

# Transform domain LMS-based adaptive prediction for lossless image coding

GUANG DENG

Department of Electronic Engineering, La Trobe University,  
Bundoora, Victoria 3083, Australia.  
Email: d.deng@ee.latrobe.edu.au  
Phone: +61 3 9479 2036 Fax: +61 3 9471 0524.

Submitted to Signal Processing Image Communication, June,  
2000. Revised June 12, 2001, July 12, 2001 (fix some  
typos)

## **Abstract**

This paper is concerned with adaptive prediction for lossless image coding. A new predictor is proposed. This predictor involves two major steps: constructing a good predictor for each pixel using the transform domain LMS algorithm and adaptively combining it with a set of fixed predictors. The first step is targeting areas where simple predictors do not perform well, while the second step is an effective method to reduce the modelling costs associated with the uncertainty of the models. When a context-based arithmetic encoder is used to encode the prediction error, the compression performance of the proposed algorithm is better than or comparable to that of other published algorithms.

## KEYWORDS

Lossless image compression, adaptive prediction, transform domain LMS algorithm, context-based entropy coding

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>The adaptive predictor</b>	<b>8</b>
2.1	The TDLMS predictor . . . . .	8
2.1.1	Basic algorithm . . . . .	8
2.1.2	Training methods and computational complexity . . . . .	10
2.1.3	Adaptive training . . . . .	12
2.2	Adaptive predictor combination . . . . .	13
2.3	Discussion . . . . .	14
<b>3</b>	<b>Error representation and adaptive entropy coding</b>	<b>16</b>
3.1	Error representation . . . . .	16
3.2	Context-based adaptive arithmetic coding . . . . .	17
3.3	The coding and decoding process . . . . .	18
<b>4</b>	<b>Experimental results</b>	<b>19</b>
<b>5</b>	<b>Conclusion</b>	<b>23</b>

# List of Figures

1	An illustration of the causal neighbouring pixels used in the transform domain LMS-based predictor. $x(n)$ is the current pixel. . . . .	9
2	The training block. $x(n)$ is the current pixel to be predicted. . . . .	11

## List of Tables

1	List of fixed predictors. See Fig.1 for the locations of the causal neighbouring pixels. . . . .	13
2	Compression results (bits/pixel) and the average bit rate (ABR) for the JPEG test image set. Default settings for JPEG-LS are used. . . . .	20
3	Compression results (bits/pixel) and the average bit rate (ABR) for the second set of images. Default settings for CALIC, JPEG-LS and HBB are used. . . . .	21
4	Comparison with JPEG-LS and TMW using the TMW test image set. The time that the algorithm takes to compress the whole set of images is presented. *This shows only the encoding time of the TMW algorithm, the time used for image analysis is not known. . . . .	21
5	Relative performance and complexity for different algorithms. RP represents relative performance measured by dividing the average bit rate for an algorithm by that of JPEG-LS. RC represents the relative complexity measured by dividing the running time required by an algorithm to encode the whole set of images by that required by JPEG-LS. *This shows only the encoding time of the TMW algorithm, the time used for image analysis is not known. . . . .	23

# 1 Introduction

Lossless image compression techniques usually have two major components: adaptive prediction and adaptive entropy coding. These techniques first predict the gray level of a pixel using an adaptive predictor, then encode the prediction error using a context-based entropy coder. This paper is concerned with adaptive prediction. For additional discussion on lossless image compression techniques, the reader is referred to survey papers by Memon *et. al.* [9], [10].

The JPEG-LS standard, based on a low complexity, context-based, lossless image compression algorithm (LOCO-I) [14], has achieved the best compromise between compression performance and computational complexity. The JPEG 2000 coding system [1] also provides functionality for progressive lossless image compression. It has been reported that while JPEG 2000 performs almost the same as JPEG-LS for natural images, the latter encodes an image significantly faster than the former. In a recent paper [11], the performance of JPEG-LS is compared to those of other standards such as JPEG 2000, lossless JPEG and PNG (portable network graphics). It is shown that JPEG-LS performs better than these standards. Thus, the JPEG-LS standard is a reasonable choice for lossless image compression applications.

