NEW SIMPLE CONTEXT-BASED PREDICTIVE TECHNIQUE FOR LOSSLESS IMAGE COMPRESSION

G. Ulacha[1] and R. Stasiński[2]

[1]Department of Computer Science, [2]Department of Electronics and Telecommunications
[1]Szczecin University of Technology, [2]Poznań University of Technology
[1]Żołnierska 49, Szczecin, Poland, [2]Piotrowo 3A, 60-965 Poznań, Poland
Phone: (+48 61) 665-2631 Fax: (+48 61) 665-2572 E-mail: gulacha@wi.ps.pl, rstasins@et.put.poznan.pl

ABSTRACT

An efficient and simple context-based data modelling technique for lossless image compression is described in the paper. Similarly as preprocessing stage of JPEG-LS, it uses only 3 contexts, which makes it time-efficient, and does not force the message headers to be long. Enhanced, but more computationally complex versions of the method are also analyzed. Extensive experiments show that indeed, the new technique is clearly better from data compression point of view than the preprocessing stage of JPEG-LS, while its computational complexity is approximately the same.

1. INTRODUCTION

It is widely known that appropriate signal modelling may significantly improve lossless image compression techniques [1]. The most obvious approach consists in coding error samples at the output of a linear predictor. More practical techniques use context depended image processing prior to coding, contexts are computed on the basis of local image properties calculated using the coded sample and few preceding it ones. They are only 3 contexts in JPEG-LS [2], and as much as 1024 in the method described in [3], other techniques place between these extremes [4, 5, 6, 7, 8]. Application of new techniques usually reduces the output signal zero order entropy, hence, potentially may improve performance of the coder, but on the other hand, increases its complexity. Moreover, large number of contexts results in increased side information to be sent in a message header, which may reduce the coding gain to nil, especially for small images.

In the paper the problem of proper balance between the header size and method efficiency is addressed. The proposed method is using only three contexts, section 2.4, but several measures have been undertaken to improve its performance. Firstly, rotation as an image preprocessing method is used, section 2.2. Secondly, optimisation of predictors has been done, sections 2.4 and 2.5. The method is compared to the MED algorithm, being data modelling stage of JPEG-LS, section 2.3. The reported in section 3 results of extensive experiments on 45 images show that indeed, the new preprocessing technique is clearly better than that for the JPEG-LS one, especially if a moderate increase in image modelling stage complexity is allowed.

2. TECHNIQUES

Generic data compression techniques like Huffman, or arithmetic coding do not deal with dependencies between source symbols very well. Then a good idea is to precede source coding by signal modelling phase, which goal is to diminish the dependencies. The best measure of such data preprocessing efficiency is the decrease in data zero order entropy:

$$H = -\sum_i p_i \cdot \log_2 p_i \tag{1}$$

where p_i are probabilities of source symbols, i is their index. The formula gives lower bound on average bit-per-symbol rate for static source coding (average codeword length for block codes), and for any coding technique if dependencies between source symbols are removed [1].

2.1 Linear prediction

The simplest technique for diminishing signal samples dependencies is to replace them by errors at the output of a linear predictor [1]:

$$e_n = x_n - \hat{x}_n \tag{2}$$

where x_n are data samples, and:

$$\hat{x}_n = \sum_{i=1}^{r} a_i \cdot x_{n-i} \tag{3}$$

is the predictor output, a_i are predictor coefficient, r its order. Predictor coefficients are usually computed by minimizing the mean square error criterion [1], being different than minimization of (1). Hence, it usually exists the optimal predictor order r, for which the zero order entropy is minimal. Probability distribution of e_n tends to be Laplacian, which results in good performance of simple source coders, e.g. the Rice-Golomb one.

In the two-dimensional case indexing of samples in (3) is based on the Euclidean distance between them and the coded one, and clockwise ordering of samples, the resultant pattern is shown in Fig.1. When predictive coding an image, the first sample is transmitted unchanged, while samples from the first column and row are coded using first

order difference. For predictor orders higher than 3 first row and column are repeated appropriate number of times, so that the formula (2) can be applied to remaining samples of an image.

		16	14	17		
	11	8	6	9	12	
15	7	3	2	4	10	18
13	5	1	0			

Figure 1 Samples location for indices i in (3), 0 indicates the location of x_n.

