Enhanced Lossless Image Coding Methods Based on Adaptive Predictors

Grzegorz Ulacha
Department of Computer Science
West Pomeranian University of Technology
Szczecin, Poland
gulacha@wi.ps.pl

Ryszard Stasiński
Faculty of Electronics and Communications
Poznań University of Technology
Poznań, Poland
rstasins@et.put.poznan.pl

Abstract— In the paper ultimately efficient image lossless coding methods are described. They are based on least-squares adaptive prediction, possibly combined with normalized LMS algorithm. Their performance is comparable to that of TMW^{LEGO}, and MRP 0.5 techniques, while their computational complexities are smaller than those for the two reference methods. The new algorithms are supported by an advanced adaptive context arithmetic coder, also described in the paper.

Keywords-lossless image coding; adaptive predictors; arithmetic coding.

I. Introduction

The construction of CALIC algorithm in 1996 [1] was an achievement, and still this is one of the best image lossless compression methods. The popularized with it context coding has been used in other highly efficient techniques, e.g. LOCO-I, which modified version has been implemented as the kernel of JPEG-LS standard [2]. At that time being better than LOCO-I the CALIC algorithm was too complex to implement in a standard. Situation changed since that time, and the place of CALIC has been taken by even better but much more sophisticated techniques like TMW^{LEGO} (2001) [3], WAVE-WLS (2002) [4] and the newest version of MRP 0.5 named VBS & new-cost (2005) [5]. Except for MRP these algorithms are based on a highly promising predictor blending approach. The wavelet transforms have been also used in lossless coding, a prominent example being JPEG2000 [6], however, this approach does not appear to be particularly efficient so far.

In the paper very effective while not as complex as e.g. MRP 0.5 lossless coding techniques are presented. They are based on adaptive predictors: weighted least-squares (WLS), section III.A, averaged WLS (AVE-WLS), section III.B, and combination of the two with normalized LMS (NLMS), section III.C. The adaptive context arithmetic coder is used, section IV. When predictor rank and neighborhood size are reasonable, complexities of new algorithms are much lower than that of MRP 0.5, section V, while their performance almost as good, and for the most sophisticated solution even better than that of TMW^{LEGO}, section VI.

II. Lossless Coding Basics

A natural image cannot be interpreted as a memoryless data source. That is why lossless coding techniques start with data modeling stage reducing pixel dependencies. An obvious measure for its efficiency is decrease in zero-order entropy:

$$H = -\sum_{i=e_{\min}}^{e_{\max}} p_i \cdot \log_2 p_i \;, \tag{1}$$

where p_i is the probability of i-th source symbol, the range of i is from $e_{\min}$ to $e_{\max}$. Zero-order entropy provides the lower limit on the average number of bits required for coding a source symbol by a (non-adaptive) entropy coding method [7].

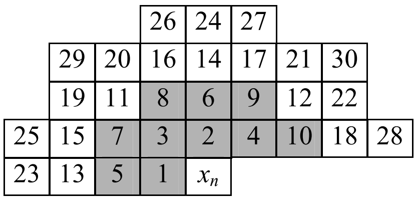

Figure 1. Numbering of neighborhood pixels.

The best image modeling techniques are usually based on linear predictors. A linear predictor generates expected value of a coded pixel:

$$\hat{x}_n = \sum_{j=1}^{r} b_j \cdot P(j) \;, \tag{2}$$

where $P(j)$ are pixels "preceding" the coded pixel x_n, b_j are predictor coefficients, and r is its rank. Pixel indexing depends on acquisition scheme, for raster scan only pixels above and to the left are considered, Fig.1. The coded pixel estimate (rounded to closest integer) is subtracted from its actual value, and the result (i.e. prediction error e_n) coded:

$$e_n = x_n - \hat{x}_n \,. \tag{3}$$

The greater the rank of a predictor the better the prediction (but usually only to some limit), however, the most distant pixels from the coded one provide the worst contribution to its estimate. It is then reasonable to order neighboring pixels in accordance with Euclidean distance from x_n, pixels having the same distance are numbered clockwise. Fig.1 shows pixel indices used in (2) for the first thirty j values.