Another well known algorithm, the CALIC (context-based, adaptive, lossless codec) algorithm [15], uses adaptive prediction and adaptive entropy coding. CALIC has been regarded as the benchmark technique in terms of compression ratio for the past few years. Its computational complexity is also relatively low. The predictors used in JPEG-LS and CALIC are simple. Their performance is thus limited.

Current lossless image coding research, which follows the basic idea of prediction and entropy coding, may be divided into two categories. The aim of the first category is to find out the ultimate compression that can be achieved regardless of the computational complexity. Since the publication of the TMW algorithm (version 0.51) [8], it is clear that if the

computational complexity is not considered, then further improvement in compression can be achieved by using better adaptive prediction and entropy coding schemes. TMW is basically a two pass algorithm. In the first pass, parameters for adaptive prediction and entropy coding are determined by an image analysis process. These parameters are then used in the second pass to encode the image. The TMW algorithm has produced the best compression results for a set of test images commonly used in testing lossless image coding algorithms. A number of researchers [7] [16] [17] have recently attempted to use adaptive predictors based on the Least Squares (LS) approach to achieve roughly the same compression performance as that of the TMW while keeping the computational complexity lower than TMW.

The aim of the second category is to achieve better compression performance than CALIC and JPEG-LS while keeping computational complexity reasonably low. Algorithms that outperform CALIC have been proposed by a number of researchers. In these algorithms, effective adaptive prediction schemes are used. These include: designing a linear predictor for each pixel based on the LS approach [17] and using the Boolean and stack filters [13]. The HBB (history-based blending) [12] algorithm uses an adaptive predictor which is a linear combination of a set of simple predictors. The combination coefficients are determined by the LS approach. Lee [6] proposed a predictor by combining an adaptive predictor with four causal neighbouring pixels based on the Bayesian principle. The coefficients of the adaptive predictor are updated by using the LMS algorithm on a pixel-by-pixel basis. The outstanding compression performance of these algorithms is partly due to using adaptive predictors that are based on well developed principles. This is in contrast to the predictor used in CALIC which is heuristically designed. However, it should be noted that the improvement in performance is at the cost of increased computational complexity.

The study presented in this paper belongs to both categories in the sense that the goal of this paper is to develop an adaptive prediction scheme that can be configured either for a trade off between performance and complexity or for optimal performance. The motivation

for this study is to explore alternative methods for adaptive prediction.

This study is an extension of our recent work on adaptive predictor combination (APC) [2] by which a group of predictors are adaptively combined to form the prediction. We have tested the algorithm which is an adaptive combination of a group of fixed predictors. Each predictor uses only some or all of four causal neighbouring pixels of the pixel to be predicted. The combination coefficients for the predictors are determined by a robust estimate of the mean square prediction errors of each predictor. We have shown that, when coupled with a context-based adaptive arithmetic coder, the performance of this algorithm is better than CALIC, HBB and JPEG-LS. This algorithm is also faster than HBB.

The predictors, which only use four causal pixels, are simple. Their ability to model complex data such as areas with rich textures or edges is limited. Thus, a natural way to improve the performance of this algorithm is to incorporate more causal neighbouring pixels in the predictors and to make them adaptive to the local characteristics of the image. The transform domain LMS (TDLMS)-based adaptive filter is a useful tool for this purpose, because it has a faster convergence rate than the normal LMS algorithm [5].

In this paper, we proposed an adaptive predictor which is a combination of a set of six fixed predictors with a TDLMS-based predictor. A distinctive property of the proposed predictor is that the TDLMS algorithm is used to design a good predictor for each pixel by a training process. Fixed and adaptive block training methods for the TDLMS algorithm are proposed. This is different from the traditional way of using the LMS algorithm to minimize the mean square error. Another distinctive property of the proposed predictor is that the adaptive combination scheme is an effective method to reduce the prediction error due to the modelling uncertainty associated with each predictor. This is because models are usually based on assumptions. The modelling uncertainty is related to the fact that assumptions may not be valid everywhere in real signals.

