Highly Effective Predictor Blending Method for Lossless Image Coding

Grzegorz Ulacha
Faculty of Computer Science and Information Systems
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—A fast and effective predictor blending method is described in the paper. The special set of 17 blended predictors is defined, including GAP, and (novel) texture context mapping one. An advanced error bias removal method is applied to some subpredictors. A sophisticated adaptive context arithmetic coder forms the final stage of the algorithm. Experimental results show that indeed, for considered execution times (few seconds for Lenna image) the presented technique is better than other state-of-the-art methods.

I. INTRODUCTION

Although the domain of entropy coding is quite matured, the era of modern lossless image coding begins at the end of the previous century. In 1996 the CALIC algorithm was described [1], the first widely recognized context coding method. Another important context coding technique of that time was LOCO-I, applied in JPEG-LS standard [2]. Being the best CALIC algorithm was too computationally complex to implement in a standard. The situation changed, and the place of CALIC was taken by better but much more complex techniques like TMW [3], WAVE-WLS [4] and the new version of MRP 0.5 – VBS & new-cost [5]. Except for MRP the algorithms are based on a very interesting predictor blending approach [8], [10], [12], [14], [15], [17], [19]. Wavelet transforms have been applied to lossless coding [6], too, however, this approach doesn’t appear to be particularly efficient so far.

In the paper a lossless image coding method based on predictor blending having medium computational complexity and high coding efficiency is presented (it is 3 times faster than GLICKBAWLS [20], while having similar coding efficiency, section IV). A new, sophisticated approach to blending of seventeen predictors is undertaken (variable neighborhood sizes), such subpredictors as GAP, [9], and texture context mapping [11] are included, section II, hence, it is proposed to name it Blend-17. The applied texture context mapping (TCM) method is completely new. An elaborated prediction error bias removal technique [13] has been applied to selected subpredictors, only. An advanced adaptive context arithmetic coder is presented in section III. Experiments show that indeed, the new algorithm and its simplified version are the best in their class, section IV.

II. ALGORITHM

A. Predictors

Entropy coders aren’t particularly efficient in coding of sources with memory. That is why modern lossless coding techniques begin with data modeling stage intended for minimization of mutual information between signal samples. If the mutual information is completely removed, then the zero-order entropy is becoming the correct entropy measure [7], it’s definition is:

\[ H = - \sum_{i=r_{\text{min}}}^{r_{\text{max}}} p_i \log_2 p_i \]  \hspace{1cm} (1)

where \( p_i \) is the probability of \( i \)-th symbol, \( i \) range is from \( r_{\text{min}} \) to \( r_{\text{max}} \).

The best data modeling algorithms are based on predictors. A linear predictor of rank \( r \) is used for estimating \( x_n \) sample value:

\[ \hat{x}_n = \sum_{j=1}^{r} b_j P(j) \]  \hspace{1cm} (2)

Figure 1. Indices of neighboring pixels.
\(P(j)\) are “previous” sample values, \(b_j\) are prediction coefficients [7]. The estimate is used to calculate the prediction error (rounded up in lossless coding):

\[
e_s = x_n - \hat{x}_s
\]

(3)

Figure 1 shows 30 pixels “preceding” the coded one \(x_n\). The higher is the pixel number, the lower its importance for estimation, thus a predictor sums up pixels from 1 to \(r\).

B. Predictor blending

A predictor blending method is defined by a set of partial predictors, called subpredictors. Low implementation complexity induces that simple constant predictors of rank from 1 to 3 are usually applied. For a predictor of rank 1 the estimate is simply a neighboring pixel value \(P(i), i\) is the pixel number, see Fig 1. Widely used slightly more complex subpredictors are [8]: GradWest = \(2P(1) - P(5)\), GradNorth = \(2P(2) - P(6)\), Plane = \(P(1) + P(2) - P(3)\), Plane2 = \(P(1) - P(2) + P(4)\), and an improved version of the non-linear predictor from the CALIC algorithm, GAP, presented in [9]. A list of subpredictors used for methods considered in this paper is given in Table 1. Blend-13 method is a hardware-oriented, hence simplified, data modeling algorithm from [10], our version of TCM approach [11] is described in section II.D below.