III. Predictor Adaptation Algorithms

A great advantage of adaptive predictors is that no side information about their coefficients is needed. Moreover, their ability to follow local image properties may result in superior performance, better than for MMSE optimized ones. The main drawback is relatively high computational cost, both on the coder and decoder sides.

In this section three types of adaptive predictors are described: weighted LS (WLS), its variant AVE-WLS, and normalized LMS (NLMS). It is suggested to apply the NLMS predictor to output data from either WLS, or AVE-WLS ones.

A. *Weighted Least-Squares (WLS)*

Vector of least-squares predictor coefficients **B** is computed from the following formula:

$$\mathbf{B} = \mathbf{R}^{-1} \cdot \mathbf{P}\,, \tag{4}$$

where **R** is the “experimental” signal autocorrelation matrix:

$$\mathbf{R}(j,i) = \sum_{y\in Q}\sum_{x\in Q} \psi_{(y,x)} \cdot P_{(y,x)}(i) \cdot P_{(y,x)}(j)\,, \tag{5}$$

vector **P** is:

$$\mathbf{P}(j) = \sum_{y\in Q}\sum_{x\in Q} \psi_{(y,x)} \cdot P_{(y,x)}(0) \cdot P_{(y,x)}(j)\,, \tag{6}$$

pixels P are taken from a training neighborhood Q of the coded pixel, its definition from [8] is shown in Fig.2, Ψ is a weighting function, for trivial weighting function (Ψ = 1) WLS reduces to the OLS method [8] (Q is an area extending W pixels to the left and up from the coded one, and to the right in rows preceding it). We are proposing an improved definition of the weighting function, compare [8]:

$$\psi = \frac{1}{350 + \sum_{j=1}^{m} \left(\overline{d}_j \cdot (P(j) - P_{\text{off}}(j))\right)^2}\,. \tag{7}$$

The definition precludes application of fast predictor update formulas, hence, WLS technique is significantly more complex than OLS, see section 6. Moreover, optimal parameter W defining extent of area Q around the coded sample should be greater for WLS method than for OLS one.

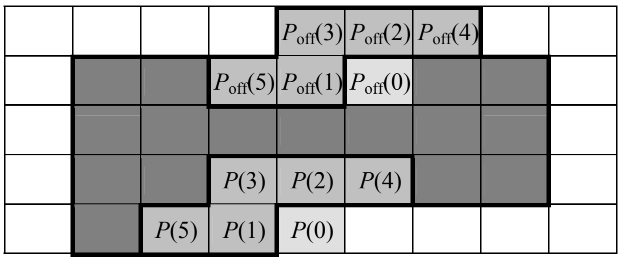

Figure 2. Example illustrating possible location of patterns (one shown) inside neighborhood region Q of pixel $P(0)$ for m = 5, W =3.

B. *AVE-WLS Method*

For each pixel it exists an optimal predictor rank [9]. It is not known a priori, so we propose to compute all predictor vectors for ranks from $r_{\min}$ to $r_{\max}$, but instead of searching for the best predictor we sum them up:

$$\mathbf{B}_{\text{ave}} = \frac{1}{r_{\max} - r_{\min} + 1} \cdot \sum_{j=r_{\min}}^{r_{\max}} \mathbf{B}_j\,. \tag{8}$$

Note that lengths of vectors $\mathbf{B}_j$ should be extended to that of $\mathbf{B}_{\max}$, we are implementing zero-padding here. A modified weighting function is implemented:

$$\psi_{(y,z)} = \frac{0.25 + 0.8^{\sqrt{(\Delta y)^2 + (\Delta x)^2}}}{350 + \sum_{j=1}^{m} \left(\overline{d}_j \cdot (P(j) - P_{\text{off}}(j))\right)^2}\,, \tag{9}$$

where $\sqrt{(\Delta y)^2 + (\Delta x)^2}$ is the Euclidean distance between pixels $x_n = P(0)$ and $P_{\text{off}}(0)$, see Fig.2.

