NEAR-LOSSLESS IMAGE COMPRESSION BASED ON MAXIMIZATION OF RUN LENGTH SEQUENCES

E. Nasr-Esfahani[1], S. Samavi[1,2], N. Karimi[1], S. Shirani[2]
Department of Electrical and Computer Engineering
[1]Isfahan University of Technology, Isfahan, Iran
[2]McMaster University, Ontario, Canada

ABSTRACT

In this paper an algorithm is proposed which performs near-lossless image compression. For each pixel in a row of the image a group of value-states are considered, which have values close to that of the pixel. A trellis is constructed for every row of the image where the nodes of the trellis are the states of the pixels of that row. The goal of the algorithm is to find a path on this trellis that creates a sequence which can be efficiently coded using run length encoding (RLE). For sections of the pixels of the row that suitable RLE cannot be achieved then minimization of the entropy is employed to complete a path on the trellis. The application of the algorithm to a wide range of standard images shows that the scheme, while having low computational complexity, is competitive with other near-lossless image compression methods.

Index Terms— Image compression, run length encoding, entropy coding, near lossless.

1. INTRODUCTION

Image compression plays a very important role in applications such as videoconferencing, mobile communications, and medical imaging. Image compression continues to be an important area of research though a lot of works have been reported in the literature and several coding techniques have emerged [1, 2].

Traditionally, image compression techniques have been classified into two categories of lossy and lossless methods. Lossy methods are required in situations where significant compression ratios are needed but do not allow exact recovery of the original images. This is the case, for example, of digital photography, where losing some of the image detail is tolerable. In lossless methods, on the other hand, compression ratio is relatively low, but the exact reconstruction of the original images is essential. This is true where small image details can be very important, such as in medical and space imaging or in remote sensing [3].

Exact lossless recovery is, however, not an essential requirement in many situations because different applications may tolerate different limits of deviation from the original value.

In many applications, the issue is not so much as whether lossy compression should be used, but rather, how to trade off between compression ratios and distortion. If high compression ratios can be achieved with small enough distortion to ensure sufficient accuracy for specific purposes then it is said that the method is near-lossless. For example if dependable diagnoses for medical imaging could be performed then the near-lossless method is acceptable. A near-lossless criterion is defined such that no pixel is changed in magnitude by more than d gray level where d is a small non-negative integer [4, 5].

Near-lossless compression could potentially lead to a significant increase in compression ratios while providing, at the same time, quantitative guarantees about the type and amount of distortion introduced.

Transform domain approach using the discrete cosine transform (DCT) or the discrete wavelet transform (DWT) along with an entropy coding is usually deployed for lossy compression. For lossless and near lossless compression using differential pulse code modulation (DPCM) based on different predictors and error-modeling schemes are popularly used due to their simplicity as well as its efficiency [6].

In this paper we use a trellis-based algorithm to perform a near-lossless image compression. For every pixel a number of states are considered with values equal and close to the intensity of that pixel. For each row a trellis is constructed from these value-neighbors. A path is chosen on the trellis through a greedy algorithm which maximizes the sequence of similar prediction errors on this trellis. By doing so better RLE coding is achieved.

The rest of the paper is organized in the following manner: In section 2 we present our Greedy Path Selection Algorithm (GPSA). In section 3 a modified version of the algorithm with reduced complexity is outlined. Simulation results are presented in Section 4. Section 5 is dedicated to concluding remarks.

2. GREEDY PATH SELECTION ALGORITHM

In this section we describe our greedy path selection method. Suppose we label pixels of i th row of the image as p_j^i where j is the column number. If the pixels of the reconstructed image after the compression process has values close to the original values then we have near lossless image compression. To perform near lossless compression we can have for every pixel in the i th row of the image $2d+1$ states. The values of these states are at most d units from the original value. The states for pixel at row i and column j are $p_{j,m}^i \in \{p_j^i - d, ..., p_j^i, ..., p_j^i + d\}, m \in [-d, d]$. Now a trellis can be formed for i th row where the nodes are the states of the pixels of that row. Each node of the trellis is labeled as $p_{j,m}^i$. Before the compression is performed the pixel values are chosen from the mentioned trellis. When the selection of a pixel value is finalized at the $(i-1)$ th row then it is labeled as $\hat{p}_j^{i-1}$. We show that picking the pixel values from the nodes of the trellis creates better compression ratios. As an example for forming a trellis suppose the previously constructed row is $\hat{P}^{i-1} = (\hat{p}_1^{i-1}, \hat{p}_2^{i-1}, \hat{p}_3^{i-1}, \hat{p}_4^{i-1}, ...) = (3,7,1,5,...)$ and the current row is $P^i = (p_1^i, p_2^i, p_3^i, p_4^i, ...) = (1,9,5,8,...)$ If d is chosen to be 1 then Figure 1 is the trellis for the first 8 pixels of the current row.