Highly efficient predictor blending methods implement adaptive schemes for selecting the (locally) dominant subpredictor. The dominant subpredictor is selected on the basis of minimum prediction error power criterion. The neighborhood error \(E_i\) for \(i\)-th subpredictor is:

\[
E_i = 1 + \sum_{j=1}^{n_i} d_j \cdot |v_{i,j}(j)|,
\]

(4)

where \(v_{i,j}(j)\) is its prediction error obtained \(j\) positions “before” the current prediction, see Fig.1. Then, weights \(w_j\) are calculated:

\[
w_j = \alpha_i \cdot \left( \frac{\delta_i}{E_i} \right)^2
\]

(5)

importance factors \(\alpha_i\) can be found in Table 1, and:

\[
\delta_i = \sum_{j=1}^{n_i} d_j,
\]

(6)

where \(d_j = 1/\sqrt{(\Delta x_j)^2 + (\Delta y_j)^2}\) is the inverse of Euclidian distance between pixels \(P(j)\) and \(P(0)\), and \(n\) is the neighborhood size. The weights of subpredictors sum up to 1, hence, they are normalized (they are \(\nu\) subpredictors):

\[
a_i = \frac{w_j}{\sum_{j=1}^{\nu} w_j}.
\]

(7)

Finally, the formula for the main (dominant) predictor is:

\[
\hat{x} = \sum_{i=1}^{\nu} a_i \cdot \hat{x}_i.
\]

(8)

As can be seen, predictor blending consists in linear combining \(\nu\) subpredictors, which weights \(a_i\) depend on the prediction error level from the closest neighborhood. An important question is how big this neighborhood should be, unfortunately, any value from an analyzed range is possible, e.g. in our research \(k\) varied from 3 to 30. In other papers fixed values have been proposed: \(k = 3\) [8], \(k = 4\) [12], \(k = 10\) [10] (Blend-13, see gray region in Fig.1). In this paper we propose individual \(k\) values, \(k_i\), provided in the last column of Table 1.

C. Contexts and predictor error bias correction

The idea of context become popular since the introduction of the CALIC algorithm [1], where, apart from the GAP prediction method (7 contexts), 576 contexts for correcting the prediction error bias have been introduced (context-based prediction error correction). A context is determined by some individual features of the closest neighborhood of the pixel to be coded.

A predictor introduces slowly changing error component, named prediction error bias. That is why adaptive methods for error bias cancellation are applied both in CALIC and JPEG-LS algorithms. For every context cumulated prediction error is determined and current prediction error corrected [1], [2]. Differences between methods are small, JPEG-LS version is better for hardware realization, as it has neither multiplications nor divisions.

<table>
<thead>
<tr>
<th>Subpredictors</th>
<th>Blend-13</th>
<th>(\alpha) Blend-13</th>
<th>Subpredictors</th>
<th>Blend-17</th>
<th>(\alpha) Blend-17</th>
<th>(k_i) Blend-17</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAP</td>
<td>1.0</td>
<td>Plane</td>
<td>1.0</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GradWest</td>
<td>2.0</td>
<td>Plane2</td>
<td>1.5</td>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GradNorth</td>
<td>2.0</td>
<td>GradWest</td>
<td>1.5</td>
<td>36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Plane</td>
<td>1.0</td>
<td>GradNorth</td>
<td>2.0</td>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Plane2</td>
<td>1.5</td>
<td>(P(1))</td>
<td>1.0</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(P(1))</td>
<td>1.0</td>
<td>(P(2))</td>
<td>1.0</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(P(2))</td>
<td>1.0</td>
<td>(P(3))</td>
<td>1.0</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(P(3))</td>
<td>1.0</td>
<td>(P(4))</td>
<td>1.0</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(P(4))</td>
<td>1.0</td>
<td>(P(5))</td>
<td>1.0</td>
<td>26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(P(5))</td>
<td>1.0</td>
<td>(P(6))</td>
<td>1.0</td>
<td>26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(P(10))</td>
<td>1.0</td>
<td>(P(10))</td>
<td>1.0</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(P(18))</td>
<td>1.0</td>
<td>(P(18))</td>
<td>1.0</td>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(P(28))</td>
<td>1.0</td>
<td>(P(28))</td>
<td>1.0</td>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>2 (P(3) - P(1))</td>
<td>1.0</td>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>GAP</td>
<td>1.0</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>TCM</td>
<td>1.5</td>
<td>30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In this paper both methods have been implemented in 4 variants each, then all 8 components summed up (\(C_{\text{mix}}\) value below), detailed description of the approach can be found in [13]. The pixel estimate is then:

\[
\hat{x} = \hat{x} + C_{\text{mix}}. \tag{9}
\]

In contrast to the suggestion from [14], the correction has been done only for prediction errors of the main predictor, the GAP, and the TCM one, only. Namely, it appeared that when applied to other subpredictors the approach increased the total prediction error.