The parameter W has been set to 14, $r_{\min}$ and $r_{\max}$ depended on image sizes: for 256x256 pixel images they were 4, and 22, respectively, for 512x512 pixel ones the range was from 4 to 28, and for the 720x576 ones the ranks were 6 and 24. In an extensive test on 45 images the mean bit per pixel value for AVE-WLS was equal to 3.94281, and was by 0.01075 bit lower than for WLS.

C. *Normalized LMS algorithm*

The coefficient update formula for LMS algorithm is:

$$b_j(n+1) = b_j(n) + \mu \cdot \overline{e} \cdot P(j)\,, \tag{10}$$

where:

$$\overline{e} = \operatorname{sgn}(e) \cdot \min\{|e|, \varphi\} \tag{11}$$

is the bounded prediction error, the bound is equal to φ. In NLMS approach the learning coefficient μ is variable, we propose the following formula for it:

$$\mu = \frac{1}{2^7 \cdot \left(10 + \sum_{j=1}^{r_{\text{NLMS}}} e^2(j)\right)}\,, \tag{12}$$

where $e(j)$ is the j-th signal sample, here – the prediction error from the preceding stage of algorithm.

We propose to implement NLMS in an auxiliary signal modeling stage: after an image is processed by WLS or AVE-WLS algorithm NLMS prediction is applied. The additional computational time due to NLMS is only 2 sec (r_{NLMS} = 72), much less than few hundreds seconds for AVE-WLS.

IV. Arithmetic Coder

A. Adaptation of probability distribution

When coding absolute values of numbers probability distributions similar to one-sided Laplace one can be expected. Its initial approximation is as follows:

$$\mathbf{N}_e(i) = \lfloor A \cdot 0.8^i \rfloor + 1 . \qquad (13)$$

where i is the number of histogram slot, a reasonable choice for A is A = 10. Occurrences of absolute prediction error values $|e|$ are collected in the histogram. A forgetting mechanism is introduced, if the number of coded $|e|$ exceeds 2^s, histogram values are divided by 2:

$$\mathbf{N}_e(i) = \left\lfloor \frac{\mathbf{N}_e(i)}{2} \right\rfloor + 1, \quad \text{for all i from 0 to emax.} \qquad (14)$$

The occurrence counter is set to the sum of histogram slots. Good results are obtained for $s \geq 10$, we use s = 13, and s = 10 for sign and quantization error coding.

B. Contexts for probability distributions

Prediction error distributions may depend on the $|e|$ values surrounding the coded pixel. These errors may be used for defining a context pointing out at the best probability distribution for coding the current $|e|$. An important factor is the decision rule for choosing the context. Our approach is based on paper of Deng [10]. We need several parameters for that, the ω_1 one is rounded up after evaluation:

$$\omega_1 = \max\{2|e(1)|, 2|e(2)|, \frac{9}{8}(|e(3)|+|e(4)|), |e(5)|+|e(10)|, |e(6)|+|e(7)|, \frac{13}{8}|e(4)|, \frac{3}{2}|e(3)|, \frac{7}{8}(|e(8)|+|e(9)|), \frac{11}{8}(|e(1)|+|e(2)|)\}$$

The next parameter is, see [5]:

$$\omega_2 = \frac{1}{\delta}\sum_{j=1}^{n} \bar{d}_j \cdot |e(j)| , \qquad (16)$$

where n = 28, and $\delta = \sum_{j=1}^{n} \bar{d}_j$. Then, a parameter is chosen:

$$\omega_3 = \max\{2 \cdot \omega_1, 10 \cdot \omega_2\} . \qquad (17)$$

Refinement of the context choice is based on:

$$\omega_4 = \max\{|P(1)-P(3)|, |P(2)-P(3)|, |P(1)-P(2)|, 1.1 \cdot |P(2)-P(4)|, 0.8 \cdot |P(1)-P(4)|, 0.9 \cdot |P(3)-P(4)|\} \qquad (18)$$

Finally we compute:

$$\omega = \omega_3 + 0.48 \cdot \omega_4 \qquad (19)$$

In the case of 16 contexts the ω value is quantized using 15 thresholds: $\mathbf{T}_h$ = {3, 8, 14, 20, 27, 34, 43, 55, 66, 80, 100, 120, 150, 180, 240}.