Section 2 presents the adaptive predictor and an adaptive block training method for the

TDLMS predictor. Section 3 presents the entropy coding method. Experimental results and a comparison with other published algorithms are presented in Section 4, which shows that the compression performance of the proposed algorithm is consistently better than those of CALIC, HBB, JPEG-LS and Lee’s algorithm. We have also presented experimental results that use JPEG-LS as a reference to compare the performance and complexity of a number of algorithms.

## 2 The adaptive predictor

This section presents the proposed adaptive predictor. The TDLMS algorithm, the block-based training method, and the associated computational costs are described. This is followed by the adaptive predictor combination scheme.

### 2.1 The TDLMS predictor

#### 2.1.1 Basic algorithm

To simplify notation, we map the image signal into a one dimensional vector through a scanning order illustrated in Figure 1. The number of causal neighbouring pixels that are used to form the predictor is called the order of the predictor. An  $N$ th order predictor is used in the following discussion. Let the current pixel to be predicted, denoted  $x(n)$ , be the  $n$ th pixel in the image and the vector  $X_n = [x(n-1), x(n-2), \dots, x(n-N)]^t$  be a column vector whose elements are the causal neighbouring pixels of  $x(n)$ . The unitary transformation of the signal vector  $X_n$  is also a column vector and can be expressed as

$$Y_n = TX_n \tag{1}$$

		x(n-15)	x(n-10)	x(n-16)	
	x(n-13)	x(n-7)	x(n-6)	x(n-8)	x(n-14)
	x(n-11)	x(n-3) NW	x(n-2) N	x(n-4) NE	x(n-12)
x(n-9)	x(n-5)	x(n-1) W	x(n)		

Figure 1: An illustration of the causal neighbouring pixels used in the transform domain LMS-based predictor.  $x(n)$  is the current pixel.

where  $T$  is a  $(N \times N)$  transform matrix. Further let  $y_n(k)$  be the  $k$ th element of the vector  $Y_n$ .

The prediction of the current pixel  $x(n)$  is given by

$$p_0(n) = A_n Y_n \quad (2)$$

where  $A_n = [a_n(1), a_n(2), \dots, a_n(N)]$  is a row vector whose elements are the prediction coefficients for pixel  $x(n)$ . In actual implementation, if the prediction value is out of the legal range (for example,  $[0, 255]$  for 8 bits/pixel images), then the prediction coefficients are reset to its default values and a default predictor will be used.

The transform domain LMS algorithm involves the following steps [5]:

1. Calculate the prediction error:

$$e(n-1) = x(n-1) - A_{n-1} Y_{n-1} \quad (3)$$

2. Update the scaling factor:

$$\lambda_{n-1}(k) = \gamma \lambda_{n-2}(k) + (1 - \gamma) y_{n-1}^2(k) \quad (4)$$

3. Update the predictor coefficients:

$$a_n(k) = a_{n-1}(k) + \frac{\beta}{\lambda_{n-1}(k)} y_{n-1}(k) e(n-1) \quad (5)$$

In the above steps,  $0 < \beta < 1$  and  $0 < \gamma < 1$  are two scaling factors. In step 2, the scaling factor  $\lambda_{n-1}(k)$  is actually a recursive estimate (at the index  $n-1$ ) of the  $k$ th diagonal elements of the covariance matrix of the transform coefficients. In all of our experiments, we set  $\gamma = 0.95$  and  $\beta = 0.02$ . Experimental results show that the performance of the predictor is not very sensitive to small variation in these two parameters. The two vectors  $A_n$  and  $\lambda_n$  are initialized as  $[1, 0, \dots, 0]$  and  $[0, 0, \dots, 0]$ , respectively.

The computation of the unitary transformation of the signal vector is a major factor in the complexity of the algorithm. Fast algorithms exist for the discrete cosine transform (DCT) and the Walsh-Hadamard transform (WHT), which are widely used in signal processing applications. The latter only requires  $N \log_2 N$  addition operations and hence is faster than the former for a general purpose processor. We have tested these two transformations in our compression algorithms. Experimental results showed that the use of the DCT has no advantage in terms of compression performance over the use of the WHT. Therefore, the WHT is used in our algorithm. We have also carried out experiments using the 8th and 16th order TDLMS predictor. Experimental results showed that the performance of the latter is only marginally better than that of the former at a significant increase of computational costs. Thus, in the experimental results presented in Section 4, an 8th order TDLMS predictor is used.