D. Texture Context Matching (TCM)

The texture context matching technique is particularly well suited for coding of periodic patterns, like stripes, chequered patterns etc [11]. Then, it is not surprising that the idea consists in finding matching neighborhoods, one around currently coded pixel, and one around some other, \(P_{\text{off}}(0)\), see Fig.2. The latter pixel is searched for in a training window \(Q\) extending \(W\) rows above \(P(0)\), the rows are \(2W+1\) pixels long and centered around column containing \(P(0)\), additionally, \(W\) pixels immediately preceding \(P(0)\) are included. \(P_{\text{off}}(0)\) is used as an estimate of \(P(0)\).

In the paper the Manhattan measure is used for evaluating distance between patterns, \(m\) is a pattern size:

\[
\Delta = \frac{1}{m} \sum_{i=1}^{m} P(i) - P_{\text{off}}(i). \tag{10}
\]

The smaller the distance, the more probable is that \(P(0)\) is similar to \(P_{\text{off}}(0)\). Nevertheless, the choice of \(W\) and \(m\) values is not trivial. We are proposing two TCM models, a linear combination of their outputs forms the subpredictor output, the pattern size is fixed, \(m = 22\). In the first model we are seeking for the global minimum of slightly modified distance measure:

\[
\Delta = \sum_{i=1}^{m} \overline{d}_i \cdot |P(i) - P_{\text{off}}(i)|, \tag{11}
\]

where \(\overline{d}_i\) is the inverse of Euclidian distance between pixels, see the comment to (6). The obtained predictor is denoted \(P_{\text{global}}\). In the second model the best predictors \(P_W\) for all neighborhood sizes \(Q\) from 1 to some maximum \(W_{\text{max}}\) are computed, and then summed up as follows:

\[
P_{\text{mix}} = \frac{\sum_{w=1}^{W_{\text{max}}} \frac{1}{1+\Delta^2_w} \cdot P_w}{\sum_{w=1}^{W_{\text{max}}} \frac{1}{1+\Delta^2_w}}, \tag{12}
\]

Finally, the two predictors are combined as follows:

\[
P_{\text{TCM}} = 0.2 \cdot P_{\text{global}} + 0.8 \cdot P_{\text{mix}}. \tag{13}
\]

III. ARITHMETIC CODER

A. Adaptation and quantization of histogram

Probability distribution for absolute number values is usually close to one-sided Laplace one. Its initial approximation is:

\[
n_e(i) = \left[A \cdot 0.8^i\right]+1, \tag{14}
\]

where \(i\) is a number of histogram slot, a good choice is \(A = 10\). Absolute prediction error value occurrences are counted in the histogram. A forgetting mechanism is implemented, if the number of errors \(|e|\) exceeds \(2^s\), histogram values are halved:

\[
n_e(i) = \left[n_e(i) \right]+1, \text{ for all } i \text{ from } 0 \text{ to } e_{\text{max}}. \tag{15}
\]

The coded errors 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.

They are 16 histograms, one for each context, hence, filling some of them may be slow. For increasing histogram collection rate its slots number can be decreased by quantization. Namely, \(|e|\) values are divided into 2 parts, the quantized one and the residue, coded separately. Quantization and coding are done as follows.

Find a new index for slot \(k\) from table \(T:\)

\[
T(k) \leq |e| < T(k+1) \text{ for thresholds } T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24, 32, 64, 128\}. \text{ Result is context coded, see section III.B.}
\]

If \(q(k) > 0\), \(e_q\) is coded by one of 7 universal adaptive arithmetic coders.

Decoding consists in reading the table entry \(T(k)\), and if \(q(k) > 0\), then \(e_q\) is restored, otherwise \(e_q = 0\). Then,}

\[
|e| = T(k) + e_q.\]
B. Contexts for probability distributions

Absolute error values $|e(i)|$ for estimates from the coded sample neighborhood are used for determination of one of 16 contexts. Our approach is based on papers [15] and [5]. Firstly, the value $\omega_1$ is evaluated:

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

(16)

Then, in accordance with [5], $\omega_2$ is computed, (6):

$$\omega_2 = \frac{1}{\delta} \sum_{j=1}^{\delta} \left| e(j) \right|, \quad (17)$$

$n = 28$. The greater of two values is chosen:

$$\omega = \max \{2 \cdot \omega_1, 10 \cdot \omega_2\}. \quad (18)$$

Finally, $\omega_3$ is combined with $\omega_4$:

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

where:

$$\omega_4 = \max \{P(1) - P(3), P(2) - P(1), P(1) - P(2), 1.1 \cdot [P(2) - P(4)], 1.1 \cdot [P(2) - P(4)]\}$$

(20)

The final result $\omega$ is quantized using 15 thresholds: $T_\omega = \{3, 8, 14, 20, 27, 34, 43, 55, 66, 80, 100, 120, 150, 180, 240\}$.