C. Quantization of prediction errors

An obvious method for increasing the histogram collection rate is to decrease the number of its slots, i.e. to quantize slot numbers. In fact $|e|$ values are divided into 2 parts, the quantized one and the residue coded separately. Quantization and coding are done as follows.

Find the quantized histogram slot number k from relation $\mathbf{T}(k) \leq |e| < \mathbf{T}(k+1)$ for quantization thresholds $\mathbf{T}$ = {0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24, 32, 64, 128}. The k value is context coded using approach from section IV.B. Quantization error e_q is treated as $\mathbf{q}(k)$-bit number, numbers of bits are $\mathbf{q}$ = {0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 3, 5, 6, 7}. If $\mathbf{q}(k) > 0$, then e_q value is processed by one of 7 universal adaptive arithmetic coders.

Decoding consists in reading the k value indexing threshold table $\mathbf{T}$ and bit number table $\mathbf{q}(k)$. If $\mathbf{q}(k) > 0$, then e_q value is restored, otherwise e_q = 0. Finally, absolute value of prediction error is computed $|e| = \mathbf{T}(k) + e_q$.

D. Prediction error sign coding

The prediction error sign is coded using 16-context arithmetic coder. Initially two-slot context histograms are set to value 5. Two first bits of context number are error signs for left and upper neighbors (1 and 2 in Fig.1). The remaining two bits are obtained by quantizing the main context number from section IV.B, thresholds are {8, 20, 180}.

V. Algorithms Computational Complexity

Execution times of experimental computer programs are reported here. All programs were written in ANSI C, have not been optimized, and tested on Pentium 4 with 2.8 GHz clock frequency.

The main computational burden of WLS method is calculation of matrix **R**. It is done for each pixel and its cost is $O(W^2r^2)$. It appears that it takes 96.1% of the algorithm time. Then, the extra matrix inversion in AVE-WLS does not add too much computational time, at least for $r < 20$, e.g. for $r_{\max} = 18$ the difference is 7%, see Fig.3. Note that matrix **R** is calculated only for $r_{\max}$, for smaller r its reduced version is used. For comparison, computational time for the MRP method and 512x512-pixel Lenna was 1228 second.

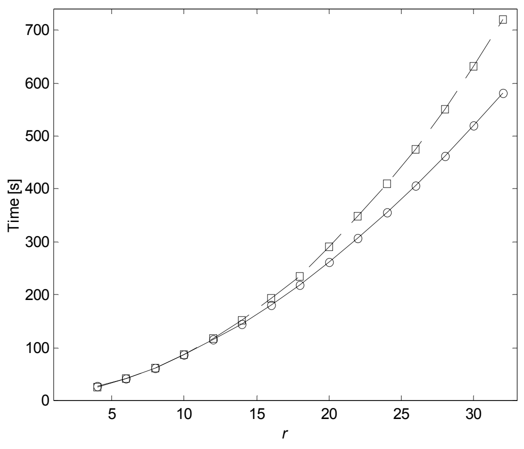

Figure 3. Execution time as a function of prediction rank r for WLS (continuous line), and AVE-WLS (dashed line) for neighborhood parameter W=14, Lenna image.

VI. EXPERIMENTS

Results of experiments are summarized in Table 1. As can be seen, when increasing complexity of algorithms proposed in this paper we are approaching the performance of two currently the best lossless coding methods, TMW$^{\text{LEGO}}$, and MRP 0.5. In fact the bitrate of the most sophisticated method presented in the paper, AVE-WLS-NLMS, is in the midway between the two. On the other hand, execution time for our algorithm is much lower than for TMW$^{\text{LEGO}}$ or MRP 0.5, for maximum predictor rank $r_{\max} = 24$ it takes only 37.2% of time needed for MRP 0.5 to process Lenna image.

