Lossless Image Compression Using Adaptive Predictor Combination, Symbol Mapping and Context Filtering*

Guang Dong and Hua Ye
Department of Electronic Engineering
La Trobe University
Bundoora, VIC 3083, Australia

Abstract

Common components in recently published lossless image compression algorithms include adaptive prediction, context-based error feedback and adaptive entropy coding. Each component has a number of building blocks. In this paper, we present a new algorithm which uses three new building blocks: an adaptive predictor, a symbol mapping scheme and a context filtering scheme. Experimental results show that the compression performance of the proposed algorithm is better than those of the three state-of-the-art algorithms: CALIC, HBB and LOCO. Experimental results also show that the proposed new building blocks are promising tools for lossless image compression.

1. Introduction

Lossless image compression techniques can be classified as a one-pass or a two-pass process. Using a one-pass technique, all of the parameters required in coding are estimated on the fly of the coding process. It is not necessary to transmit these parameters as side information. Using a two-pass technique, parameters are estimated in a training pass for a particular image or a set of images. These parameters are then used in the subsequent coding process and must be transmitted to the decoder as side information. Since a two-pass technique can gather more information on the image to be compressed, it generally has better compression performance. However, it may require much more computation in achieving this.

The state-of-the-art one-pass sequential techniques, including CALIC [13], HBB [10] and LOCO [8], have three major components: adaptive prediction, error feedback and adaptive entropy coding. A recently published two-pass sequential technique is the TMW [6] [7] algorithm.

Although a lot of efforts have been put into lossless image compression research and the new JPEG-LS standard has significant improvements over the lossless JPEG, it remains a challenge for researchers to find out the ultimate image compressibility regardless the computational complexity and to develop new ideas and algorithms for practical applications.

In this paper, we develop a new predictor by adaptively combining a set of simple and fixed predictors. When coupled with symbol mapping, context filtering, error feed back and adaptive arithmetic coding, the compression performance of the proposed algorithm is better than those of CALIC, HBB and LOCO.

In the next section, we give a brief review of several adaptive prediction schemes. We then present our new scheme for linear combination of predictors and the whole compression algorithm using this new prediction scheme. We next apply this algorithm on the JPEG 8-bit test image set. Finally, we present our conclusions in Section 6.

2. A brief review of several adaptive prediction schemes

In this section, we briefly describe three kinds of adaptive predictors. For additional discussion on recent developments in lossless image compression techniques, the reader is referred to [4] and [5]. Adaptation can be achieved in a number of ways. One way is to use a fixed predictor structure such as a linear combination of the causal neighboring pixel values and to adaptively change the coefficients for each pixel. The coefficients can be calculated based on the local gradients [9] [5]. They can also be calculated by using the least mean square error based algorithm [2].

Another way is to have a set of simple and fixed predictors and adaptively select one predictor as the final predictor. The median edge detection (MED)
3. A new scheme for linear combination of predictors

Let $p_i (i = 1, 2, \ldots, N; p_i \neq p_j \text{ when } i \neq j)$ represent the causal predictors for the current pixel denoted $x$. They can be linear or nonlinear. The proposed predictor denoted $P$ is a linear combination of the above predictors:

$$P = \sum_{k=1}^{N} \alpha_k p_k$$

(1)

where $\alpha_k$ are the combination coefficients and $\sum_{k=1}^{N} \alpha_k = 1$. The optimum coefficients result in the least mean square prediction error. Thus we can define a target function as:

$$F = E(e^2) + \beta \left( 1 - \sum_{k=1}^{N} \alpha_k \right)$$

(2)

where $\beta$ is a constant to be determined. The optimal coefficients $\alpha_k$ are obtained by minimizing this function $F$.

Let $C$ be an $N \times N$ matrix whose elements are $\sigma_{ij} = E(e_i e_j)$, and $e_i = x - p_i$. $\sigma_{kk}$ is the mean square error for the $k$th predictor. Further, let $A$ and $B$ be two column vectors, $A = [\alpha_1, \ldots, \alpha_N]^t$ and $B = \left[ \frac{\sigma_{11}}{2}, \frac{\sigma_{12}}{2}, \ldots, \frac{\sigma_{N1}}{2} \right]^t$, where $t$ denotes the matrix transpose. Then the optimal coefficients are given by

$$A = C^{-1} B.$$ 

(3)

When using the above predictor in real applications, matrix $C$ must be known and invertible. It is also assumed that the signal is stationary. Due to these problems, this predictor is not practical.

