Lossless Image Compression LOCO-R Algorithm for 16 bit Image

Komal Ramteke
Faculty of Information Technology Department,
NYSS College of Engineering and Research,
Nagpur

Sunita Rawat
Faculty of Computer Technology Department,
NYSS College of Engineering and Research,
Nagpur

ABSTRACT
Lossless image compression is used for reducing the volume of image data without compromising the data quality. The aim is to reduce the demand on processors and to increase the speed at which images can be compressed. LOCO-R algorithm is used for image compression. It is based on the LOCO-I (Low complexity Lossless Compression for image) algorithm. The LOCO-R algorithm has already been implemented for image with 8-bit pixel values.

In this paper, we proposed the LOCO-R algorithm for 16 bit image; it reduces the implementation complexity and reduced the compression ratio. This algorithm is based on prediction and context models; the model is tuned for efficient performance in conjunction with a collection of Huffman codes, which realized with Golomb-Rice code.

Keywords
Lossless Image Compression, Huffman Coding, Context Prediction method.

1. INTRODUCTION
Digital images commonly contain lots of redundant information, and thus they are usually compressed to remove redundancy and minimize the storage space or transport bandwidth.

Compression can be categorized in two broad ways: Lossless Compression and Lossy Compression. Lossless Compression in which data is compressed and can be reconstituted (uncompressed) without loss of detail or information. In Lossy Compression, the aim is to obtain the best possible fidelity for a given bit-rate or minimizing the bit-rate to achieve a given fidelity measure.

Lossless image compression is used for reducing the volume of image data without compromising the data quality. The aim is to reduce the demand on processors and to increase the speed at which images can be compressed. The compressed image produced is a sequence of bits from which the original image can be reconstructed. Scientific or legal considerations make lossy compression unacceptable for many high performance applications such as geophysics, telemetry, non-destructive evaluation, and medical imaging, which will still require lossless image compression.

LOCO-R is Lossless Image Compression Algorithm. It is based on LOCO-I (Low complexity Lossless Compression for image) algorithm. The LOCO-R algorithm has already been implemented for image with 8-bit pixel values. In this paper, we proposed the LOCO-R algorithm for 16 bit image.

2. RELATED WORK
Liu Zheng-lin', Qian Ying2, Yang Li-ying', Bo Yu', Li Hui' proposed LOCO-R which is based on LOCO-I algorithm with the modifications and betterment. This algorithm reduces obviously the implementation complexity and compared with the Rice Compression algorithm, results illustrates that this algorithm is better than the Rice Compression typically by around 15 percent.[1]

Marcelo J. Weinberger and Gadiel Seroussi proposed, the LOCO-I algorithm and discuss the principles underlying the design of LOCO-I and its Standardization into JPEG-LS. It is conceived as a "low complexity projection" of the universal context modeling paradigm, matching its modeling unit to a simple coding unit. By combining simplicity with the compression potential of context models, it is based on a simple fixed context model, which approaches the capability of the more complex universal techniques for capturing high-order dependencies. LOCO-I attains compression ratios similar or superior to those obtained with state-of-the-art schemes based on arithmetic coding. Moreover, it is within a few percentage points of the best available compression ratios, at a much lower complexity level. [4][8].

Hua Cai and Jiang Li presents a new image coding algorithm based on a simple architecture that is easy to model and encode the residual samples. In the proposed algorithm, each residual sample is separated into three parts: (1) a sign value, (2) a magnitude value, and (3) a magnitude level. A tree structure is then used to organize the magnitude levels. By simply coding the tree and the other two parts without any complicated modeling and entropy coding, good performance can be achieved with very low computational cost in the binary-encoded mode. Moreover, with the aid of context-based arithmetic coding, the magnitude values are further compressed in the arithmetic-coded mode. [16]

3. LOCO-R ALGORITHM FOR 16 BIT
Previously proposed LOCO-R algorithm is design for only 8 bit image pixels. It gives the better performance as compared with the other lossless compression algorithm. We proposed modified LOCO-R algorithm which is work on 16 bit images. It reduces the computational efficiency of 16 bit image and compression rate.