### 2.1.2 Training methods and computational complexity

There are basically two methods to obtain the prediction coefficients through a training process. The aim is to train the prediction coefficients such that the resulting predictor is a good predictor for the current pixel. Using a pixel-based training method, the predictor coeffi-

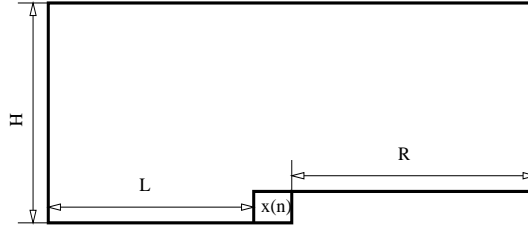


Figure 2: The training block.  $x(n)$  is the current pixel to be predicted.

coefficients are initially set to some default values and are updated using the current prediction error. This is shown in the above three steps. The coefficients are then used to form the prediction for the next pixel.

Using a block-based training method, the prediction coefficients for the current pixel are obtained by running the TDLMS algorithm through a block of previously encoded pixels shown in Figure 2. It is expected that after training the set of coefficients is very close to the set of optimum coefficients that results in the minimum mean square error for the training block. It is assumed that if the predictor is optimum or near optimum for the training block, then it will also be a good predictor for the current pixel due to the correlations between the current pixel and its neighbouring pixels. This assumption is valid if small segments of the signal can be considered stationary. The size of the training block can be described by three numbers:  $H$ ,  $L$  and  $R$ . Denoting the number of pixels in the block  $M$  ( $M = H(L + 1) + R(H - 1)$ ), it can be easily seen that the number of operations required for training process are  $(6N + 1)M$  multiplications,  $(2N + 1)M$  additions and  $MN \log_2 N$  addition operations required by the WHT. A small training block requires significantly less operations than that required for a large training block. However, our experimental results show that a larger training block generally results in better compression performance. Therefore, there is a trade-off between the computational complexity and the compression performance.

### 2.1.3 Adaptive training

One way to solve the complexity problem is to adaptively change the size of the training block. Since the basic idea of using the TDLMS predictor is to achieve good prediction in areas of the image where simple predictors do not perform well, a simple strategy is to use the TDLMS predictor only in those “difficult” areas rather than in the whole image. For those areas where the TDLMS predictor is required, the training block size can also be adaptively adjusted according to the local gradients and prediction errors. An adaptive training algorithm is described below.

When the current pixel to be predicted is located in a smooth area, then the TDLMS based predictor is not used. If the sum of the absolute values of the local gradients ( $G$ ) are less than a threshold, then the current pixel is regarded as being in a smooth area. In the proposed algorithm,  $G$  is defined as:

$$G = |x(n-5) - x(n-1)| + |x(n-1) - x(n-3)| + |x(n-3) - x(n-2)| + |x(n-2) - x(n-4)|$$

If  $G$  is greater than a threshold, then the TDLMS predictor is used and the size of its training block is determined by the context ( $C(n)$ ) used in adaptive arithmetic coding (described in Section 3.2). The context is a combination of the past prediction errors with the local gradient values. It can be regarded as a representation of the local characteristics of the image. To simplify the adaptation process, we assume that  $H = L = R$ . The block size adaptation rule described below is determined by experiments:

if ( $C(n) > 16$ ) then  $H = 5$   
else if ( $C(n) > 12$ ) then  $H = 4$   
else if ( $C(n) > 8$ ) then  $H = 3$   
else if ( $C(n) > 4$ ) then  $H = 2$   
else if ( $C(n) > 2$ ) then  $H = 1$

$p_0 =$ TDLMS predictor $p_1 = W$ $p_2 = N$ $p_3 = N + W - NW$ $p_4 = NE$ $p_5 = (N + W)/2$ $p_6 = NW$
--

Table 1: List of fixed predictors. See Fig.1 for the locations of the causal neighbouring pixels.

