ISSN:0975-9646

Use of Local Minimization for Lossless Gray Image Compression

Narendra Kumar[1], Dr. Sachin Gupta[2]
[1]Computer Science & engineering , 1,Knowledge park 2, Greater Noida Nkteotia2004@gmail.com,
[2]IT Deptt, RKGIT, Ghaziabad, imsachingupta@rediffmail.com

Abstract

A novel approach for the lossless compression of gray images is presented. A prediction process is performed followed by the mapping of prediction residuals. The prediction residuals are then split into bit–planes. Two-dimensional (2D) differencing operation is applied to bit-planes prior to segmentation and classification. Performing an Exclusive-OR logic operation between neighboring pixels in the bit planes creates the difference image. The difference image can be coded more efficiently than the original image whenever the average run length of black pixels in the original image is greater than two. The 2d difference bit-plane is divided in to windows or block of size 16*16 pixels. The segmented 2d difference image is partitioned in to non-overlapping rectangular regions of all white and mixed 16*16 blocks. Each partitioned block is transformed in to Boolean switching function in cubical form, treating the pixel values as a output of the function. Minimizing these switching functions using Quine- McCluskey minimization algorithm performs compression.

Keywords: *Logic functions, Boolean functions, cube Quine Mc-Cluskey , prediction residuals*

1. Introduction

The remainder of the paper is organized as follows: Section(2) focuses on the proposed compression techniques. Section (3) emphasizes on the decompression techniques. Section (4) focuses on the format of the compressed image. Section (5)shows the implementation results.

2. Proposed Compression Technique.

2.1 File Extraction

The image pixels are extracted from the Gray image and then stored in a two-dimensional array. To handle images of different sizes, the dimensions of the image are adjusted to the next higher multiple of sixteen. For example, the dimensions of an image are 600*800 pixels. Then the dimensions of the image are adjusted to 608 *800 pixels. The additional pixels will be filled with zeros.

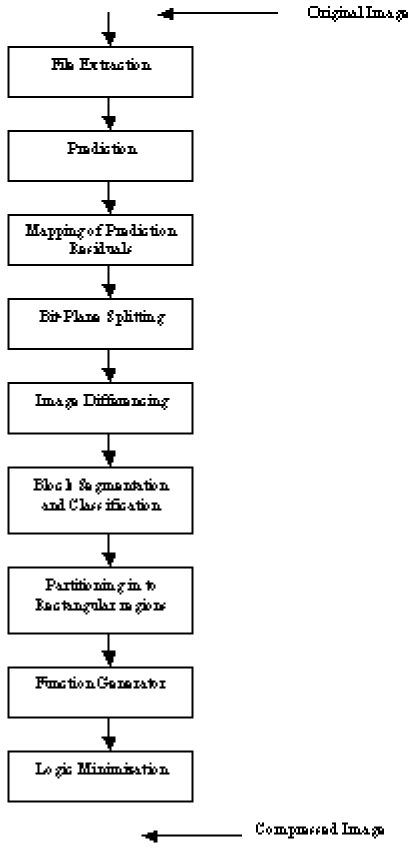

Figure 1. Proposed Compression techniques.

2.2. Prediction.

A prediction process can reduce the high degree of correlation between the adjacent pixels in an image. In this work, three different prediction methods are used.
a). For the pixels in the first row except the first pixel I[0, 0] (which is used as the reference pixel and is coded as it is). The predicted value is the value of the pixel on the left of the current pixel

$$\hat{I}[0, y] = I[0, y-1]$$

Where y=1. . . ; w-1 and w denotes the width of the image.

b) For the pixels in the first column except the reference pixel the predicted value is the value of the pixel directly above

$$\hat{I}[x, 0] = I[x-1, 0]$$

Where y=1 . . . h-1 and h denotes the height of the image.