To deal with these problems, we propose an algorithm which is based on a simplification of matrix $C$, where it is assumed that $\sigma_{ij} = 0 \text{ for } i \neq j$. In this case, it can be shown that the predictor coefficients are given by:

$$\alpha_k = \frac{1}{D \sigma_{kk}}$$

(4)

where

$$D = \sum_{k=1}^{N} \frac{1}{\sigma_{kk}}$$

is a normalization factor. It is also easy to show that the mean square error of the proposed predictor is:

$$e_{\text{min}} = \frac{1}{D}.$$ 

(5)

If we regard each predictor as a resistor and its associated mean square error as its resistance, then the
The mean square error of the proposed predictor can be regarded as the total resistance of all resistors in parallel. It is well known that for resistors in parallel, the total resistance is smaller than any individual resistor and that the more resistors are put together, the less the total resistance. Thus, the proposed predictor is always better than each individual predictor in terms of mean square error, i.e., $\text{ermin} < \sigma_{kk}, k = 1, 2, \ldots, N$, and the mean square error $C_{\text{min}}$ will be smaller as more predictors are used.

A robust and adaptive method to estimate $\sigma_{kk}$ for each pixel to be predicted is the key to the success of this predictor. We propose a simple method to estimate $\sigma_{kk}$. Suppose the current pixel to be predicted $x$ is the $n$th pixel in the image. If we denote the mean square error for the $k$th predictor for this pixel as $\sigma_k(n)$, we can calculate these mean square errors from

$$\sigma_k(n) = \frac{1}{2} [\sigma_k(n-1) + E_k(n)]$$

(6)

where $E_k(n)$ is an estimate of the “local” mean square error for the $k$th predictor. Let $e_k(i, j)$ indicate the prediction error of the $k$th predictor for the pixel $s(i, j)$. $E_k(n)$ is given by

$$E_k(n) = e_k^2(i, j - 1) + e_k^2(i - 1, j - 1) + e_k^2(i - 1, j) + e_k^2(i - 1, j + 1)$$

Equation (6) is a simple recursive filter that is employed to smooth out noise in the estimate and to make the estimate more robust.

A set of simple and fixed linear predictors are employed in our experiments. They are listed below:

1. $p_1 = s(i, j - 1)$
2. $p_2 = s(i, j - 1) + s(i - 1, j) - s(i - 1, j - 1)$
3. $p_3 = s(i - 1, j)$
4. $p_4 = s(i - 1, j + 1)$
5. $p_5 = \frac{1}{2} [s(i - 1, j) + s(i, j - 1)]$
6. $p_6 = s(i - 1, j - 1)$
7. $p_7 = s(i, j - 1) + \frac{1}{2} [s(i - 1, j) - s(i - 1, j - 1)]$
8. $p_8 = s(i - 1, j) + \frac{1}{2} [s(i - 1, j - 1) - s(i, j - 1)]$
9. $p_9 = \frac{1}{4} [2s(i, j - 1) + s(i - 1, j) + s(i - 1, j + 1)]$
10. $p_{10} = \frac{1}{2} [s(i - 1, j) + s(i - 1, j + 1)]$
11. $p_{11} = s(i, j - 1) + s(i - 1, j - 1) - s(i - 1, j)$
12. $p_{12} = s(i, j - 1) + s(i - 1, j + 1) - \frac{1}{2} [s(i - 1, j) + s(i - 1, j - 1)]$
13. $p_{13} = s(i, j - 1) + s(i - 1, j) - \frac{1}{2} [s(i - 1, j - 2) + s(i - 1, j - 1)]$

This set of predictors contains the whole set of predictors recommended by the JPEG standard. It also contains other simple predictors which are similar to their respective JPEG recommended predictors. For example, predictors 2, 11-13 are similar.

Our experiments show that although using more predictors generally results in better compression, using only the first 6 predictors has produced very good results. There is a trade-off between the compression performance and the computational complexity. This will be discussed in Section 5.

The above proposed adaptive predictor presented in Section 2 is an extension to the work by Seemann and Tischer [10]. In their work, the predictor design is formulated in an intuitive way where a “bad” predictor should be “penalized”. The contribution of our work is to address the problem in a least mean square error framework and to use the variance of the prediction error to measure the “badness” of a predictor and a robust way to “penalize” it.

4. The Proposed Algorithm

In [1], we have presented a lossless image compression algorithm which has the following building blocks: (1) a predictor, (2) error representation by symbol mapping, (3) a context filtering scheme, (4) context based error feedback and (5) adaptive arithmetic coding. The proposed algorithm in this paper uses all these building blocks, except that the proposed predictor is used. The proposed algorithm can be summarized into the following steps:

1. Calculate the adaptive prediction $P$ by using algorithm described in the last section.
2. Calculate the context for the current pixel.
3. Calculate the final prediction error.
4. Update all parameters required in encoding.
5. Map the prediction error into three parts and encode them.
6. Move to the next pixel.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>balloyn.clr</td>
<td>2.83</td>
<td>2.90</td>
<td>2.80</td>
<td>2.66</td>
<td>2.84</td>
<td>2.76</td>
<td>2.74</td>
</tr>
<tr>
<td>barby2y.clr</td>
<td>4.53</td>
<td>4.69</td>
<td>4.48</td>
<td>4.38</td>
<td>4.59</td>
<td>4.49</td>
<td>4.47</td>
</tr>
<tr>
<td>barby.clr</td>
<td>4.41</td>
<td>4.69</td>
<td>4.28</td>
<td>4.08</td>
<td>4.51</td>
<td>4.31</td>
<td>4.29</td>
</tr>
<tr>
<td>board.y</td>
<td>3.56</td>
<td>3.68</td>
<td>3.54</td>
<td></td>
<td>3.64</td>
<td>3.50</td>
<td>3.48</td>
</tr>
<tr>
<td>boats.y</td>
<td>3.84</td>
<td>3.93</td>
<td>3.80</td>
<td>3.61</td>
<td>3.90</td>
<td>3.76</td>
<td>3.75</td>
</tr>
<tr>
<td>girly.clr</td>
<td>3.77</td>
<td>3.93</td>
<td>3.47</td>
<td></td>
<td>3.85</td>
<td>3.69</td>
<td>3.67</td>
</tr>
<tr>
<td>goldy.clr</td>
<td>4.39</td>
<td>4.48</td>
<td>4.37</td>
<td>4.27</td>
<td>4.41</td>
<td>4.34</td>
<td>4.33</td>
</tr>
<tr>
<td>hostely.clr</td>
<td>4.25</td>
<td>4.38</td>
<td>4.27</td>
<td></td>
<td>4.32</td>
<td>4.20</td>
<td>4.19</td>
</tr>
<tr>
<td>zedday.clr</td>
<td>3.75</td>
<td>3.89</td>
<td>3.72</td>
<td></td>
<td>3.78</td>
<td>3.70</td>
<td>3.68</td>
</tr>
<tr>
<td><strong>Average bit-rate</strong></td>
<td><strong>3.99</strong></td>
<td><strong>4.06</strong></td>
<td><strong>3.89</strong></td>
<td></td>
<td><strong>3.98</strong></td>
<td><strong>3.80</strong></td>
<td><strong>3.84</strong></td>
</tr>
</tbody>
</table>

Table 1: Compression results (bits/pixel). The test images are 720x576 8 bit/pel.

5. Experimental Results

The set of test images are the JPEG 8-bit test images. The proposed algorithm with all building blocks described above is used to compress the test set. We use P-6 and P-13 to represent the cases where the proposed adaptive predictor uses the first 6 predictors and all 13 predictors, respectively. We also use P-MED to represent that in our algorithm the adaptive predictor is replaced by the MED predictor used in LOCO.

For comparison, CALIC and LOCO are also applied to all test images and where applicable compression results by using the HBB and TMW algorithms are also quoted. The results are shown in Table 1. It can be seen that the average bit rate of the proposed algorithm is better than that of CALIC, LOCO and HBB but is worse than that of TMW.

When developing a lossless image compression algorithm, one often has to pursue a trade-off between the compression performance and the computational complexity. In this sense, LOCO is the best algorithm.

On the other hand, one can pursue the maximal compression regardless of the computational complexity. In this sense, to the best of our knowledge, TMW is better than any other published algorithms. However, TMW is also computationally much more complicated than other algorithms. In our experiments, it takes 144 seconds to encode a 512 x 512 image on a PC with a Pentium Pro (200MHz) processor running the Linux operating system. It should be noted that the time is only for the encoding pass of TMW. The running time for the training pass is not known to us.

Using the same computer and for images of the same size, our proposed algorithm takes 3.8 seconds to finish the encoding (using 6 predictors). If 13 predictors are used, then the encoding time is 5.8 seconds. The time a particular algorithm takes to encode a 512 x 512 image is listed in Table 2. It should be noted that the time listed in this table is only a rough estimate of the computational complexity of an algorithm. The actual coding time depends on the implementation of the algorithm. A carefully written and refined implementation that eliminates unnecessary operations obviously requires less time to run. In this sense, it is expected that the coding time for our algorithm and the TMW algorithm can be reduced.

6. Conclusions

In this paper, we have proposed a new algorithm for lossless image compression. In particular, we have presented an investigation of optimal predictor design using a linear combination of simple predictors. It has also been shown that the proposed locally adaptive combination of simple predictors is a useful building block for lossless image compression. Experimental results have shown that the performance of the proposed algorithm is better than those of CALIC, HBB and LOCO.

References


<table>
<thead>
<tr>
<th>Algorithm</th>
<th>CALIC</th>
<th>LOCO</th>
<th>P-MED</th>
<th>P-6</th>
<th>P-13</th>
<th>TMW</th>
<th>P-MED</th>
<th>P-6</th>
<th>P-13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>2.18</td>
<td>0.32</td>
<td>5.03</td>
<td>9.09</td>
<td>13.5</td>
<td>144</td>
<td>2.46</td>
<td>3.8</td>
<td>5.8</td>
</tr>
</tbody>
</table>

Table 2: Running time (seconds) for various algorithms.