## 2.2 Adaptive predictor combination

In the proposed algorithm, the TDLMS predictor is adaptively combined with a set of six fixed predictors shown in Table 1 to form the final prediction. Let  $p_k(n)$  ( $k = 0, 1, 2, \dots, 6$ ) represent the  $k$ th causal predictor for the current pixel. The proposed predictor is a linear combination of these predictors:

$$P(n) = \frac{1}{D} \sum_{k=0}^6 \alpha_k p_k(n) \quad (6)$$

where  $\alpha_k$  ( $\alpha_k > 0$ ) are the coefficients and  $D = \sum_{k=0}^6 \alpha_k$  is a normalization factor. The prediction is truncated to its nearest integer in actual implementation.

Intuitively, the coefficient for a predictor could be determined by its predictive performance. A larger coefficient should be assigned to a better performing predictor. Since the mean square prediction error (MSE) is an important indicator of the predictor performance, it is used to calculate the coefficient. As natural images are non-stationary signals, a predictor could perform very well in one area and could fail in other areas. Thus, it is necessary for the MSE to track the changing performance of the predictor. To achieve this goal, the MSE must emphasize the local prediction errors. An estimate for the local mean square error for the  $k$ th predictor is denoted:  $\sigma_k(n)$ . The proposed adaptive predictor coefficient is calculated by using:

$$\alpha_k = \frac{1}{1 + \sigma_k(n)} \quad (7)$$

A simple method to estimate  $\sigma_k(n)$  is given by the following equation:

$$\sigma_k(n) = \frac{1}{2} [\sigma_k(n-1) + E_k(n)] \quad (8)$$

where  $E_k(n)$  is an estimate of the local mean square error for the  $k$ th predictor.  $E_k(n)$  is given by:

$$E_k(n) = e_k^2(n-1) + e_k^2(n-2) + e_k^2(n-3) + e_k^2(n-4) \quad (9)$$

where  $e_k(n-m) = x(n-m) - p_k(n-m)$  ( $m = 1, 2, 3, 4$ ) is the prediction error of the  $k$ th predictor at location  $n-m$ . Equation (8) is a simple recursive filter that is employed to smooth out noise in the estimate. The initial value of the estimate, denoted  $\sigma_k(-1)$ , is set to zero.

A context based error feedback scheme, which is similar to that used in JPEG-LS, is applied to the prediction to form the final prediction. The prediction error is then encoded by a context-based entropy coder. This will be described in Section 3.

## 2.3 Discussion

It can be seen that the proposed adaptive predictor combination scheme involves two major steps: designing a good predictor for each pixel and combining it with a group of fixed predictors to reduce the average modelling cost. In this section, we briefly discuss issues such as: the TDLMS predictor, the adaptive combination scheme and its limitations, the difference between the proposed algorithm and Lee's algorithm.

The block based coefficient training represents an alternative way for adaptive prediction filter design. Its goal is to design a good predictor for each pixel to be predicted. Such a

method is based on the assumption that the statistical property of the signal in a small area is close to stationary. Thus, if the prediction filter is obtained by minimizing the MSE over a block of causal pixels, then it is expected that this filter will perform well for the current pixel. It is noted that the predictor designed in this way does not assume any optimal property.

The prediction filter can be obtained by using the LS approach by which the mean square error (MSE) for the training block is minimized. However, the LS approach is computationally intensive. While the TDLMS algorithm is simpler than the LS approach, the predictor it produces is generally inferior to the predictor determined by using the LS approach. Thus, using the TDLMS algorithm to obtain the prediction filter is a trade off between complexity and performance.

In general, a predictor, including the TDLMS predictor, represents a model of the image. Each model is based on certain assumptions. For a given pixel and its neighbouring pixels, if the assumption for a model (predictor) is true, then it is expected that the model will make an accurate prediction. Otherwise, the prediction may be far from optimum. The cost of a model can be defined as a function of the modelling (prediction) errors. Since there is almost always modelling uncertainty whether or not the assumption holds, using only one of the predictors is a risky process that simply ignores the modelling costs.

The combination of a group of predictors is an effective way to reduce the modelling costs. The proposed adaptive combination scheme presented in Section 2.2 follows essentially the Bayesian principle. The combination coefficient for a model is inversely proportional to the localized estimate of the modelling cost. Models with higher costs are given less weights. Therefore, it is expected that the average modelling cost of the proposed method is less than that of using only one model.