c). The MED used by LOCO-1(low complexity lossless coder)[10] is used for the interior pixels where x = 1 . . . h-1 and y = 1 . . . w-1. Similar to GAP, MED also detects horizontal or vertical edges by examining the values of north (n), west (w) and north-west (nw) neighbors of the current sample x [i, j] as illustrated in Figure below

	C	b	d
e	a	x	

The north pixel is used as the predictor in case of vertical edge detection. The west pixel is applied in case of horizontal edge. Finally, if neither a vertical edge nor a horizontal edge is detected, an interpolation is used to compute the prediction value. Specifically, prediction is performed according to the following equation:

$$\hat{x} = \begin{cases} \min(a,b) \text{if} & c \geq \max(a,b) \\ \max(a,b) \text{if} & c \leq \min(a,b) \\ a+b-c & \text{otherwise} \end{cases}$$

2.3. Mapping of Prediction Residuals.

The prediction residuals is calculated as.

E= I (x, y) – ^I (x, y)

As such, it is appropriate to map the positive and negative prediction residuals e into positive numbers ^e using the following equations:

$$\hat{e} = \begin{cases} 2|e|-1 & \text{if } e < 0 \\ 2e & \text{if } e \geq 0 \end{cases}$$

2.4. Bit-Plane Splitting.

Consider an M*N image in which each pixel value is represented by k bits. By selecting a single bit from the same position in the binary representation of each pixel, an M*N binary image is formed. In this way the original image can be decomposed into a set of k (M*N) bit-planes numbered 0 for the least-significant bit (LSB) plane to k-1 for the most significant bit (MSB) plane. Bit-plane decomposition offers a viable lossless compression scheme whereby the image can be reconstructed progressively. This technique encodes the bit-planes independently and takes advantage of the existence of large uniform areas in each plane to achieve high compression.

Given an X-bit per pixel image, slicing the image at different planes (bit-planes) plays an important role in image processing. An application of this technique is data compression. In general, 8-bit per pixel images are processed. We can slice an image into the following bit-planes. Zero is the least significant bit (LSB) and 7 is the most significant bit (MSB):

2.5. Image Differencing.

The pixels in a given row or column are de-correlated by performing an Exclusive-OR logic operation between neighboring pixels. In contrast to the 1D image differencing [1] in, the proposed technique performs a 2D image differencing of the pixels by first applying the operation to the rows followed by the columns as follows:

$$D1[x, y] = \begin{cases} I[x, y] & \text{if } y=0 \\ I[x, y] \oplus I[x, y-1] & \text{if } y \neq 0 \end{cases}$$

$$D2[x, y] = \begin{cases} D1[x, y] & \text{if } x=0 \\ D1[x, y] \oplus D1[x, y] & \text{if } x \neq 0 \end{cases}$$

Where $\oplus$ represents the Exclusive-OR logic operation, and y and x denote the rows and columns of the images, respectively. I [x, y], and D2 [x, y] refer to the pixels in the original, 1D and 2D difference images, respectively. In the case of 1D difference image, the operation is applied to the rows alone. The image differencing operation has resulted in a significant reduction of white pixels in the difference image.

2.6. Block Segmentation and Classification.

In image splitting an image is divided in to sub image (windows or blocks) of size 16*16 pixels. This done because of the fact that the correlation on the image is generally local so that the smaller blocks often results in better compression. The 2D difference image[1] is segmented into 16*16 blocks, which are classified as:

(a) All-black: all the 256 pixels in the 16*16 block are black.
(b) All-white: all the 256 pixels in the 16*16 block are white.
(c) Mixed: the 16*16 block consists of both black and white pixels.

The number of all-white blocks is nearly zero since the image difference operation effectively reduces the no. of white pixels in the original image

If the 16*16 block consists of all 0's, it is a uniform block that is represented by its header only i.e. by 00 .

If the 16*16 block consists of all 1's, it is a uniform block that is represented by its header only i.e. by 11.