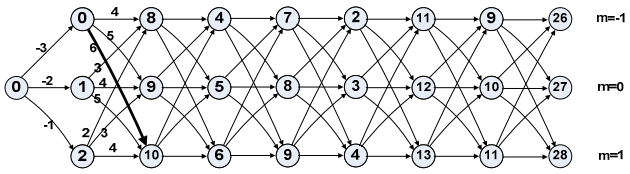

Figure 1. Initial trellis for eight pixels.

We always add a zero-value node to the left of the trellis as a starting point. The edges of the trellis of Figure 1 are labeled with a prediction error value. The predictor uses values from the previously constructed row as well as the values from the trellis of the current row. The predictor that we use is in our algorithm is $e_j^i = p_{j+1,m}^i - [p_{j,m}^i + \hat{p}_{j+1}^{i-1} - \hat{p}_j^{i-1}]$. For example the edge that is drawn thicker in Figure 1, connecting node $p_{1,-1}^i = 0$ to node $p_{2,1}^i = 10$ has label $e_1^i = p_{2,1}^i - [p_{1,-1}^i + \hat{p}_2^{i-1} - \hat{p}_1^{i-1}] = 10 - [0 + 7 - 3] = 6$

From each column of the trellis a node is to replace the original pixel value. Therefore, is seems that the trellis has to be traversed, column by column, along a path, from left to right. It is the edge values that are to be compressed and not the value of the nodes. Suitability of the selected path depends on the coding scheme that is to be used for compression of the sequence of the errors.

In [8] a path selection method is offered with the aim of minimizing entropy. The shortcoming there is that a number of iterations are required. This number depends on a set of initial values. Our proposed method (GPSA), tries to find a path with maximum numbers of consecutive equal errors and in case of a deadlock it tries to minimize the entropy of the produced errors on the edges of the trellis. Steps of the algorithm are shown in Figure 2.

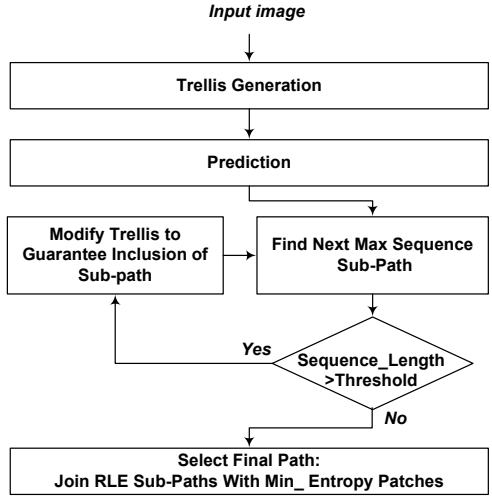

Figure 2. Steps required to compress an image by GPSA

Initially all of the nodes are examined to find the longest sub-path with consecutive equal edge values. This sub-path could be anywhere in the trellis. The sequence of edge errors of this sub-path is obviously suitable for run length encoding. We have to ensure that any total path that is finally selected includes this sub-path. Figure 3 illustrates a sub-path selection example based on the trellis of Figure 1. In this example an initial search finds 4 consecutive zeros. This sub-path has to be included in the final selected path. Suppose that the sub-path starts from column k and runs all the way to column m. By eliminating all other edges between columns k and m except the ones that are on the sub-path inclusion of this sub-path in the final selected path can be guaranteed.

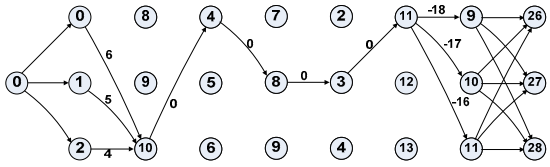

Figure 3. An example of a sub-path.

Then the algorithm excludes the found sub-path and finds the next best sub-path in the remaining columns. This process is stopped when the length of the obtained sub-path falls below a certain threshold. In our experiments we choose 16 as the threshold. If we were to keep selecting

sub-paths below that threshold then RLE could result in expansion rather than compression. Now we have a number of sub-paths and a final path has to be selected. Since these sub-paths are good candidates for RLE coding, we refer to them as RLE sub-paths. The final overall path will include all of the RLE sub-paths as well as some non-RLE patches. The final path is selected to pass through some non RLE-nodes in a way which tries to reduce the entropy. Hence, these non-RLE parts of the final path can be coded by entropy coding in an efficient manner.