If prediction coefficients are optimized, then their values (and possibly the new predictor rank) are transmitted as side information. This increases the lower limit on bit-per-symbol rate (codeword length):

$$L = H + \frac{h}{X \cdot Y} \tag{4}$$

where h is the minimum message header length (e.g. for 14 bits per predictor coefficient $h = 14r$), H is the zero order entropy (1), X, Y are image dimensions in pixels. In this paper predictor coefficients are fixed-point 14-bit signed numbers, 12 bits are used for coding the fractional part of a coefficient.

2.2 Rotation

Image rotation by multiplicity of 90 degrees can be done without data modification, the same concerns its mirror-image reflection. There are 8 combinations of these operations, we name them 'phases' of image rotation. It has been shown that application of rotation for lossless image compression may result in decrease in the L value (4) by more than 0.05, on the other hand, the phase can be coded using only 3 bits [7,8]. Of course, in the worst situation the rotation phase optimization may increase the number of image processing steps 8 times.

2.3 Median edge detector (MED)

Median edge detector, or MED, is used as image preprocessing stage in JPEG-LS standard [2]. It is working in 3 different contexts described by the following condition on 3 samples preceding the coded one:

$$\hat{x}_{med} = \begin{cases} \min(x_{n-1}, x_{n-2}) \text{ for } x_{n-3} \geq \max(x_{n-1}, x_{n-2}) \\ \max(x_{n-1}, x_{n-2}) \text{ for } x_{n-3} \leq \min(x_{n-1}, x_{n-2}) \\ x_{n-1} + x_{n-2} - x_{n-3} \text{ otherwise} \end{cases} \tag{5}$$

The result of (5) is then predicted using a filter (3) of order $r = 14$ associated with the chosen context, finally, the prediction error is coded (2). The results of MED can be further improved by optimizing the predictors, and image phase, section 2.2. In our research MED technique is always combined with image phase optimization, which makes it somewhat more efficient than the original data modelling stage of JPEG-LS [7].

2.4 New contexts definition

The contexts of the method proposed in the paper are computed on the basis of weighted variance of the coded pixel neighborhood:

$$\sigma^2 = \frac{1}{q} \sum_{j=1}^{n} d_j \cdot (P(j) - \overline{p})^2 \tag{6}$$

where $P(j)$ are pixel values, d_j are their distances from the coded pixel:

$$d_j = \frac{1}{\sqrt{(\Delta x_j)^2 + (\Delta y_j)^2}} \tag{7}$$

weighted mean pixel value is

$$\overline{p} = \frac{1}{q} \sum_{j=1}^{n} d_j \cdot P(j) \tag{8}$$

finally

$$q = \sum_{j=1}^{n} d_j \tag{9}$$

In the basic version of the method the neighbourhood size is $n = 10$, thresholds defining 3 contexts are $0.15 \cdot \sigma^2$ and $1.3 \cdot \sigma^2$, and predictor rank for each context is $r = 14$. Additionally, image phase optimization is done, section 2.2. In an alternative approach it has been assumed that 50% of pixels belong to the lowest-energy context, while 15% to the highest-energy one, but as compression results appeared to be not better than those for the basic method, they are not reported here (actually, threshold values appeared to be similar to those for the basic method). Time complexity of the algorithm is proportional to that of MED, and in fact not much greater, which means that the method is fast. Somewhat more complex versions of the technique have been also considered, in their case predictor orders have been $r = 30$, and $r = 54$.

2.5 Method optimization

If the computational load for the coder does not matter, the most obvious way to improve the method performance consists in searching for predictor orders. The order may be the same for all 3 contexts, or optimized individually for each of them, the search range for it has been $r = 3, \ldots, 54$. Another parameter that can be optimized is the coded pixel neighbourhood size, the following values of n have been considered: 4, 5, 6, 10, 18, 24. Finally, thresholds optimization has been done. Individual optimization of each parameter did not improve the average bit-per-symbol rate of the method too much, results of their joint optimization has been somewhat better. It is then clear that results for practical algorithms described in previous sections are close to theoretical limits. The outline of the joint optimization procedure is given below, $prog_1$, $prog_2$ are thresholds, L is the mean bit-per-symbol rate (codeword length):