If the 16*16 block consists of both 1's and 0's,it is a mixed block that is represented by its header i.e 10

2.7. Partitioning into Rectangular Regions.

After the 2D image differencing operation, there are very few all-white blocks left if any. It is more efficient to partition the segmented 2D difference image into rectangular regions of all-white and mixed 16*16 blocks and code only these regions. The two methods for the rectangular partitioning are the overlapping (non-disjoint) and non overlapping (disjoint) methods. Although the number of rectangular regions obtained by the overlapping partitioning method is always less than or equal to the non-overlapping method, this advantage has been obtained at the expense of computation effort and time. Moreover, the overlapping method requires the program to keep track of regions that may have been covered more than once thus resulting in further computation effort and time. As such, the non-overlapping method is chosen.

2.8. Function Generator.

Function generation [7] generates a switching function in cubical form for each window, by assigning the pixels to minterms according to Gray code. Gray code is chosen because of its unit distance property, to capture the correlation likely to be present among adjacent pixels, by assigning geometrically adjacent pixels of a window to logically adjacent minterms.

Pixels are scanned row wise with a reversal of the direction for adjacent rows as shown in figure , to ensures that pixels at the ends of consecutive lines are mapped to logically adjacent code. An image with M pixels is converted to a function of [log2M] variables.
Figure shows the conversion of a Gray image with 16pixels in to a four variable switching function according to the scheme. Fig. shows the gray image where the shaded squares correspond to black pixels. Fig. illustrates the karnaugh map of the function and fig, the unminimized and minimized ON-set respectively.

2.8. Logic Minimization.

Logic minimization is performed on the generated function using Quine-Mcclusky minimization algorithm to find the equivalent minimized representation. For a particular function , in general, the number of cubes in its minimized ON-set and OFF-set are different. Better compression can be achieved by choosing the set with lesser number of cubes, since both represent the same function. An additional bit is needed to encode this information.

3. Decompression Techniques

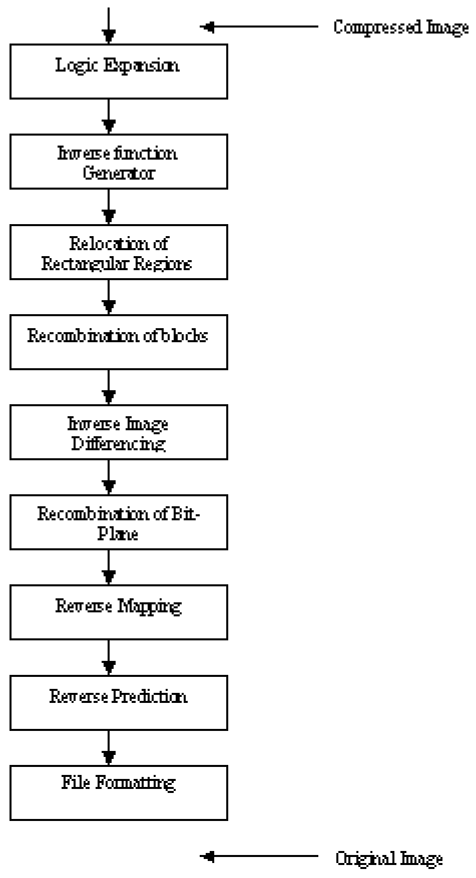

Figure 2. Proposed Decompression techniques

The block diagram for the decompression of a compressed binary image is shown in Fig.

To recover a copy of the original image, the following steps are performed:
(a) The dimensions of the image are extracted from the compressed image.
(b) The logic expansion is relatively simple and consists of recovering the minterms corresponding to a block and expanding the function back into its truth table form.

Logic Expansion of a compressed block is done by computing the value of the function for each of its possible minterms using the cube subsuming operation and assigning these values to the corresponding pixels of the block. If a minterms subsumes the minimized cube of the ON-set of a block then the pixels evaluates to 1;otherwise the pixel evaluates to 0. Figure illustrates the recovery of a block from its minimized cubical form using cube subsuming operation. The minterm of the function is generated by using a Gray Counter and cheek whether each minterm subsumes the cube. Cube subsuming operation [7] involves only comparisons

Definition 6: let A and B be two cubes of n variables where A= {a1, a2………. an} and B = {b1, b2……… bn }, A subsumes B if $a_i \leq b_1$ for i=1…N.