The LOCO-R algorithm based on the LOCO-I algorithm, it takes as input a rectangular image with 16-bit pixel values (the pixel values are within the range 0 to 65535). The compressed image produced is a sequence of bits from which the original image can be reconstructed. Let wd be the image width and ht be the image height. Pixels are identified by coordinates (x, y) with x in the range [O, wd-l] and y in the range [O, ht-l]. This paper supposes: (0, 0) corresponds to the upper left comer of the image. [5]

The LOCO-R algorithm is based on predictive compression. During compression, the pixels of the image are processed in raster scan order. Specifically, y is Incremented through the

range [O,ht-l], and for each y value, x is incremented through the range[O,wd-l]. (Thus, the y dimensions, the slowly varying dimension.)The first two pixels, with coordinates (0,0) and (1,0), are simply put into the output bit stream uncoded.

For all other pixels of the image, the processing that occurs can be conceptually divided into following steps:

1. Find out the Histogram Values of the input image. These would give us the probability of the pixels
2. Find out the Huffman codes for the given input image with the probabilities. This would complete the prediction step.
3. Estimate the data obtained from the previous step.
4. Once estimated, we can encode the pixel values based on predictive sampling.
5. Again sum up the encoded data to form the newer version of context data.
6. Repeat this process for every pixel.

3.1 Context Modeling

The reducing the number of parameters is a key objective in a context modeling scheme. In a sequential formulation, the goal is to avoid "context dilution" while in a two-pass scheme we wish to reduce unnecessary table overhead. The total number of parameters in the model depends on the number of free parameters defining the coding distribution at each context and on the number of contexts. The context that conditions the encoding of the current prediction residual in LOCO-R is built out of the differences $g_1=d-a$, $g_2=a-c$, $g_3=c-b$, and $g_4=b-e$. These differences represent the local gradient, thus capturing the level of activity (smoothness, edginess) surrounding a pixel, which governs the statistical behavior of prediction errors. Notice that this approach differs from the one adopted in the Sunset family and other schemes, where the context is built out of the prediction errors incurred in previous encodings. By symmetry, g_1, g_2, and g_3 influence the model in the same way. Since further parameter reduction is obviously needed, each difference g_j, j=1, 2, 3, is quantized into a small number or approximately equiprobable regions (the same regions for j = 1, 2,3). In this maximizes the mutual information between the current pixel and its context, an information-theoretic measure of the amount of information provided by the conditioning context on the pixel value to be modeled. Difference g_4, being farther away from the predicted pixel, is quantized more coarsely.

In principle, the number of regions into which each context difference is quantized should be adaptively optimized. However, the low complexity requirement dictates a fixed number of "equiprobable" regions. By symmetry, there is one region centered at the difference value 0, and if the interval $[r_1, r_2]$ represents a region, then so does $[-r_1, -r_2]$.Thus, the total number of quantization regions for g_j, j=1, 2, 3, is an odd integer $2R + 1$, while g_4 is quantized into $2T + 1$ regions, $T < R$. This leads to a total of $(2T+1) \times (2R+1)^3$ different contexts. A further reduction in the number of contexts is obtained after observing that, by symmetry, it is reasonable to assume $\text{Prob}\{e_{i+1} = \Delta | Ci = [q_1, q_2, q_3, q_4]\} = \text{Prob}\{e_{i+1} = -\Delta | Ci = [-q_1, -q_2, -q_3, -q_4]\}$

where Ci represents the quantized context quartet and q_j, j = 1, ... ,4, are quantized differences corresponding, respectively, to gj, j = 1,. . . . . , 4.

To complete the definition of the contexts in LOCO-R, it remains to specify the values of the boundaries between quantization regions. F or an 8-bit per pixel alphabet, the quantization regions for g_j, j = 1, 2, 3, are $\{0\}$, $\{1, 2\}$, $\{3, 4, 5, 6\}$, $\{7, 8, \ldots, 14\}$, $\{ e \mid e \geq 15 \}$, and their corresponding negative counterparts. The three quantization regions for g_4 are $|g_4| < 5$, $g_4 \geq 5$, and $g_4 \leq -5$.

