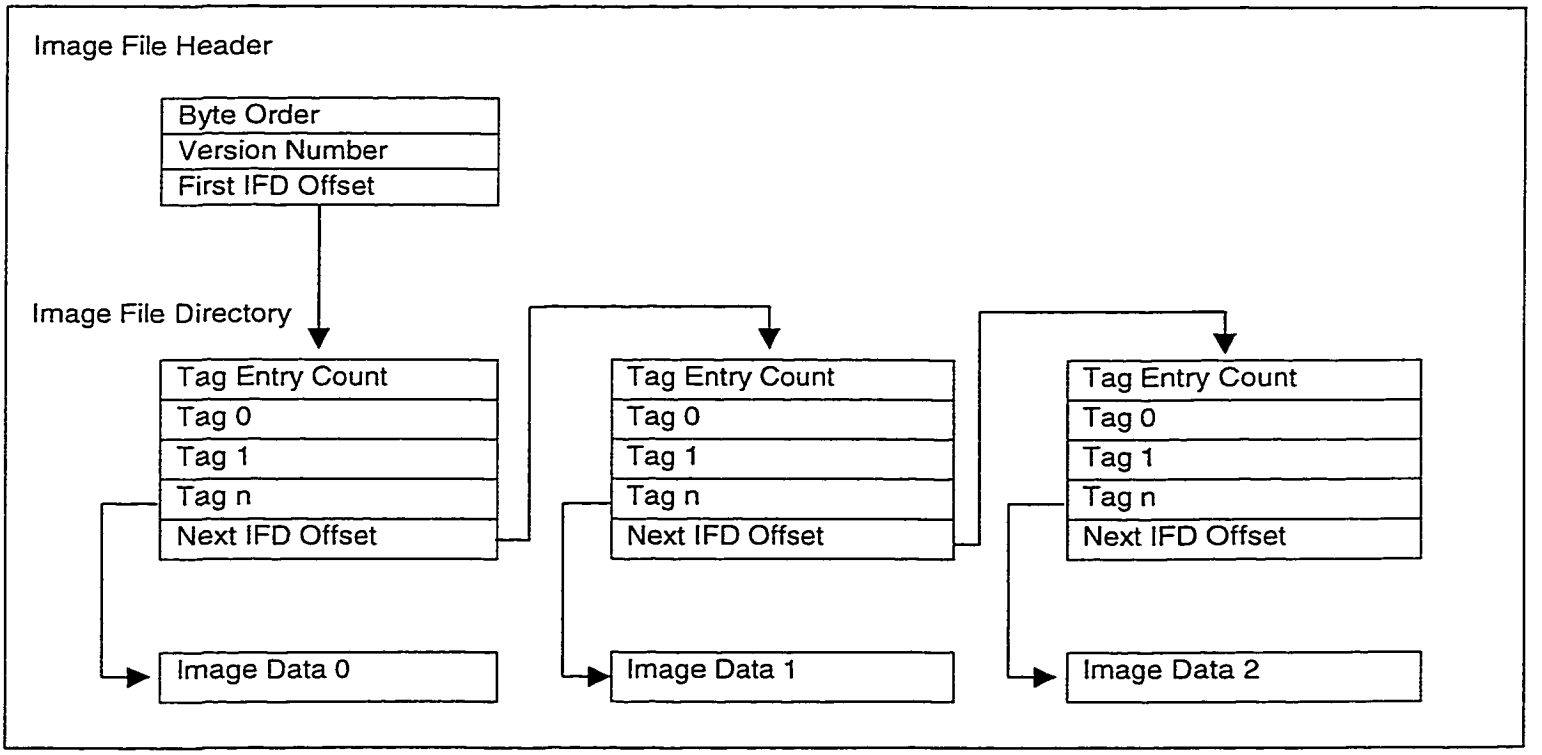


Figure 31. Logical Organization of a TIFF File

A.1.1 File Organization

TIFF files are organized into three sections: the Image File Header (IFH), the Image File Directory (IFD), and the bitmap data. A TIFF file that contains multiple images has one IFD and one bitmap per image stored.