≤	0	1	X
0	Y	N	Y
1	N	Y	Y
x	N	N	Y

Table 1. Subsuming operation

(c) The locations of the non-overlapping rectangular regions (isolated followed by non-isolated) of all-white 16 · 16 blocks are then determined.

(d) The rectangular regions (isolated followed by non-isolated) of mixed 16 *16 blocks are reconstructed.

(e) All the 16*16 blocks corresponds to the same bit plane can be recombined together to form the Bit-Plane

(f) To recover the pixels I[x, y] in the original image from the corresponding pixels D2[x, y] in the 2D difference image, the following inverse differencing operation is performed on the columns followed by the rows:

$$D1[x, y] = \begin{cases} D2[x, y] & \text{if } x=0 \\ D2[x, y] \oplus D1[x-1, y] & \text{if } x \neq 0 \end{cases}$$

$$I[x, y] = \begin{cases} D1[x, y] & \text{if } y=0 \\ D1[x, y] \oplus I[x, y-1] & \text{if } y \neq 0 \end{cases}$$

prediction residuals. A reverse mapping of the prediction residuals is performed as follows:

$$e = \begin{cases} \hat{}e/2 & \text{if } e \bmod 2=0; \\ \hat{}(e+1)/2 & \text{otherwise} \end{cases}$$

(h) A reverse prediction process is applied to the prediction residuals to obtain the image pixels.

(I) The image pixels are decoded to their original intensity and then stored as a Gray Image.

3.4. Format of the compressed Image

The original Image of the size R*C pixels is divided in to eight bit planes, and each of the bit-plane is divided in to windows of size 16*16 pixels, the compressed image has a global header portion followed by the compressed bit-plane. The global header has N1,N2 bits to indicate R,C respectively. The first 3 bits of the windows are used to represent the bit-plane no., the Next 2 bits , as given below, indicate the encoding Scheme used for each window

- If the 16*16 block consists of all 0's, it is a uniform block that is represented by its header only i.e. by 00 .
- If the 16*16 block consists of all 1's, it is a uniform block that is represented by its header only i.e. by 11.
- If the 16*16 block consists of both 1's and 0's,it is a mixed block that is represented by its header i.e 10

If the no. of cubes of the minimized function is not zero, the next m bits are used to indicate the no. of cubes, encoded Cubes are placed after this.

Width	Height	Compressed Bit-Plane
2 Bytes	2Bytes	Variable

Fig.3. Format of the compressed image.

Bit Plane No.	Encoding Scheme	No. of Cubes	Encoded Cubes
2 Bytes	2 Bytes	M bits	----------

Fig.4. Format of the Compressed Bit-plane

5. Experimental Results

5.1. Introduction

The compression and decompression have been implemented in Matlab Language on a personal computer and tested on the set of several images [8]. Compression effectiveness was evaluated by comparing the size of the compressed output with the size of the raw pixel data; header data was excluded from the calculations. the effectiveness of compression is measured inn three ways:

- Relative to the uncompressed file size,
- Relative to the nominal number of bits per pixels,
- Relative to the compression and decompression time.

5.2. Compression Ratio

Compression ratio is defined as:

$$\frac{\text{Input bits}}{\text{Output bits}}$$

Image	Original size(kb)	Compression ratio of the Proposed techniques
Airplane	65	1.58
Boat	257	1.20
Moon	65	1.22
Couple	257	1.24
Man	1025	1.12

Table 2 Compression Ratio of the proposed Techniques

5.3. Entropy in Bits/ Pixel

From the below Table the entropy achieved by the proposed technique is always smaller than that of the original image for the set of images.