3.2 Huffman Coding

Huffman coder always assigns long codeword's to less frequent symbols and short codeword's to frequent symbols. Huffman codes are optimal in the sense that they generate a set of variable length binary codeword's of minimum average length, as long as the source alphabet and PMF are available. Huffman codes always produce an average code length within one bit of the entropy bound. Huffman coders are adaptive and estimate the source PMF from the coded samples. In order to encode images:

- Divide image up into 8x8 blocks
- Each block is a symbol to be coded
- compute Huffman codes for set of block
- Encode blocks accordingly

The basic idea in Huffman coding is to assign short codeword's to those input blocks with high probabilities and long codeword's to those with low probabilities. A Huffman code is designed by merging together the two least probable characters, and repeating this process until there is only one character remaining. A code tree is thus generated and the Huffman code is obtained from the labeling of the code tree.

3.3 Prediction

Ideally, guessing the value of the current pixel $xi+1$ based on a, b, c, d, and e should be done by adaptively learning a model conditioned on the local edge direction. However, our complexity constraints rule out this possibility. Yet, a primitive edge detector is still desirable in order to approach the best possible predictor. The approach in LOCO-R consists on performing a primitive test to detect vertical or horizontal edges. If an edge is not detected, then the guessed value is $a+b-c$, as this would be the value of x_i+1 if the current pixel belonged to the "plane" defined by the three neighboring pixels with "heights" a, b and c.This expresses the expected smoothness of the image in the absence of edges. Specifically, the LOCO-R predictor guesses:

$$\hat{x}_{i+1} \triangleq \begin{cases} \min(a, b) & \text{if } c \geq \max(a, b) \\ \max(a, b) & \text{if } c \leq \min(a, b) \\ a + b - c & \text{otherwise.} \end{cases}$$

Assuming, without loss of generality, that $a \leq b$, then the predictor of (1) can be interpreted as picking a in many cases where a vertical edge exists left of the current location, b in many cases of an horizontal edge above the current location, or a plane predictor $a + b - c$ if no edge has been detected.

The above predictor has been employed in image compression applications, although under a different interpretation. The guessed value is seen as the *median* of three fixed predictors, a, b, and $a + b - c$. Notice that the predictor of (1) does not employ either d or e, which will be used in context modeling.

The lossless coding process employs a simple predictive coding model called differential pulse code modulation (DPCM). This is a model in which predictions of the sample values are estimated from the neighboring samples that are already coded in the image. Most predictors take the average of the samples immediately above and to the left of the target sample. DPCM encodes the differences between the predicted samples instead of encoding each sample independently. The differences from one sample to the next are usually close to zero. A typical DPCM encoder is displayed in Fig.2. The

block in the figure acts as a storage of the current sample which will later be a previous sample.

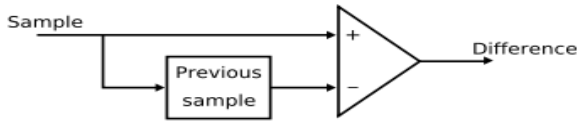

Fig.1: DPCM encoder model[13]

The main steps of lossless operation mode are depicted in Fig.3. In the process, the predictor combines up to three neighboring samples at a, b, and c , in order to produce a prediction of the sample value at the position labeled by x. The three neighboring samples must be already predicted samples.

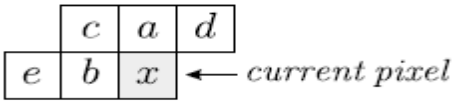

Fig. 2: Three neighboring samples around the sample to be predicted

3.3 Estimation

A more accurate estimate will produce a smaller and thus more compressible residual. A preliminary estimate is computed with a fixed (i.e., non-adaptive) estimator, and the adaptively computed bias is added to form the final estimate.
The final estimate of a pixel is obtained by adding the bias value (or its negative, if the invert flag is set) to the initial estimate. The bias value attempts to adaptively influence the fixed predictor. The addition of the bias could put the estimate outside the range of 0 to 65535; therefore, the estimate is clipped to this range.

After the encoder determines the estimate of the current pixel value, the difference between the actual pixel value and the estimate is losslessly encoded. (If the invert flag is set, the negative of this difference is encoded instead). We denote this value by €. Although € can take on values from -65535 to 65535, only 65536 of these values are possible for a given pixel estimate.