To form the final path the edges of non-RLE nodes are selected by the following criteria: out of all of the edges that enter the node, the one with lowest error is selected. In order to get a path with lowest entropy we need to know the probability of errors. But these probabilities are not known until the path is selected. Using the predictor explained in Section 2, it is possible to increase the probability of occurrence of errors that have small absolute values. Higher number of occurrences ensures longer sequences of equal edge values and better RLE coding. Also, for those non-RLE segments of the path, smaller error values produce smaller entropy.

GPSA has the advantage of not being iterative. The most sluggish routine of the algorithm is its sub-path selection part. These sub-paths are coded with RLE coding. These RLE codes along with error values from non-RLE parts are looked at as string of values. Then the overall string is coded using Huffman method.

In order to evaluate the quality of the reconstructed image PSNR criteria according to Equation 1 is used [7].

$$PSNR = 10\log_{10}\left(\frac{255^2}{\frac{1}{H*W}\sum_{i=1}^{W}\sum_{j=1}^{H}(I_1(i,j)-I_2(i,j))^2}\right) \quad (1)$$

Where $I1(i,j)$ and $I2(i,j)$ are respectively the original and reconstructed pixels at coordinates (i,j) The images have the height of *H* and width of *W*. Each pixel has an integer value between 0 and 255.

The maximum difference between the reconstructed and original pixels is d, hence, $\max|I_1(i,j)-I_2(i,j)|=d$. Inserting this value into Equation 1 produces Equation 2. This gives us the minimum possible value for the PSNR of a reconstructed image that is compressed by a near lossless method. Normally we get better PSNR values for these types of images.

$$PSNR_{\min} = 20\log_{10}\left(\frac{255}{d}\right) \quad (2)$$

Table (1) shows the minimum value of PSNR for different values of d.

Table 1. Diminishing PSNR with increasing value of d.

d	1	2	3	4	5
PSNR	48.13	42.11	38.58	36.09	34.15

3. LOW COMPLEXITY GPSA

The sub-path selection part of GPSA is time consuming. In this section we propose a method which improves the speed of the algorithm. We refer to this algorithm as low-complexity GPSA or in short LCGPSA. The algorithm starts at one end of the trellis and works its way toward the other end. The evaluation of the situation for choosing a path is done locally at each column of the trellis. Details of the algorithm are shown in Figure 4.

```
Algorithm Low_Complexity_GPS(j,m)
begin
 MaxIndex ← NULL;
 MinErrorIndex ← NULL;
 SameExist ← False;
 Max_RunLength ← -1;
 Mini_EdgeError ← ∞;
for k=1 to 2*d+1 do
    if EdgeError(L^k_{j,m})==LastEdgeError(PH_{j,m})
        SameExist ← True;
        if RunLength(PH_{j,m}+L^k_{j,m})>Max_RunLength
            MaxIndex ← k;
            Max_RunLength ←RunLength(PH_{j,m}+L^k_{j,m});
        if |EdgeError(L^k_{j,m})|<Min_EdgeError
            MinErrorIndex ← k;
            Min_EdgeError ← |EdgeError(L^k_{j,m})|;
if SameExist
    return MaxIndex;
else
    return MinErrorIndex;
End.
```

Figure 4. Details of the algorithm.

In this algorithm $L^k_{j,m}$ is the kth edge that connects node m of column j to the nodes in the next column. Function EdgeError($L^k_{j,m}$) finds the error assigned to the edge and function LastEdgeError($PH_{j,m}$) keeps the error of the edge that connects node $p^i_{j,m}$ to its previous node. Also, RunLength($PH_{j,m}+L^k_{j,m}$) calculates the number of consecutive repeated errors of the investigated edge and the path connected to it.

Therefore, in this algorithm local edge selection is initially done by maximizing the run length of the consecutive errors of the selected edge and its immediate predecessors. If none of the current edges can form a run length with the paths to this point then the edge with minimum error is selected. This is shown to reduce the entropy of the string of errors.

As the value of d increases the quality of the reconstructed image diminishes, but since the length of the paths with consecutive equal edge labels increases, better RLE codes are obtained. Therefore, the quality of the

reconstructed image and the compression ratio could be controlled by d.

4. SIMULATION RESULTS

A wide range of standard images were compressed and reconstructed by GPSA and its low complexity version, LCGPSA. Local-search characteristic of LCGPSA has made it much faster than the original GPSA. The amount of the achieved speedup is illustrated in Table 2.