Image	Original Image	Compressed Image by the proposed techniques
Airplane	6.4887	5.7909
Boat	7.2055	6.4714
Moon	6.7026	5.8058
Couple	7.2330	7.1158
Man	7.5389	7.2011

Table 3. Entropy in bits per pixel

6. Comparison With Existing State-Of-Art Technologies

In this chapter, a comparative study of lossless compression algorithms is presented. The following algorithms are considered: UNIX compress (LZW) JPEG-LS based on LOCO. In cases where the algorithm under consideration may only be applied to binary data, the bitplanes of the gray scale image are separated, with and without Gray encoding, and the compression is applied to individual bit planes. Testing is done using a set of Gray images.

Table below lists the results of the compression ratios as well as a comparison with WinZip algorithm and LOCO-I . From Table below it is obvious that the performance of the proposed technique is always better than WinZip. The new technique compresses better than Loco-1 in three out of the five images. When compared to the state-of-the-art techniques such as the WinZip and LOCO-I, the compression ratios for the new technique is better in three out of five images This is mainly due to the lower number of intensity levels present in these three images and similar results can be expected for other images with the same property.

Image	Size	WinZip	Proposed Techniques	LOCO-1
Airplane	65	1.58	1.41	2.09
Boat	257	1.20	2. 82	1.61
Moon	65	1.22	1.35	1.83
Couple	257	1.24	2.71	1.71
Man	1025	1.12	1.88	1.78

Table4. Comparison of Compression Ratios

7.Conclusion

A lossless compression technique for gray-scale images has been proposed and implemented. The raw image is extracted from the gray images. A preprocessing step to arrange the pixel intensities in the order of their probabilities in continuous order is performed followed by series of predictions for the boundary and interior pixels. The prediction residuals are mapped into all positive numbers before splitting them into binary bit-planes. Two-dimensional (2D) differencing operation is applied to bit-planes prior to segmentation and classification. Performing an Exclusive-OR logic operation between neighboring pixels in the bit planes. The segmented 2d difference image is partitioned in to non-overlapping rectangular regions of all white and mixed 16*16 blocks. Each partitioned block is transformed in to Boolean switching function in cubical form, treating the pixel values as a output of the function. Compression is performed by minimizing these switching function using Quine Mcclusky minimization algorithm.
A major advantage of this approach is it performs better than UNIX Compress for most of the test images in terms of compression ratio. The execution time for this proposed techniques is rather high.

References

[1]. Bogdan.J.falkowski "Lossless binary image compression using logic functions and spectra" School of Electrical and Electronic Engineering, Nanyang Technological University, Block S1, Nanyang Avenue,Oct 2001.

[2]. Bogdan.J.falkowski "Lossless Binary image Compression using logic function methods". Proc S.E.Europe workshop Computational Intelligence and Information Technologies, Nis, Yogoslavia, June 2001

[3] Slaven Marusic, Guang Deng. " A Study of Two NewAdaptivePredictorsforLosslessImageCompression" In: Proc.ICIP;1997

[4] Quddus A, Fahmy MM. "A new compression technique for Binary text images". In: Proc. 2nd IEEE Symp. Computers and Communications, July 1997. p. 194–8.

[5] Memon, N., Sippy, V., and Wu, X.: 'A comparison of prediction schemes proposed for a new lossless image compression standard'.Proc. 29th IEEE Int. Symposium on Circuits and Systems, Atlanta, GA, May 1996, vol. 2, pp. 309–312

[6] S. Mohamed and M. Fahmy. "Binary image compression using efficient partitioning into rectangular regions". IEEE Trans. on Communications, 43:1888-1892, May 1995.

[7]. Jacob Augustine, Wen Feng, James Jacob "Lossless Minimization Based approach for Compressing Image Data" In: Proc.IEEE 8th Int. Conf. VLSI Design; January 1995. P.225-228.

[8] http://sipi.usc.edu/database/database.cgi?volume= misc&image=25#top

[9] Wakerly JF. Digital Design: Principles and Practices. Second ed. Englewood Cliffs: Prentice-Hall; 1994.