LOSSLESS COMPRESSION OF IMAGES USING LOGIC MINIMIZATION

Anil Kumar Chaudhary, Jacob Augustine and James Jacob

Department of Electrical Communication Engineering
Indian Institute of Science, Bangalore 560 012 INDIA.
E-mail: james@ece.iisc.ernet.in

ABSTRACT

A novel approach for the lossless compression of images is presented. After preprocessing, the image is split into bit planes which are then divided into smaller blocks. Mixed blocks are converted to Boolean switching functions and then subjected to minimization to arrive at a compact representation, leading to possible data compression. Blocks are then classified into distinct events based on the outcome of logic minimization. A Huffman code is constructed and the blocks are encoded suitably. This approach provides a framework to incorporate various possible switching theoretic techniques into the basic coding scheme proposed by us. This paper also shows that logic minimization can be used to handle blocks of larger size, than practically possible with Huffman coding, to yield significant compression on gray-level images. Our approach compares well with JPEG in terms of compression ratio.

1. INTRODUCTION

Kunt and Johnsen [1] have proposed a lossless compression technique called **Block Coding** for binary images and have subsequently extended to gray-level images, by applying it on the bit planes. Pixel values of the image are replaced by equivalent Gray codes to reduce transitions on the bit planes. In *Block coding,* each bit plane is divided into smaller blocks of size $n \times m$ which are classified into three types namely, **all-white, all-black,** and **mixed** and encoded using prefix codes '0', '11', and '10' respectively. Codeword of a *mixed* block is obtained by the nm bits of the block, preceded by the prefix '10'. It is possible to achieve better compression by entropy coding of the *mixed* blocks. The 2^{nm} different possible bit patterns of the block can be considered as source messages and coded using a variable length code such as Huffman [2]. By making n and m as large as possible, better result can be obtained with Huffman coding [1, 3]. However, for large alphabet size the design and implementation of Huffman code is complicated as it requires the measurement of 2^{nm} probabilities and table look-up involving a dictionary of possibly large size. In [1], the *mixed* blocks are Huffman coded when their size is small enough (3×3 or smaller), otherwise transmitted as they are. The *mixed* blocks have also been arithmetic coded to increase the compression ratio in the case of binary images [4].

In this paper we propose a scheme wherein the *mixed* blocks are classified into many categories according to the outcome of logic minimization. Huffman codes are used to indicate various types of blocks, which makes the **logic coding** [5] scheme proposed by us more efficient. It may also be observed that logic minimization can be used to handle blocks of larger size than practically possible with Huffman coding, and in combination with block coding can yield significant compression on gray-level images. The probability measurement is done on a much smaller set of events rather than on 2^{nm} events.

2. BACKGROUND

We define few terms [6, 7] required for explaining our compression scheme. A Boolean **switching function** $\mathbf{F}$ is a mapping $\mathbf{F} : \mathbf{B}^N \rightarrow \mathbf{B}$, where $\mathbf{B} = \{0, 1\}$. In the truth table of a switching function of N variables, there are 2^N rows. Each of these rows which represents an input state vector is called a **minterm.** In a switching function, the **ON-set** is the set of minterms whose outputs are mapped to 1 and the **OFF-set** is the set of minterms whose outputs are mapped to 0. A **cube** is an N-tuple $\mathbf{A} = \{a_1, a_2, ..., a_N\}$, where element $a_i \in \{0, 1, X\}$. **Dimension** of a cube is the number of Xs in it. An α cube has 2^α minterms (zero cubes) within it. Cube **A subsumes** cube **B** (denoted as $\mathbf{A} \sqsubseteq \mathbf{B}$), if all minterms of **A** are contained in **B**. We define *compression efficiency* or **compression ratio** as,

$$\frac{\textit{total input bytes} - \textit{total output bytes}}{\textit{total input bytes}} \times 100\% \quad (1)$$

In our earlier work [5] the possibility of compressing *mixed* blocks using logic minimization was demon-