Table 2. Speedup of LCGPSA compared to GPSA.

Image	Lena	Lake	Airplane	Peppers
Speed up	34.01	23.02	24.11	25.11

Compression results are measured by bit per pixel, bpp, and the quality of the reconstructed images are calculated by PSNR. The original GPSA produces an average PSNR that is 0.04dB higher than that of its low complexity version. Also bit per pixel results of the original GPSA is 0.02 bpp lower than that of the low complexity version. Hence, in our comparisons we only use the LCGPSA. Table 3 shows the results from our LCGPSA algorithm with d=1 and d=2. Also, in Table 3 results from algorithm of the reference [1], reference [8] and the near-lossless JPEG-LS are presented for comparison purpose. Reference [8] tries to minimize the entropy. Our algorithm produces better compression results and comparable PSNR as compared to reference [8]. As compared to reference [1] in terms of PSNR our algorithm is advantageous but we produce lower compression ratios. As mentioned before, with increasing the value of d better bpp values are achieved at the expense of lower PSNR. In average lower bit per pixel is produced by LCGPSA as compared to reference [1] and JPEG-LS.

5. CONCLUSION

In this paper we proposed an algorithm for near-lossless compression of images aimed at obtaining good RLE compression. We also offered a sub-optimal algorithm with much lower computational complexity compared to the original algorithm. The modified algorithm produces images with comparable quality with those of the original version. As compared to results of reference [1], reference [8] and JPEG-LS our algorithm is either better in terms of reconstructed image quality or is better in terms of the compression ratio. The low-complexity version of the algorithm could be a suitable candidate for hardware implementation.

6. ACKNOWLEDGEMENT

We should express our gratitude to Dr. Pejman Khadivi and Mr. Seyed Ahmad Razavi for their constructive discussion on the subject.

7. REFERENCES

[1] Xiang Xie, et.al, "A New Near-Lossless Image Compression Algorithm Suitable For Hardware Design in Wireless Endoscopy System", in *Proc. ICIP 2005,* Italy, Vol. 1, pp. 1125-1128, 2005.

[2] C. S. Lee and H. W. Park, "Near-lossless/lossless compression of er-ror-diffused images using a two-pass approach," IEEE Trans. Image Proc. Vol. 12, no. 2, pp. 170-175, Feb. 2003.

[3] Sayood, K., *Introduction to Data Compression*, Morgan Kaufmann, Third Edition, 2005.

[4] N. Memon, N. Moayeri, "New Error Criterion For Near-Lossless Image Compression", in *Proc. ICIP'97*, USA, pp. 662-665, 1997.

[5] R. Iordache, I. Tabus, J. Astola, "Fixed-Slope Near-Lossless Context-Based Image Compression", in *Proc. ICIP'98*, USA, pp. 512-515, 1998

[6] P.K. Meher, T. Srikanthan, J. Gupta, H. K. Agarwal, "Near Lossless Image Compression Using Lossless Hartley Like Transform", in *Proc. ICSP 2003*, pp. 213- 217, 2003.

[7] Gonzalez R, Woods R., *Digital image processing*, Prentice Hall Book Co., 2002.

[8] K. Ligang, M.W. Marcellin**,** "Near-Lossless Image Compression: Minimum-Entropy, Constrained-Error DPCM", *IEEE Tran. Image Proc*., Vol. 7, no. 2, pp. 225 - 228, 1998.

Table 3. Comparison of our methods with reference [1] and JPEG-LS

Image (512x512)	LCGPSA (d=1)		LCGPSA (d=2)		Reference [1]		JPEG-LS (Near-LossLess)		Reference [8] (d=1)	
	PSNR	***bpp***	***PSNR***	***bpp***	***PSNR***	***bpp***	***PSNR***	***bpp***	***PSNR***	***bpp***
Lena	49.00	3.16	43.9	2.81	46.38	3.51	45.12	4.89	49.14	3.91
Lake	48.40	5.07	42.90	4.51	46.39	4.3	45.13	4.14	49.52	5.36
Baboon	48.37	5.2	42.68	4.9	46.38	4.8	52.38	6.69	49.37	5.18
Airplane	48.76	3.78	43.58	3.23	46.37	2.98	45.17	3.01	49.30	3.84
Peppers	48.27	4.56	42.58	4.2	46.53	3.53	45.24	5.23	49.71	4.93
House	48.89	2.15	43.64	2	46.40	3.48	45.15	3.9	49.44	1.74
Average	***48.61***	***3.98***	***43.21***	***3.60***	***46.4***	***3.76***	***46.36***	***4.64***	***49.41***	***4.16***