We note that our focus in this paper is on the compression of natural images which include natural scene, medical and radar images. Other types of images, such as binary

images and computer generated graphics, have their distinct probability distributions which are usually different from those of natural images. While the proposed adaptive predictor is best suited for the former type of images, it is not optimized for the latter. Therefore, we only use natural images to test the proposed algorithm in Section 4 of this paper. We also note that the proposed adaptive predictor can be easily extended to cope with the latter type of images by adopting a similar strategy as that used in LOCO-I (the run-mode).

The proposed adaptive predictor combination algorithm differs from Lee's algorithm in two aspects. Firstly, Lee's algorithm uses a time domain LMS based predictor whose coefficients are updated using the prediction error of the previous pixel, the proposed algorithm uses a transform domain LMS predictor whose coefficients are determined by an adaptive block training method. Secondly, in Lee's algorithm the combination coefficients are determined by modelling the posterior probability density function of the prediction errors for each predictor. In the proposed method, the combination scheme, although it follows the same Bayesian principle, is a non-parametric method.

### 3 Error representation and adaptive entropy coding

#### 3.1 Error representation

Mapping the prediction error from a large alphabet size to a small one has been used in JPEG [4]. Using this technique, the prediction error is mapped into three parts: the quantization index  $e_q(n)$  that represents the quantization level, the quantization error  $q(n)$  and the sign bit. The absolute value of the prediction error is quantized according to a quantization scheme which is determined by a number of intervals:  $[v_0, v_1), \dots, [v_k, v_{k+1}), \dots, [v_l, 256)$ .

The symbol mapping scheme is:

$$\begin{aligned} \text{If } & v_k \leq |e(i, j)| < v_{k+1} \\ \text{then } & e_q(n) = k \text{ and } q(n) = |e(n)| - v_k. \end{aligned}$$

The quantization intervals used in this paper are: [0,1), [1,2), [2,3), [3,4), [4,5), [5,6), [6,7), [7,8), [8,10), [10,12), [12,16), [16,20), [20,28), [28,36), [36,52), [52, 68), [68,100), [100,132), [132,196), [196,256). The numbers of bits required to represent the quantization errors for the above intervals are fixed. They are: 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6 and 6, respectively. For example, if the prediction error is 73, then it is in the 17th ( $k = 16$ ) interval [68,100) which requires 5 bits to record the quantization error. The prediction error is mapped into three parts:  $e_q(n) = 16$ ,  $q(n) = 77 - 68 = 9$ , and the sign bit. In this case, the quantization error is represented as a 5-bit binary number (01001) in actual coding.

For images of more than 8 bits/pixel, the absolute value of the prediction error could be greater than 255. One simple way to cope with this problem is to add one quantization interval ( $[255, 2^b)$ ,  $b > 3$ ) to the above quantization scheme. If the absolute value of the prediction error is greater than 255, then its quantization index is encoded and the pixel value is written to the output file bit by bit. Otherwise, the coding scheme remains the same. This method is efficient as long as the probability of the prediction error being greater than 255 is low. In a recent study, we have shown that this simple method works well for 12-bit medical images [3]. Another way is to extend the quantization scheme by adding more quantization intervals to cope with those prediction errors whose absolute values are greater than 255.

### **3.2 Context-based adaptive arithmetic coding**

Context-based adaptive entropy coding refers to a scheme where symbols are classified into different categories (contexts) and are encoded by using the probability model indicated by the context. We follow the common practice to calculate the context by a combination of information from the previous prediction errors with the local gradients of the image. The

formula for the calculation of the context for the current pixel is:

$$C(n) = \frac{1}{4} \text{Max}[Q_1, Q_2] + \text{Max}[e_q(n-1), e_q(n-2)] \quad (10)$$

where  $Q_1$  and  $Q_2$  are the quantization indexes of the local gradient  $|x(n-1) - x(n-3)|$  and  $|x(n-2) - x(n-3)|$ . A simple context filtering algorithm [2] is also used to smooth out the effect due to random noise and to make the resultant context a robust approximation of its optimal value.