0-7803-3258-X/96/$5.00 © 1996 IEEE

strated. Results obtained using the technique on gray-level images were comparable to that of the lossless mode of JPEG [8]. Our approach consisted of recoding and Gray coding the intensity levels, bit plane decomposition, and *logic coding*. Recoding consists of arranging the intensity values in an image contiguously and Gray coding substitutes the recoded intensity values by their equivalent Gray codes. The image is then decomposed into its individual bit planes and each bit plane is divided into smaller blocks of size $n \times m$. In our switching theoretic approach, a *mixed* block is converted to a Boolean switching function of N variables ($N = log_2 nm$) by treating the binary values of the nm pixels as the output of the function. We have chosen integer powers of 2 as values for n and m. Truth table of the switching function for a block is generated by assigning the pixels to minterms according to Gray code, such that geometrically adjacent pixels are mapped to logically adjacent minterms. This assignment helps in minimization since any cluster of 2^α logically adjacent minterms combine to form a single α-cube. A pixel in an $n \times m$ block has N logically adjacent pixels. Blocks are scanned row wise with a reversal of the direction for adjacent rows. Then each function is minimized using a two-level logic minimizer such as ESPRESSO [6] and if minimization results in compression, the minimal two-level sum-of-products form of the function is encoded to generate the compressed image. Decoding of *logic coded* blocks can be done using the cube *subsuming* operation [5]. Decoding is relatively simple and consists of recovering the encoded cubes/minterms corresponding to a block and expanding the function back into its truth table form.

3. HUFFMAN CODING OF THE SWITCHING THEORETIC EVENTS

In the present scheme we further classify *mixed* blocks on the basis of the result of logic minimization. Events and the bounds of bits required to code them, excluding the Huffman code for these events are given in Table 1. The experiment was restricted to block sizes of 8×4 and 4×8. In this case $N = log_2 32 = 5$ and in general, the blocks reducible to 3 or less cubes as well as those with 4 or less minterms are logically compressible. We have considered a total of 17 events as indicated in Table 1. They are, *all-white* blocks, *all-black* blocks, blocks whose ON-set can be compressed to 1, 2, 3 cubes or 1, 2, 3, 4 minterms, blocks whose OFF-set can be compressed to 1, 2, 3 cubes or 1, 2, 3, 4 minterms and the incompressible blocks. A Huffman code is constructed for these 17 events based on the statistics and blocks are encoded suitably.

Table 1: Switching theoretic events and bit requirement.

Event #	Switching theoretic event	Lower bound	Upper bound
1	all-white block	0	0
2	all-black block	0	0
	mixed blocks logically compressible to		
3	1 cube (ON-set)	N	$1.67N$
4	2 cubes (ON-set)	$2N$	$2 * 1.67N$
5	3 cubes (ON-set)	$3N$	$3 * 1.67N$
6	1 minterm (ON-set)	N	N
7	2 minterms(ON-set)	$2N$	$2N$
8	3 minterms (ON-set)	$3N$	$3N$
9	4 minterms (ON-set)	$4N$	$4N$
10	1 cube (OFF-set)	N	$1.67N$
11	2 cubes (OFF-set)	$2N$	$2 * 1.67N$
12	3 cubes (OFF-set)	$3N$	$3 * 1.67N$
13	1 minterm (OFF-set)	N	N
14	2 minterms (OFF-set)	$2N$	$2N$
15	3 minterms (OFF-set)	$3N$	$3N$
16	4 minterms (OFF-set)	$4N$	$4N$
17	mixed blocks, logically incompressible	nm	nm

In the case of events where the minimized ON/OFF sets have 1, 2 or 3, cubes, a code set {0, 10, 11} which satisfies the prefix property is used for the set of cube symbols {0, 1, X} by alloting the one bit code to the symbol with maximum frequency of occurrence on a given bit plane. In the worst case (equally likely symbols 0, 1 and X) the average length of the prefix code is $\frac{5}{3}$ bits/symbol. Bits required for the representation of an N variable cube is upper bounded by $1.67N$ or $1.67 log_2 nm$. A cube can be coded using N bits when there are only two symbols (1 bit/symbol), which corresponds to the lower bound. When there are more cubes, these bounds should be multiplied by the corresponding number. Rows 3 and 10, 4 and 11, 5 and 12, of Table 1 show the bits required in the cases of 1, 2, and 3 cubes respectively. Minterms have only two symbols (0 and 1) and can be coded using 1 bit/symbol. Bits required for the cases of minterms are shown in the rows 6, 7, 8, 9, 13, 14, 15, and 16 of Table 1. Bits required for an incompressible block is nm as shown in the last row.

3.1. Format of the Compressed Image

Each encoded bit plane has the structure shown in Figure 1. First three bit field denotes the **bit plane type** as indicated below:

000 : *all-white* bit plane.

001 : *all-black* bit plane.

