

# CELLULAR AUTOMATA BASED DOCUMENT COMPRESSION TECHNOLOGY FOR ON-LINE NETWORK TRANSMISSION\*

Chandrama Shaw Biplab K Sikdar N C Maiti

Dept. of Computer Science & Technology, Bengal Engineering College (D U), Howrah, India

Email: {cshaw, biplab, nm}@cs.becs.ac.in

## ABSTRACT

This paper reports an efficient document compression technology suitable for on-line network transmission. It identifies the different segments – such as text, image, and background within a scanned document. Three distinct compression techniques are developed around the regular structure of Cellular Automata (CA) for the compression of text, image and background segments to achieve better compression. As the low cost high speed VLSI implementation of CA is done, the proposed technology ideally suits for on-line network transmission of documents.

## 1. INTRODUCTION

The inter-network society has been experiencing an explosion of data communication. The voluminous data are transmitted as a document over communication channels. A high speed data transmission demands an efficient document compression technique to reduce the data rate. A document image often contains well defined regions of text, line graphics, images and background. For text and line graphics we need lossless compression [7], whereas for the images and background we can tolerate lossy reconstruction depending on the expected quality of a document image and the required bit rate.

In the past decades, a considerable interest has been shown in the development of Cellular Automata (CA) and its potential for modeling physical processes has been explored by a number of researchers [2, 3]. This fact motivates us to design a CA based compression technology for the documents. The proposed technology extracts the different segment blocks of pixels (text, image and background) from a scanned document. The blocks of each segment type are then compressed around a codebook [4], designed for that segment type. The compression is tuned to any limit between lossless to lossy. The domain knowledge of a segment type helps to determine the limit.

The major contributions of the present work are - (i) it develops technique to improve the classification capacity of CA, (ii) proposes CA based implementation of vector quantization technique (TSVQ), and (iii) reports a scheme to tune the TSVQ structure that suits for efficient document compression technology for on-line-transmission. The preliminaries of CA are next introduced.

## 2. CELLULAR AUTOMATA (CA)

The major contributions of the present work are - (i) it develops technique to improve the classification capacity of CA, (ii) proposes CA based implementation of vector quantization technique (TSVQ), and (iii) reports a scheme to tune the TSVQ structure that suits for efficient document compression technology for on-line transmission. The preliminaries of CA are next introduced.  $T[1]$  where

$$T[i][j] = \begin{cases} 1, & \text{the the next state of the } i\text{th cell depends on} \\ & \text{the present state of the } j\text{th cell} \\ 0, & \text{otherwise} \end{cases}$$

The state transition of such a CA is characterized by

$$S_{t+1} = T \times S_t + F$$

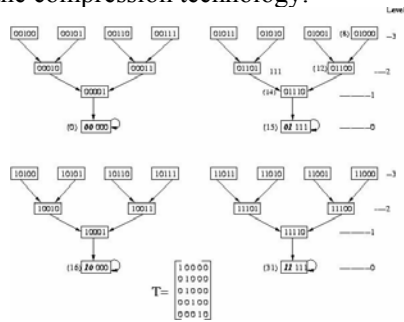
where  $S_t$  &  $S_{t+1}$  represent the CA states at  $t^{\text{th}}$  and  $(t+1)^{\text{th}}$  instance of time.

### 2.1 Non Group CA

For a non-group CA,  $\det[T]=0$ . The set of non-cyclic states in a non-group CA forms trees rooted at the cyclic states (attractors), where the leaf nodes are non-reachable states. The states rooted at attractor  $\alpha$  form  $\alpha$ -basin. The depth  $d$  is the number of steps required to reach the nearest cyclic state from a non-reachable state. In the example CA (T) of Fig. 1, there are four attractors (00000, 01111, 10000, and 11111). Its depth  $d = 3$ . The states (00100, 00101, ..., 00000) form the 00000-basin.

A non-group CA with multiple (single cycle) attractors is called MACA. The CA of Fig.1 is an MACA. In an  $n$ -cell MACA with  $k = 2^m$  attractors,

there exists 'm' bit positions, referred to as *PEF*, at which the attractors generate pseudo exhaustive  $2^m$  patterns. The MACA of Fig.1 has 4 attractors with two MSB as the PEF. The MACA is employed to design the compression technology.