We eliminate the unused values by mapping € to the range -128 to 127, accomplished by simply truncating the two's complement representation of € to 16 bits, and interpreting the resulting number as a two's complement 16 bit integer, referred to as €'. The €' values have a distribution that is usually approximately two-sided geometric. the paper map €' to M(€') to get a quantity with a distribution that is approximately (one-sided) geometric, with the range from 0 to 65535. This is accomplished with the transformation

$$M(\epsilon') = \begin{cases} 2\epsilon' & \text{If } \epsilon' >= 0 \\ -2\epsilon' - 1 & \text{If } \epsilon' < 0 \end{cases}$$

3.4 Golomb coding

Golomb coding uses a tunable parameter M to divide an input value into two parts: q, the result of a division by M, and r, the remainder. The quotient is sent in unary coding, followed by the remainder in truncated binary encoding. When $M = 1$ Golomb coding is equivalent to unary coding.

Golomb-Rice codes can be thought of as codes that indicate a number by the position of the bin (q), and the offset within the bin (r). The above figure shows the position q, and offset r for the encoding of integer N using Golomb-Rice parameter M. Formally, the two parts are given by the following expression, where x is the number being encoded: and $r = x - qM - 1$ The final result looks like:

Note that r can be of a varying number of bits, and is specifically only b bits for Rice code, and switches between b-1 and b bits for Golomb code (i.e. M is not a power of 2): Let . If , then use b-1 bits to encode r. If , then use b bits to encode r. Clearly, $b = \log2(M)$ if M is a power of 2 and we can encode all values of r with b bits. The parameter M is a function of the corresponding Bernoulli process, which is parameterized by $p = P(X = 0)$ the probability of success in a given Bernoulli trial. M and p are related by these inequalities: The Golomb code for this distribution is equivalent to the Huffman code for the same probabilities, if it were possible to compute the Huffman code.

3.5 Updating Context Data

After the encoding operation takes place, the data associated with the context are updated. The goal of this process is to ensure that later values with the same context are encoded efficiently. As previously mentioned, there are 32 bits of data associated with each context:
(1) Occurrence count (count, 6 bit unsigned integer)
(2) Magnitude sum of residuals (msum, 13 bit unsigned integer)
(3) Sum of residuals (rsum, 16 bit signed integer)
(4) Bias value (bias, 5 bit signed integer)
For all contexts, the values of these data before compression are set identically: count is 2, msum is 12, rsum is 0, and bias is O. A small compression improvement might be obtained by carefully choosing initial values to minimize the typical time to adapt to an image.

Note that during the update process count, rsum, and bias may take on values outside their usual ranges (e.g., the paper temporarily allow count to be 64 even though it is stored as a 6 bit integer).

The following steps occur during the update process:
(1) Increment count by 1.
(2) Add €' to rsum. If the new rsum is greater than 0, bias is increased by 1 (unless it already equals its maximum value, 15) and rsum is decreased by count; if the new rsum is less than -count, bias is decreased by 1 (unless it equals its minimum value, -16) and rsum is increased by count.
(3) Clip rsum to the range [-128,127].
(4) Increase msum by I€'I.
(5) If count is 64, then count, msum, and rsum are all divided by 2 (accomplished with a right shift, with sign extension in the case of rsum).

4. RESULT AND DISCUSSION

In this paper, we used LOCO-R algorithm for 16 bit image having size of 800x600. It gives an efficient result which compresses an image upto 50%.

Fig.3: Shows an original image.

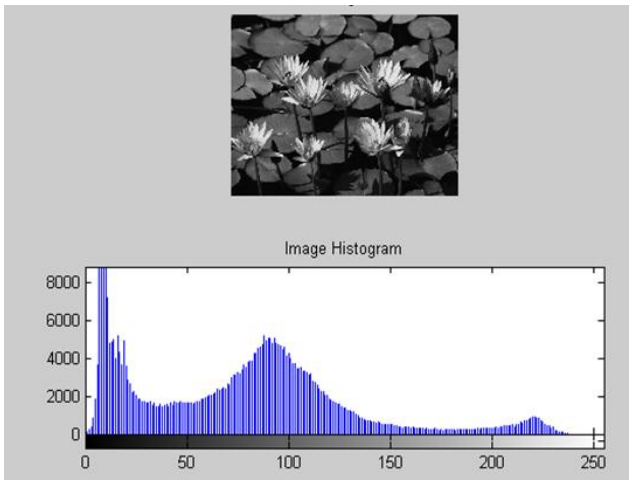

Fig.4: Shows the histogram of the original 16 bit gray scale image with their actual probability values.