C. Prediction error sign coding

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 estimate error signs for left and upper coded pixel neighbors (indexed 1 and 2 in Fig.1). The remaining two bits are obtained from quantization of the main context (20), here quantization thresholds are: $\{8, 20, 180\}$.

IV. EXPERIMENTS

We started with some “method tuning” experiments, their results are reported in Figures 3 and 4, averaged data for 45 test images are shown. These are widely used 8-bit gray-scale images of size 720x576 pixels (9 images, including Boats, Zelda, etc.), 512x512 pixels (26 images, e.g. Baboon, Lenna), and 256x256 pixels (10 images, e.g. Camera, Omaha). In Figure 3 the influence of training window size implemented in the TCM subpredictor on bit rate value for the whole Blend-17 method is visualized, $W = 0$ means that TCM predictor is switched off.
Fig. 4 shows results obtained for an even more crude proposed method simplification consisting in choosing only $n$ the best predictors, $n = 1, 2, ..., 17$, $W = 3$, if applicable. The trade-off between algorithm performance and computational complexity ($\sim$window size, or predictors number) is clearly seen, as well as manifestation of the rule of diminishing returns.

In Fig. 5 computational complexity of TCM method as a function of training window size $W$ is demonstrated. Its time complexity is $O(nW^2)$, which is clearly seen from the figure, nevertheless, due to careful algorithm implementation for small $W$ the growth is nearly linear. Timing is shown for 512x512-pixel Lenna grey image processed on 2.8 GHz Pentium 4 (for other images the curve has the same shape, coding times depend only on image size).

Results of the main experiment can be found in Table 2, in which results for several relatively fast state-of-art methods are compared to Blend-17, and its version without TCM subpredictor, denoted Blend-16, for reference a more complex GLICBAWLS [20] technique has been added. Among the compared algorithms the predictor blending ones are: HBB [14], Lee [17], P13 [15], APC-A [19], and Blend-13 [10]. As can be seen, both new methods reveal outstanding performance.

Coding time of Lenna grey image by highly optimized Blend-13 version was 0.48 sec, for Blend-16 and Blend-17 times were 4.48 and 6.75 sec, respectively, which was less than 1/3 of the coding time for GLICBAWLS. The timing difference disappeared for large training window applied in TCM based prediction, for $W = 15$ the time was 24.25 sec. Coding times for other 512x512 pixel images were practically the same, for larger images the differences in coding times increased. Note that JPEG-LS algorithm is simpler and much faster than other compared techniques, hence, it is included for reference, only.

V. CONCLUSION

A new, relatively fast and efficient predictor blending method for image lossless coding has been described in the paper. The algorithm is blending carefully chosen 17 subpredictors, hence its name Blend-17. The improved performance of the technique is mainly due to individually optimized parameters of its subpredictors ($a_k$ and $k_k$ in Table 1), and due to careful treatment of predictor error bias (a sophisticated procedure applied to selected subpredictor errors, only). An advanced context adaptive arithmetic coder is worth noting, too. Of some interest are presented in the paper performance analyses of predictor blending approach for increasing number of subpredictors, and for influence of training windows size applied in texture context matching based subpredictor. Experimental results show that indeed, the improvements resulted in exceptionally high efficiency of the new algorithm.

ACKNOWLEDGMENT

The work has been partially sponsored by Polish Ministry of Science and Higher Education grant “Algorytmy bezstronnej i prawie-bezstronnej kompresji danych multimedialnych dedykowane architekturze Networks on Chip”.

REFERENCES

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Balloon</td>
<td>2.889</td>
<td>2.80</td>
<td>2.78</td>
<td>2.79</td>
<td>2.74</td>
<td>2.75</td>
<td>2.73</td>
<td>2.746</td>
<td>2.670</td>
<td>2.671</td>
</tr>
<tr>
<td>Board</td>
<td>3.674</td>
<td>3.54</td>
<td>3.51</td>
<td>3.50</td>
<td>3.48</td>
<td>3.48</td>
<td>3.43</td>
<td>3.465</td>
<td>3.375</td>
<td>3.364</td>
</tr>
<tr>
<td>Boats</td>
<td>3.930</td>
<td>3.80</td>
<td>3.78</td>
<td>3.76</td>
<td>3.75</td>
<td>3.74</td>
<td>3.70</td>
<td>3.716</td>
<td>3.638</td>
<td>3.633</td>
</tr>
</tbody>
</table>