**Figure 1** State transition diagram of a 5-cell non-group CA

### 3. THE DOCUMENT COMPRESSION TECHNOLOGY

The proposed document compression technology consists of two major components - segmentation and compression. The segmentation operation accepts a document containing text, images and backgrounds. A scanned document is first partitioned into, say, 16X16 block of pixels. Each block is characterized as one color block (the uniform background regions), two color block (text) or multi-color block (image). The two color blocks are compressed losslessly, whereas multi-color blocks are compressed employing lossy compression technique. For a border region block - that is, a block containing part of text, image or background, we consider it as the three color block.

A three color block with neighboring two color blocks is compressed losslessly.

For compression operation, we design the codebook for each type of blocks. The Vector Quantization (VQ) [4] method has been applied to generate the codebook. The following inherent drawbacks of VQ:

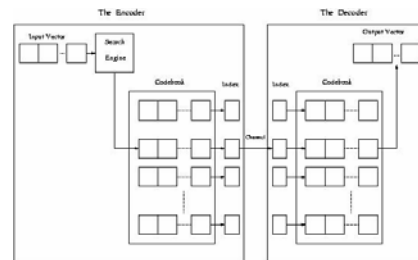
- difficulty in generation of codebook that would ensure good quality of image with high compression ratio; and
- the high processing overhead at encoding stage are addressed in the proposed CA based design.

Rather than developing general compression scheme, we concentrate on specific type of document (say encyclopedia) to improve the quality of compression as well as the compression ration. While designing a codebook for a type of blocks, the scheme extracts the domain knowledge from the training set of document files.

The encoding overhead of a VQ based technique is reduced substantially by employing the CA technology. The CA acts as the implicit memory to store the codebook and to search for the best match of an input data block.

### 3.1 Codebook

Each vector of a codebook is known as the codevector or codeword. A cluster is defined as the set of vectors having minimum deviation from a codevector. Thus a codevector is the nearest neighbor of the vectors in a cluster.



**Figure 2** Encoder and Decoder

The proposed compression technique encodes each block (16X16, 8X8, 4X4, and so on) of input files to be compressed with the index of a code vector in the codebook (*Fig.2, encoder*). The index is then sent to the destination end. The decoder (*Fig.2, decoder*) at the destination replaces the entry associated with the index.

The codebook is kept at the destination site. However, the codebook at the encoder end is replaced by an MACA. It eliminates the requirement for memory to store the codebook as well as it implements faster searching in codebooks.

For the current application, separate codebooks are designed for text, image and background blocks. For the image blocks, we design three layer codebooks based on TSVQ [4] technique. We consider block size 16X16, 8X8, and 4X4. For the text blocks, the five layer codebooks are designed.

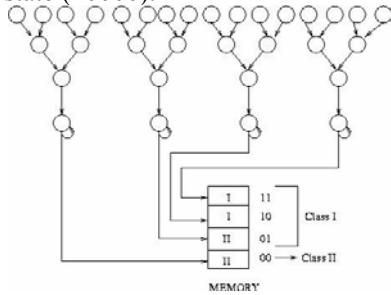
During compression of a document, whenever a match between a higher dimension block (say 16X16) and the entries of corresponding codebook fails, the search technique considers successive lower dimension (8X8, 4X4, 2X2, or so on) blocks and tries to find out the best match in the lower dimension codebook within a *threshold limit*.

#### 3.1.1 Design of Codebook

The purpose of a codebook is - while a block (vector) from an input data file arrives, it is compared with the codevectors of the codebook and results in an index representing the codevector. We

implement this structure with MACA based 2-class classifier.

The example MACA of Fig.1 is a 2-class classifier, where Class I is represented by the set of attractor basins  $[I] = 01111, 10000$  &  $11111$  and for Class II, it is  $[II] = 00000$ . All the patterns of  $[I]$  belong to Class I. When the CA is loaded with an input pattern (say 10100) and is allowed to run for a number (3) of cycles (the depth of the MACA), it reaches to an attractor state (10000).