$prog_1 = 0.15 \cdot \sigma^2$; $prog_2 = 1.3 \cdot \sigma^2$;
For each n from the set {4, 5, 6, 10, 18} {
  For each phase (section 2.2) {
    For each r in the range from 3 to 54 {
      Find parameters resulting in the smallest L;
    }
  }
}
Find $prog_1$ resulting in the smallest L
  from $0.05 \cdot \sigma^2$ to $0.8 \cdot \sigma^2$ step $0.05 \cdot \sigma^2$;
Find $prog_2$ resulting in the smallest L
  from σ^2 to $3 \cdot \sigma^2$ step $0.1 \cdot \sigma^2$;
Once more, Find $prog_1$ resulting in the smallest L
  from $0.05 \cdot \sigma^2$ to $0.8 \cdot \sigma^2$ step $0.05 \cdot \sigma^2$;
For each predictor, i=1,2,3 {
  For each r_i in the range from 3 to 54 {
    Find r_i resulting in the smallest L;
  }
}

This purely research algorithm is asymmetric, the coder is much more computationally demanding, the suboptimal solution has been found after approximately 10 minutes of work of Pentium IV, 2.8 GHz, not optimized C++ code, This is much more than <1s for non-optimized algorithm.

3. EXPERIMENTS

The most important results of experiments are summarized in Table 1. The table contains lower limits on average codeword length of a hypothetical ideal static source block code used for coding the preprocessed gray-scale images, equal to data zero order entropy plus contribution due to header size (4). There are 45 test images: of size 512×512 (26 images, first section of Table 1), 720×576 (9 images, second section) and 256×256 (10 images, the remaining part of Table 1), their names can be found in the first column. The compared techniques are MED, the basic version of the new method, its two extended versions with predictors of rank $r = 30$, and $r = 54$, and its optimized version, the algorithm from section 2.5 has been used.

As can be seen from Table 1, compression results for the basic version of the new method are on the average 4.4% better than those for the MED (with phase optimization, which is not standard), additionally, for 37 from 45 images the new method appeared to be clearly better than MED. The average result is better than that for MED with optimized predictors, too (not reported in the Table 1): 4.36094 vs. 4.41503 bits-per-symbol. Further results exhibit the effect of diminishing returns, when using predictors of rank $r = 54$ the data volume diminishes by approximately 1%, full optimization squeezes them even more, but only by an additional 0.5%. It should be noted that the last result is almost as good as that for the fully optimized CALIC data modelling stage, reported in one of our previous papers: L=4.28814.

4. CONCLUSION

In the paper simple and efficient data modelling algorithm for image lossless compression has been described. The new method is dividing image pixels into only 3 groups (contexts), but new, different than in JPEG-LS contexts definition is proposed. The basic version of the algorithm is time-efficient, more complex, but enhanced from the data compression efficiency point of view versions of the method are also considered. The proposed algorithms have been applied to 45 test images, the results of exhaustive research are reported in the paper. It is shown that indeed, the redefinition of data modelling stage results in reduced signal zero order entropy, while the message header is kept short, and this is achieved either without sacrificing its low computational complexity, or at its moderate increase. The new technique is intended to be used in predictor blending approach, described in [3,9].

The research work presented in this paper was sponsored by Polish Ministry of Science and Higher Education (years 2007-2010).

REFERENCES

[1] K. Sayood, “Introduction to Data Compression”, 2nd edition, Morgan Kaufmann Publ., 2002.
[2] M. J. Weinberger, G. Seroussi, G. Sapiro, “LOCO-I: Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS”, IEEE Trans. on Image Processing, vol. 9, No. 8, pp. 1309-1324, August 2000.
[3] G. Deng, H. Ye, “A general framework for the second-level adaptive prediction”, Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP’03, vol. 3, pp. 237-240, 2003.
[4] X. Wu, N. D. Memon, “CALIC – A Context Based Adaptive Lossless Image Coding Scheme”, IEEE Trans. on Communications, vol. 45, pp. 437-444, May 1996.
[5] F. Golchin, K. K. Paliwal, “A lossless image coder with context classification, adaptive prediction and adaptive entropy coding”, Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP’98, Seattle, pp. 2545-2548, 1998.
[6] I. Matsuda, N. Ozaki, Y. Umezu, S. Itoh, “Lossless coding using Variable Blok-Size adaptive prediction optimized for each image”, Proceedings of 13th European Signal Processing Conference, EUSIPCO-05, Antalya, Turkey, CD, 2005.
[7] G. Ulacha, R. Stasiński, “On context-based predictive techniques for lossless image compression”, Proceedings of 12th International Workshop on Systems, Signals & Image Processing, IWSSIP 2005, Chalkida, Greece, pp. 345-348, 2005.
[8] G. Ulacha, R. Stasiński, “A New Context-based Lossless Image Coding Method”, Proceedings of 13th International Conference on Systems, Signals and Image Processing, IWSSIP 2006, Budapest, pp. 215-218, 2006.
[9] T. Seemann, P. Tisher, “Generalized locally adaptive DPCM”, Department of Computer Science Technical Report CS97/301, Monash University, Australia, pp. 1-15.