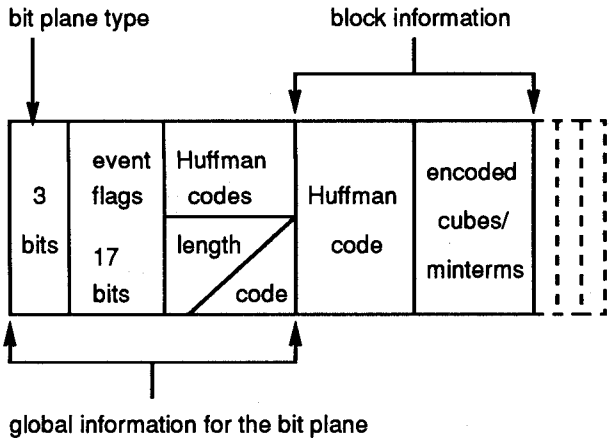

Figure 1: Format of the compressed bit plane.

010 : incompressible bit plane.

100 : logically compressible bit plane with code allotment for cube symbols as,
$0 \rightarrow 0$, $10 \rightarrow 1$, $11 \rightarrow$ X.

101 : logically compressible bit plane with code allotment for cube symbols as,
$0 \rightarrow 1$, $10 \rightarrow$ X, $11 \rightarrow 0$.

110 : logically compressible bit plane with code allotment for cube symbols as,
$0 \rightarrow$ X, $10 \rightarrow 0$, $11 \rightarrow 1$.

Next 17 bits called **event flags** are used to indicate the presence/absence of a given event type, as it is possible that some of them never occur in a given bit plane. Then we store the **Huffman code** information required for decoding. This consists of the code length followed by the Huffman code alloted to the events (from the set of 17 possible events) that occur on a given bit plane. This is followed by the **block information** for each block, which comprises of the Huffman code indicating the block type and the encoded cubes (if any) appended to it.

4. EXPERIMENTAL RESULTS

The compression and decompression techniques have been implemented in 'C' on an IBM RS-6000/580 workstation running under UNIX, and tested on standard images, *lena, baboon, boats* (nic.funet.fi: /pub/graphics/misc /test-images) and *girl* (eedsp.gatech.edu: /database /images), of subsampled (picking alternate pixels in both directions) size 256 × 256 pixels and 8 bits/pixel. We have used a fixed block size of 8 × 4 (4 × 8 for *boats*) for all the bit planes of an image to carry out the experiment. Preprocessing step of recoding and Gray coding of the pixel values are applied before splitting each image to its individual bit planes.

Table 2: Statistics of block types for *lena*.

#	bit plane	s7	s6	s5	s4	s3
1	all-white	14.1	41.2	17.9	7.4	0.9
2	all-black	51.0	12.3	19.2	14.3	2.5
	mixed blocks with ON-set compressed to					
3	1 cube	1.3	1.7	1.6	0.7	0.2
4	2 cubes	2.2	3.7	3.2	3.1	1.4
5	3 cubes	1.7	2.2	4.2	3.3	3.2
6	1 minterm	1.8	3.6	3.4	2.6	0.6
7	2 minterms	0.7	1.2	1.2	1.5	0.8
8	3 minterms	0.2	0.2	0.6	0.6	0.4
9	4 minterms	0.2	0.4	0.2	0.3	0.3
	mixed blocks with OFF-set compressed to					
10	1 cube	3.1	1.6	1.9	1.5	0.5
11	2 cubes	4.0	3.5	4.2	2.6	1.9
12	3 cubes	4.6	4.6	5.9	4.3	3.5
13	1 minterm	4.1	2.7	3.3	2.8	2.5
14	2 minterms	1.0	0.5	1.9	1.6	1.4
15	3 minterms	0.4	0.4	0.6	0.3	1.0
16	4 minterms	0.2	0.1	0.0	0.2	0.5
17	mixed logically incompressible	9.4	20.1	30.7	52.3	78.6
18	mixed logically compressible	25.5	26.4	32.2	26.0	18.0
19	compression ratio	71.2	59.3	44.6	27.5	6.6
20	Overall compression ratio (%) (assuming expanding bit planes are stored as they are)				26.1	

Entries in the table are percentages (%).

Table 2 gives the statistics of various types of blocks on the bit planes of *lena*. The MSB plane is s7. Rows 1 to 17 give the percentages of the seventeen events indicated. Row 18 gives the total percentage of the various types of logically compressible blocks, which is the sum of entries in rows 3 to 16. Sum of the values in rows 1, 2, 17 and 18 should add up to 100. Total compression ratio obtained on each bit plane is given in row 19. Compression ratio for the entire image is given in the last row. It may be observed that *mixed* blocks constitute a large portion and hence our attempt to compress such blocks is justified. Also the occurrence of different types of logically compressible blocks justify the usage of Huffman codes, to efficiently indicate their type. Note that the least 3 bit planes (s2, s1 and s0) are stored as they are since they remain incompressible by our method and no data is given in Table 2 for them.