**Figure 3** MACA based classification strategy

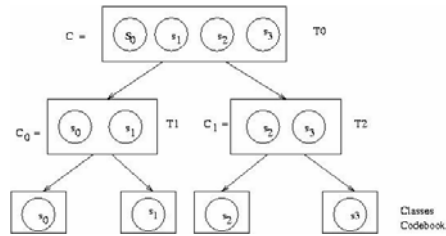
The pseudo-exhaustive fields (PEF = 10) of the attractor (10000) identifies the class (Class I) of the patterns (101000). The PEF yields the memory address where the class information is stored.

Fig.4 illustrates the design of a set of MACA for the codebook. Suppose, we want to classify the codevectors  $C = \{\{S_0\}, \{S_1\}, \{S_2\}, \{S_3\}\}$  into four classes so that the classifier would output the correct class  $i$  ( $i = 0, 1, 2, 3$ ), that is the index of a codevector, for a given input codevector  $P_i$ . At the first level, is divided into 2-classes -  $C_0$  and  $C_1$ , where  $C_0 = \{\{S_0\}, \{S_1\}\}$  and  $C_1 = \{\{S_2\}, \{S_3\}\}$ . The MACA ( $T_0$ ) is designed for this classification (Fig. 4). The similar process is then applied for  $C_0$  and  $C_1$  to isolate  $\{S_0\}, \{S_1\}$  and  $\{S_2\}, \{S_3\}$  respectively. It results in a multi-class classifier with 3 2-class-classifiers –  $T_0, T_1, T_2$ . Thus the logical structure of the multi-class classifier (Fig.4) is equivalent to the TSVQ, representing a codebook [4].

### 3.1.2 Searching in codebook

For a given codevector  $P_j$  ( $P_j \in S_1$ ) we need to identify the codebook entry closest to  $P_j$ . If the MACA ( $T_0$ ) of first level is loaded with  $P_j$  and allowed to run, it returns the class  $C_0$ . The classifier of the next level - that is,  $T_1$  is then loaded with  $P_j$  which outputs the class of  $P_j$  as  $S_1$ . In order to identify the best match in a TSVQ scheme, the input vector is compared with the centroids of two vector clusters in each layer of the tree (a vector cluster represents a set of entries in a codebook). A sequence of comparisons are to be done in the subsequent levels till the leaf node is reached.

A GA (genetic algorithm) based scheme [8] has been formulated to design the multi-class classifier, based on two class classifiers, for the vectors in a codebook. The GA framework outputs the set of MACA (T-matrices).



**Figure 4** The multi-class classifier equivalent to TSVQ

### 3.2 Requirements for the design

In the testing phase, when a vector say  $V = \begin{pmatrix} 0001 \\ 0011 \\ 0111 \\ 1111 \end{pmatrix}$

approaches for its best match, it always gets attracted to the class that has centroids with lesser hamming distances [8] from the vector itself. If

Class1	ClassII
Centroid	Centroid
0001	1001
0011	1010
0111	1100
1111	0100

are the centroids, then it is expected that the vector  $V$  will fall in Class I.

Though it is possible to train the classifier in such a way that the  $V$  should fall in Class II, the inherent tendency of a classifier, in the testing phase, is to follow the hamming distance. That is, the hamming distance is the determining metric in CA space which supplements euclidian distance (as in TSVQ based search) in real space. However, the vectors representing the real life data of a class are more prone to their euclidean distance. For example, 7 (0111) and 8 (1000) are close to each other if we consider the euclidean distance, but according to hamming distance (4) these two are far away. We realize discretization on real life data so that it maintains the minimum hamming distance if the vectors belong to the same class. For Example, let assume  $c1 = \{12, 7, -2, 6, 9\}$ , and  $c2 = \{3, 18, 10, 26, -3\}$  are the centroids of Class I and Class II. The mean centroid  $c = (c1+c2)/2 = \{8, 13, 4, 16, 3\}$ . If  $V = \{10, 10, 6, 19, 1\}$ , then after discretization  $V$  becomes  $\{1, 0, 1, 1, 0\}$ . The discretization process improves the classification capacity at the cost of little more processing overhead.