Table 1. Mean bit-per-symbol rates (codeword length) for image preprocessing methods described in the paper.

Images	MED	Basic method	Extended, $r = 30$	Extended, $r = 54$	Optimized method
Aerial	5.31868	5.14270	5.11175	5.10725	5.10431
Airfield	5.26946	5.03179	5.02680	5.02335	5.01476
Airplane	4.20149	4.05718	4.03498	4.02863	4.02490
Baboon	6.26367	6.13072	6.12184	6.11528	6.11110
Barbara	5.47153	4.98384	4.90306	4.82539	4.82366
Boat	5.07865	4.72555	4.72758	4.72265	4.70462
Bridge	3.76691	3.73356	3.72373	3.71972	3.71296
Couple	4.42892	4.45817	4.44239	4.43670	4.43264
Crowd	4.36868	4.15684	4.12892	4.12196	4.11606
Elaine	5.32891	4.77667	4.60104	4.51522	4.49123
Finger	5.64343	5.22100	5.20717	5.20131	5.20010
Frog	4.95335	5.09474	5.11605	5.11597	5.08473
Goldhill	4.87596	4.80374	4.77635	4.76116	4.75501
Harbour	4.99900	5.05952	5.03658	5.03220	5.02616
Lax	5.97569	5.81835	5.82034	5.81364	5.80215
Lennagrey	4.53063	4.25967	4.25770	4.25430	4.24657
Lena_{TMW}	4.88515	4.61425	4.61400	4.61174	4.59976
Man	4.79789	4.71869	4.71159	4.70622	4.70321
Peppers	4.93692	4.52017	4.51395	4.48754	4.47102
Sailboat	5.16502	4.87342	4.83388	4.78715	4.77658
Seismic	2.94005	2.26174	2.19512	2.08351	2.07530
Shapes	1.26214	2.01070	1.92561	1.95959	1.68445
Tank	4.10458	3.97780	3.97502	3.97260	3.96513
Truck	4.40911	4.25527	4.25336	4.25311	4.24806
Woman1	4.41667	4.29582	4.27722	4.26986	4.26299
Woman2	3.56471	3.30520	3.30709	3.29326	3.28065
Balloon	3.11800	2.90828	2.87223	2.83889	2.82811
Barb	5.20188	4.72454	4.65369	4.59264	4.58624
Barb2	5.18074	4.90895	4.84949	4.82193	4.80809
Board	3.94716	3.71884	3.66395	3.63589	3.62234
Boats	4.28693	4.05778	4.02880	4.00766	3.98499
Girl	4.20591	3.86004	3.82169	3.79736	3.78832
Gold	4.71463	4.64511	4.61721	4.59627	4.59115
Hotel	4.73083	4.55610	4.53321	4.52138	4.51958
Zelda	4.11242	3.74647	3.72163	3.69712	3.67044
Bridge256	5.88236	5.87347	5.87403	5.88148	5.85538
Camera	4.73620	4.77432	4.76497	4.77527	4.73627
Couple256	3.98628	4.04700	4.04135	4.04597	4.02751
Earth	3.59090	3.73726	3.73747	3.79116	3.70513
Elif	3.33296	2.90761	2.87566	2.82444	2.81909
Noisesquare	5.72510	5.25952	5.24139	5.24865	5.17663
Omaha	5.99805	6.44411	6.42530	6.45209	6.31384
Sena	3.63380	3.16760	3.12163	3.06912	3.05175
Sensin	3.93242	3.47096	3.41014	3.36437	3.33938
Sinan	3.65756	3.14728	3.10608	3.04288	3.03554
Mean	4.55403	4.36094	4.33340	4.31609	4.29284