Table 3 gives the results of the compression experiment as well as a comparison with the lossless mode of JPEG. We have used the PVRG-JPEG (Portable Video Research Group at Stanford) Codec1.1 (have-

Table 3: Results for *logic coding* with Huffman header.

Image		lena	baboon	boats	girl
	block size	8 × 4	8 × 4	4 × 8	8 × 4
Logic coding	compression ratio (%)	26.1	12.6	29.3	41.0
	compression time (sec.)	285.1	394.2	360.0	259.2
	decompression time (sec.)	1.7	1.3	2.0	1.7
PVRG-JPEG	Compression ratio (%)	22.7 - 29.4	6.7 - 13.7	24.9 - 33.1	30.7 - 38.1
lossless	compression time (sec.)	0.4	0.4	0.4	0.4
mode	decompression time (sec.)	0.3	0.3	0.3	0.3

Compression and decompression CPU time in seconds is on IBM RS-6000/580.

fun.stanford.edu: /pub/jpeg/JPEGv1.2.tar.Z). JPEG was run with all the 7 predictors available and the best and worst results are reported in Table 3. Compression ratios obtained by *logic coding* are comparable to those obtained by JPEG. The best result for *logic coding* is superior to the best of JPEG result for *girl* , while for the remaining images our results are within 3.8% of the best JPEG result, though we have not employed any decorrelation scheme as done in JPEG. Average compression ratio for the 4 images is 27.3 % by this approach whereas it is 28.6 % using JPEG with the most suitable predictor for each image.

CPU time reported for the compression scheme is presently rather high. It is primarily due to the fact that ESPRESSO is used as a stand alone package and considerable communication and file manipulation overheads are incurred. As the cube subsuming operation used in decompression is relatively simple, the time required to reconstruct the *logic coded* blocks is significantly less and the total decompression time is comparable to that of JPEG. Improving the time performance, however, has not been attempted in this work.

5. CONCLUSIONS

Contribution of this work is primarily two fold. Firstly, it suggests a practical solution to the problem of handling large alphabet size in Huffman coding of the picture blocks, by translating a group of events to a single event. For example, there may be a large number of blocks that are minimizable to a single cube, which is treated as a single event. Secondly, it provides a possible direction for further improvement, by incorporating other switching theoretic techniques for compressing the blocks, logically incompressible according to the present scheme. It is well known that the minimum two-level sum-of-products description is not the most compact representation for most switching functions. Alternative representations such as Binary Decision Diagrams (BDDs) [9] or multilevel logic forms [7] can lead to more compact representations. One can explore the different possible representations of the function corresponding to each block and encode the optimal representation. We hope that such an approach will ultimately result in a scheme better than the currently available lossless compression schemes.

6. REFERENCES

[1] M. Kunt and O. Johnsen, "Block Coding of Graphics: A Tutorial Review," *Proc. IEEE*, vol. 68, no. 7, pp. 770-786, July 1980.

[2] D. A. Huffman, "A Method for the Construction of Minimum Redundancy Codes," *Proc. IRE*, vol. 40, no. 10, pp. 1098-1101, September 1952.

[3] B. Y. Kavalerchik, "Generalized Block Coding of Black and White Images," *IEEE Trans. on Image Processing*, vol. 1, no. 4, pp. 518-520, October 1992.

[4] P. Fränti, "A Fast and Efficient Compression Method for Binary Images," *Signal Processing: Image Communication (Elsevier Science)*, pp. 69-76, 6(1994),

[5] J. Augustine, W. Feng, A. Makur, and J. Jacob, " Switching Theoretic Approach to Image Compression," *Signal Processing (Elsevier Science)*, vol. 44, pp. 243-246, June 1995.

[6] R. K. Brayton, G. D. Hachtel, C. T. McMullen, and A. L. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer Academic Publishers, Boston, 1984.

[7] S. Devadas, A. Ghosh, and K. Keutzer, *Logic Synthesis*, McGraw-Hill Inc., New York, 1994.

[8] W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Compression Standard*, Van Nostrand Reinhold, New York, 1993.

[9] R. E. Bryant, "Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams," *ACM Computing Surveys*, vol. 24, no. 3, pp. 293-318, September 1992.