## 4. EXPERIMENTAL RESULTS

For designing the codebook, 10 scanned documents of a specific class, that consist of text, images and background, are considered as the training set. For testing, we start with the set of gray level documents. The experiment is done on Intel P-IV, 1.2 MHz platform.

**Table 1** Execution Time (In Milli Seconds)

Block Size	Full Search	TSVQ	MACA
4X4	0.0121	0.00824	0.00562
8X8	0.0473	0.03312	0.01367
16X16	0.1941	0.13192	0.04102

The efficiency of the MACA based search engine is compared with that of the full search and TSVQ [4] in Table I. Column I of Table I depict the size of the block. The columns II, III, and IV report the average search time for an input block to find its closest codevector. The results of Table I confirm that the search time in the proposed MACA based classifier (software version) is significantly lesser than that of full search and TSVQ. However, the major gain could be derived through hardwired implementation of MACA tree as proposed in [1].

In the proposed technology, the compression algorithm sets different threshold values to find the best match in a codebook while performing lossy compression. It affects the compression ratio as well as the quality of decompressed document. Table II notes the effect of threshold value chosen on the compression ratio (CR) and the quality of compression (PSNR). The compression ratio and the PSNR for three different threshold values are shown.

The efficiency of the current design is compared with the JPEG 2000 [9]. Table III reports the comparative performance in term of compression ratio and the PSNR. The results reported in Table III establish that the CA based document compression technology achieves higher PSNR than JPEG 2000 for the same compression ratio.

An efficient segmentation technique improves the quality of the proposed compression. We have employed the simple segmentation algorithm for the current design. The misclassification errors due to segmentation, may affect the compression ratio of the document while the image of a document is segmented as text block. The scheme employs lossless compression on that block. Further, the scheme employs lossy compression on a block segmented as image which could be come from the text or background portion of the document. In effect we loose the quality of compression.

**Table 2** Compression ratio and PSNR for different threshold

Name	Threshold 1		Threshold 2		Threshold 3	
	CR	PSNR	CR	PSNR	CR	PSNR
doc 1	80.12	28.33	89.99	26.30	94.69	24.50
doc 2	78.23	30.33	91.12	25.23	92.63	25.12
doc 3	84.89	28.10	90.37	27.54	96.33	23.10
doc 4	76.34	28.34	91.12	26.45	95.60	23.56
doc 5	75.34	29.43	89.99	26.61	94.66	24.80

**Table 3** Comparison between CA based document compression and jpeg2000

Document title	Compression Ratio (%)		PSNR (db)	
	CA	JPEG2K	CA	JPEG2K
document1	89.50	89.53	45.55	44.54
document1	74.42	74.51	45.24	43.35
document1	85.60	85.60	44.99	43.62
document1	62.50	62.22	45.47	43.67
document1	74.98	74.84	45.50	43.34
document1	71.95	71.95	44.35	43.06
document1	75.65	75.66	44.62	43.64

## 5. CONCLUSION

The CA based document compression technology proposed in this paper is a unique in application of CA. The sparse network of CA can be employed for low cost hardware implementation of the compression technique to support efficient on line network transmission of document.

## REFERENCES

- [1] P. P. Chaudhuri et al., "Additive cellular automata, theory and applications, vol. 1," *IEEE Computer Society Press, California*, ISBN-08186-7717-1, 1997.
- [2] S. Wolfram, "Theory and application of cellular automata" *World Scientific*, 1986.
- [3] S. Wolfram, "A New Kind Of Science," *World Scientific*, 2002.
- [4] Allen Gresho et al., "*Vector Quantization and Signal Compression*", Kluwer Academy Publishers.
- [5] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression", *IEEE Trans. on Infor. Theory* IT-23 (1977) 337-343.
- [6] N. Ganguly et al., "Evolving Cellular Automata as Pattern Classifier", Proceedings of 5<sup>th</sup> International Conference on Cellular Automata for Research and Industry, ACRI 2002, Switzerland.
- [7] Diego Santa-Cruz et al., "JPEG 2000 still image coding versus other standards", In SPIE's 46th annual meeting, Applications of Digital Image Processing XXIV, Proc. of SPIE, volume 4472, pages 267-275, San Diego, California, Jul. 29-Aug. 3, 2001.