After context filtering, the context number is hard-limited by:

$$C(n) = \text{Min}[C(n), U] \quad (11)$$

where  $U$  is a threshold determined by experiments. The limitation of the number of contexts is useful to reduce the computational complexity and memory requirements. In addition, the use of a large number of contexts may deteriorate the performance of compression because the number of pixels belonging to some contexts are so small that it is very difficult to model their statistical properties for efficient entropy coding. In all of our experiments,  $U = 20$ .

### 3.3 The coding and decoding process

In the encoder, the quantization index  $e_q(n)$  is encoded by using a multi-symbol context-based adaptive arithmetic coder. The quantity  $C(n)$  is used as the context. The quantization errors  $q(n)$  are written to the encoded file bit-by-bit. If the prediction error is not zero, then the sign bit is directly written to the encoded file.

In the decoder, the quantization index is decoded first. The quantization error can then be extracted from the data file, since the number of bits used to represent them is known. The prediction error can be re-assembled by using:  $e(n) = v_k + q(n)$ . If  $e(n)$  is greater than zero, then the sign bit is decoded from the data file and the prediction error changes its sign

according to the sign bit.

## 4 Experimental results

Our experiments were performed on three sets of images. One is the original 8 bits/pixel JPEG test image set which contains nine images whose size is  $576 \times 720$ . Another one has 18 images of various type, including human face, natural scenery, medical (mskull, MRI, xray) and radar images (airport, lax, landsat). Except for two images, “MRI” and “xray” whose size is  $256 \times 256$ , the size of all other images is  $512 \times 512$ . The third one is the TMW test image set (without the two artificially generated images “noise square” and “shape”).

In the following, APC-0 is used to represent the configuration of combining only the six fixed predictors. In all other configurations, the TDLMS predictor is combined with the six fixed predictors. APC-1 is used to represent the configuration where the training block consists of only three pixels  $W$ ,  $N$  and  $NE$ . The pixel  $NW$  is excluded from the training block because it is found experimentally that the performance of the predictor remains almost the same when it is included in the training block. APC-2 is used to represent the configuration that the size of the training block is  $(4 \times 4 \times 3)$ . APC-A and APC-P are used to represent the configurations of adaptive training block size and pixel-based training, respectively. In all of these configurations, the same context-based arithmetic coder is used to encode the prediction error. It can be seen that the above configurations represent the proposed algorithm set to different levels of complexity, ranging from the lowest (APC-0, no TDLMS predictor) to the highest (APC-2, fixed size block training). In between are: APC-P (pixel based training), APC-1 (small block training) and APC-A (adaptive block size).

Our experiments can be divided into two parts. The aim of the first part is to test the performance of the proposed algorithm in different configurations and compare it with those of other published algorithms. Results are shown in Tables 2 and 3. Comparing the average bit rates (ABR) produced by different configurations of the proposed algorithm, we can see

	CALIC[15]	JPEG-LS	HBB[12]	Lee[6]	APC-0	APC-1	APC-2	APC-A	APC-P
balloon	2.78	2.90	2.80	2.79	2.76	2.74	2.72	2.73	2.75
barb2	4.46	4.69	4.48	4.47	4.47	4.41	4.39	4.40	4.38
barb	4.31	4.69	4.28	4.20	4.29	4.14	4.04	4.04	4.20
board	3.51	3.68	3.54	3.50	3.48	3.45	3.43	3.43	3.47
boats	3.78	3.93	3.80	3.76	3.76	3.73	3.70	3.70	3.75
girl	3.72	3.93	3.47	3.70	3.67	3.63	3.61	3.61	3.66
gold	4.35	4.48	4.37	4.35	4.33	4.31	4.30	4.30	4.32
hotel	4.18	4.38	4.27	4.24	4.18	4.17	4.15	4.15	4.18
zelda	3.69	3.89	3.72	3.68	3.69	3.64	3.63	3.63	3.65
<i>ABR</i>	<i>3.86</i>	<i>4.06</i>	<i>3.89</i>	<i>3.85</i>	<i>3.85</i>	<i>3.80</i>	<i>3.77</i>	<i>3.77</i>	<i>3.81</i>

Table 2: Compression results (bits/pixel) and the average bit rate (ABR) for the JPEG test image set. Default settings for JPEG-LS are used.

that the performance of the proposed algorithm improves due to the TDLMS predictor and a larger training block. Experimental results also show that the performance of the proposed algorithms (APC-P, APC-1, APC-A, and APC-2) is consistently better than those of CALIC, JPEG-LS, HBB, and Lee’s algorithm.