A.1.2 Image File Header

The TIFF Image File Header (IFH) contains three fields of information: an Identifier, a Version number and an IFD Offset address.

The Identifier contains either the value 4949h (II) or 4D4Dh (MM). These values indicate whether the data in the TIFF file is written in little-endian (Intel format) or big-endian (Motorola format) order, respectively. All data encountered past the first two bytes in the file obey the byte-ordering scheme indicated by this field. These two values were chosen because they would always be the same, regardless of the byte order of the file.

The Version number is always 42, regardless of the TIFF revision, so it is regarded more as an identification number than a version number.

The IFD Offset is a 32-bit value that represents the offset position of the first Image File Directory in the TIFF file.

A.1.3 Image File Directory

An Image File Directory (IFD) is a collection of information similar to a header, and it is used to describe the bitmapped data to which it is attached. Like a header, it contains

information on the height, width, and depth of the image, the number of color planes, and the type of data compression used on the bitmapped data. Unlike a typical fixed header, however, an IFD is dynamic and may not only vary in size, but also may be found anywhere within the TIFF file. There may be more than one IFD contained within any file.

A TIFF file may contain any number of images, from zero on up. Each image is considered to be a separate subfile and has an IFD describing the bitmapped data. Each TIFF subfile can be written as a separate TIFF file or can be stored with other subfiles in a single TIFF file. Each subfile bitmap and IFD may reside anywhere in the TIFF file after the headers, and there may be only one IFD per image.

An IFD may vary in size, because it may contain a variable number of data records, called tags. Each tag contains a unique piece of information, just as fields do within a header.

An IFD has three parts: The first part is the field that contains the number of tags in the second part. The second part is an array of tags. The last field is the offset address to the next IFD.

A.1.4 Tags

A tag can be thought of as a data field in a file header. However, whereas a header data field may only contain data of a fixed size and is normally located only at a fixed position within a file header, a tag may contain, or point to, data that is any number of bytes in size and is located anywhere within an IFD.

A TIFF tag has the following structure: a Tag Id, a Data Type, a Data Count and a Data Offset.

The Tag Id is a numeric value identifying the type of information the tag contains. More specifically, the Tag Id indicates what the tag information represents. Typical information found in every TIFF file includes the height and width of the image, the depth of each pixel, and the type of data encoding used to compress the bitmap.

The Data Type contains a value indicating the scalar data type of the information found in the tag. The following values are supported: byte (8-bit unsigned integer), ascii (8-bit, NULL-terminated string), short (16-bit unsigned integer), long (32-bit unsigned integer) and rational (two 32-bit unsigned integers). The TIFF 6.0 revision added more new data types.

The Data Count indicates the number of items referenced by the tag.

The Data Offset is a 4-byte field that contains the offset location of the actual tag data within the TIFF file. If the tag data is four bytes or less in size, the data may be found in this field. If the tag data is greater than four bytes in size, then this field contains an offset to the position of the data in the TIFF file.

A.1.5 Organization of TIFF Tag Data

To keep developers from having to guess which tags should be written to a TIFF file and what tags are important to read, the TIFF specification defines the concept of baseline TIFF images. These baselines are defined by the type of image data they store: bi-level, gray-scale, palette-color, and full color. Each baseline has a minimum set of tags, which are required to appear in each type of TIFF file.

A.1.6 Compression

TIFF supports perhaps more types of data compression than any other image file format. It is also quite possible to use an unsupported compression scheme just by adding the needed user-defined tag. TIFF 4.0 supported only Run-Length Encoding (RLE) [1.1] and CCITT T.4 and T.6 compression [1.1, 3]. These compression schemes are typically only for use with 8-bit color and gray-scale, and 1-bit black-and-white images, respectively. TIFF 5.0 added the LZW compression scheme [1.1, 7.1], typically for color images, and TIFF 6.0 added the JPEG compression method [1.1, 7.2] for use with continuous-tone color and gray-scale images.