As can be seen, any of the compared coding methods outperforms by a wide margin any of widely used lossless standards, results for other JPEGs aren't much better, if better at all.

VII. CONCLUSION

In the paper one of the most efficient while not the most computationally complex image lossless coding methods are presented. The algorithms use adaptive LS predictors, an extra stage of NLMS prediction is proposed. Entropy coding is done by an advanced adaptive context arithmetic coder. Experiments show that indeed, the best coder is approximately as good as TMW$^{\text{LEGO}}$ and MRP 0.5 ones.

REFERENCES

[1] Wu, X., Memon, N.D.: CALIC – A Context Based Adaptive Lossless Image Coding Scheme. IEEE Trans. on Communications 45, 437—444 (1996).

[2] Weinberger, M.J., Seroussi, G., Sapiro, G.: LOCO-I: Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS. IEEE Trans. on Image Proces. 9, pp. 1309-1324 (2000).

[3] Meyer, B., Tischer, P.: TMWLego - An Object Oriented Image Modeling Framework. Proc. Data Compression Conf., p. 504 (2001).

[4] Ye, H.: A study on lossless compression of greyscale images. PhD thesis, Dept. Electronic Eng., La Trobe University (2002).

[5] Matsuda, I., Ozaki, N., Umezu, Y., Itoh, S.: Lossless Coding Using Variable Blok-Size Adaptive Prediction Optimized for Each Image. Proc. EUSIPCO-05 (on CD), (2005).

[6] Marcellin, M., Gormish, M., Bilgin, A., Boliek, M.: An Overview of JPEG-2000. Proc. Data Compression Conf., pp. 523—541 (2000).

[7] Sayood, K.: Introduction to Data Compression, 2nd edition. Morgan Kaufmann Publ. (2002).

[8] Ye, H., Deng, G., Devlin, J.C.: A weighted least squares method for adaptive prediction in lossless image compression. Proc. Picture Coding Symp., pp. 489—493 (2003).

[9] Aiazzi, B., Baronti, S., Alparone, L.: Near-lossless image compression by relaxation-labeled prediction. Signal Processing 82, pp. 1619-1631 (2002).

[10] Deng, G., Ye, H.: Lossless image compression using adaptive predictor combination, symbol mapping and context filtering. Proc. IEEE ICIP-99, vol. 4, pp. 63-67 (1999).

TABLE I. BIT RATE VALUES FOR IMAGE LOSSLESS CODING ALGORITHMS COMPARED IN THE PAPER

Images	**JPEG-LS** [2]	**OLS** [8] W = 15, r = 18	**OLS** W = 10, r = 14	**WLS** [8]	**WLS** W = 14, r = 18	**AVE-WLS** W = 14, r = 6-24	**AVE-WLS-NLMS** W = 14, r = 6-24	**TMW$^{\text{LEGO}}$** [3]	**MRP 0.5** [5]
Balloon	2.889	2.690	2.694	2.64	2.658	2.644	2.602	2.60	2.579
Barb	4.690	3.939	3.906	3.90	3.796	3.781	3.759	3.84	3.815
Barb2	4.684	4.310	4.293	4.25	4.217	4.211	4.200	4.24	4.216
Board	3.674	3.388	3.378	3.33	3.318	3.312	3.293	3.27	3.268
Boats	3.930	3.638	3.628	3.58	3.566	3.559	3.537	3.53	3.536
Girl	3.922	3.576	3.558	3.50	3.489	3.482	3.461	3.47	3.465
Gold	4.475	4.273	4.272	4.25	4.241	4.239	4.231	4.22	4.207
Hotel	4.378	4.162	4.127	4.08	4.054	4.051	4.036	4.01	4.026
Zelda	3.884	3.549	3.561	3.53	3.537	3.526	3.522	3.50	3.495
Mean	**4.058**	**3.725**	**3.713**	**3.673**	**3.653**	**3.645**	**3.627**	**3.631**	**3.623**