Then the probability values of the each pixel are calculated using actual value of histogram divided by size of the image. Compression Ratio is calculated by the difference between actual image and the output image by size of the actual image. The image compression Ratio is 0.5221.

Fig. 5: Shows a decompressed image

5. CONCLUSION

The compression achieved by LOCO-R is upto 50 percent. An algorithm of lossless image compression for 16 bit image is based on context prediction is proposed. The algorithm and implementation can easily be adapted to handle modifications in various parameters. Some parameters of the design could be made controllable with the interface to allow greater flexibility.

As LOCO-R is designed for 16 bit (gray-scale) images, it represents significant progress toward producing lossless image compression with improved compression performance as compared with currently available solution.

6. REFERENCES

[1] Liu Zheng-lin, Qian Ying2, Yang Li-ying, Bo Yu, Li Hui (2010), ”An Improved Lossless Image Compression Algorithm LOCO-R”, International Conference On Computer Design And Appliations (ICCDA).

[2] Dang Gang,Cheng Zhi-Quan, ZHOU Jingwen, LI Liang, Jin Shiyao (2010),”An improved progressive Lossless Compression Algorithm” ,IEEE

[3] Ming Yang,Nikolaos Bourbakis(2005),”An overview of Lossless Digital Image Compression Techniques”,IEEE, Information Acquisition and Processing.

[4] Marcelo J. Weinberger, Gadiel Seroussi, and Guillermo Sapiro,(2000) “The LOCO-I Lossless Image Compression Algorithm:Principles and Standardization into JPEG-LS”, IEEE Transaction on Image Processing, VOL. 9, NO. 8

[5] M.Klimesh,1 V.Stanton,1 and D. Watola1,(2001) “Hardware Implementation of a Lossless Image Compression Algorithm Using a Field Programmable Gate Array” TMO Progress Report 42-144.

[6] M.Rabbani and P.Jones,(1991) “Digital Image Compression Techniques”, Bellingham, Washington: SPIE Publications.

[7] Marcelo J. Weinberger, Gadiel Seroussi, and Guillermo Sapiro,(“LOCO-I: A Low Complexity, Context-Based, Lossless Image Compression Algorithm” IEEE Trans. Image Processing.

[8] N.D.Memon and K.Sayood,(1995) “Lossless image compression: A comparative study,” in Proc. SPIE (Still-Image Compression), vol.2418, pp. 8–20.

[9] N. Merhav, G.Seroussi, and M. J.W einberger,(1996) “Modeling and low-complexity adaptive coding for image prediction residuals.” To be presented at the 1996 Int’l Conference on Image Processing, Lausanne.

[10] X. W u, N.Memon, and K.Sayood,(1995) “A context-based, adaptive, lossless/nearly lossless coding scheme for continuous-tone images (CALC).” A proposal submitted in response to the Call for Contributions for ISO/IEC JTC 1.29.12.

[11] M. J.W einberger, J.Rissanen, and R.Arps, “Applications of universal context modeling to lossless compression of gray-scale images.” To appear in IEEE Trans. Image Processing.

[12] Information Technology- Lossless and Near-Lossless Compression of Continuous-Tone Still Images,(1999), ISOIlEC 14495-1, ITU Recommendation T.87.

[13] Das, M., and Chande, S.,(2001) “Efficient Lossless Image Compression Using a Simple Adaptive DPCM Model”,IEEE , pp.164-167.

[14] Boulgouris, N.V., Tzovaras, D., and Strintzis, M.G.,(2001) “Lossless Image Compression Based on Optimal Predication, Adaptive Lifting, and Conditional Arithmetic Coding”, IEEE Transactions on Image Processing, Vol. 10, No. 1, pp.1-14.

[15] W. Szpankowski,(2000) “Asymptotic Average Redundancy of Huffman (and Other) Block Codes”, IEEE Trans. Information Theory, 46 (7), pp. 2434–2443.

[16] Hua Cai and Jiang Li **“**Lossless Image Compression with Tree Coding of magnitude levels”,IEEE,pp. 1-4.