The aim of the second part of the experiments is to compare the performance and the complexity of the proposed algorithm with those of JPEG-LS and TMW. JPEG-LS can be regarded as on the one extreme where the performance is sacrificed for low complexity, while the TMW can be regarded as on the other extreme where complexity is ignored for the best performance. Therefore, results presented in Table 4 provide valuable information for the evaluation of the proposed algorithm.

When we compare the proposed algorithm with JPEG-LS and TMW, we make the following observations. To achieve the performance improvement of 0.24 bits/pixel over JPEG-LS, the APC-1 algorithm is about 18 times slower than JPEG-LS. To achieve the performance improvement of 0.03 bits/pixel over the APC-A algorithm, the TMW algorithm is more than 4 times slower than APC-A. We can also observe that the compression performance of APC-2 is close to that of TMW at a fraction of its complexity.

Next, we compare the relative complexity of different configurations of the proposed

	CALIC	JPEG-LS	HBB	APC-0	APC-1	APC-2	APC-A	APC-P
lenna	3.94	4.07	3.92	3.87	3.84	3.81	3.81	3.86
lena	4.11	4.25	4.09	4.04	4.0	3.97	3.97	4.03
woman1	4.54	4.67	4.51	4.45	4.42	4.39	4.39	4.44
woman2	3.20	3.30	3.11	3.08	3.05	3.05	3.05	3.07
CT	2.16	2.23	2.20	2.14	2.12	2.10	2.10	2.12
MRI	3.15	3.36	3.10	3.03	2.97	2.89	2.89	3.03
Xray	2.59	2.46	2.42	2.36	2.35	2.32	2.32	2.39
goldhill	4.63	4.71	4.62	4.58	4.55	4.54	4.54	4.56
Hursley house	4.39	4.42	4.37	4.26	4.25	4.24	4.24	4.26
F-16	3.54	3.61	3.53	3.46	3.43	3.44	3.44	3.45
lake	4.90	4.98	4.84	4.80	4.79	4.78	4.78	4.80
crowd	3.76	3.91	3.81	3.71	3.68	3.66	3.66	3.71
peppers	4.20	4.29	4.17	4.10	4.08	4.06	4.06	4.10
lax	5.63	5.76	5.63	5.57	5.57	5.55	5.55	5.57
airport	6.55	6.71	6.42	6.52	6.51	6.51	6.51	6.51
landsat	3.99	4.08	3.97	3.94	3.93	3.93	3.93	3.93
milkdrop	3.56	3.63	3.56	3.48	3.45	3.46	3.47	3.48
mandrill	5.74	5.89	5.68	5.66	5.63	5.63	5.62	5.64
<i>ABR</i>	<i>4.14</i>	<i>4.24</i>	<i>4.11</i>	<i>4.06</i>	<i>4.04</i>	<i>4.02</i>	<i>4.02</i>	<i>4.05</i>

Table 3: Compression results (bits/pixel) and the average bit rate (ABR) for the second set of images. Default settings for CALIC, JPEG-LS and HBB are used.

	JPEG-LS	APC-0	APC-1	APC-2	APC-A	APC-P	TMW
ABR (bits/pel)	4.36	4.16	4.12	4.09	4.10	4.14	4.07
Time (sec)	1.86	15	33	174	142	22	605*

Table 4: Comparison with JPEG-LS and TMW using the TMW test image set. The time that the algorithm takes to compress the whole set of images is presented. \*This shows only the encoding time of the TMW algorithm, the time used for image analysis is not known.

algorithm by comparing the time<sup>1</sup> they take to encode the whole set of test images. We can see that APC-A is about 18% faster than APC-2, while maintaining the same performance. This confirms that adapting the size of the training block to the local properties of the image is a good solution to the complexity problem associated with the TDLMS predictor. We can also see that to achieve an improvement in average bit rate of about 0.04 bit/pixel, APC-A

<sup>1</sup>A PC with a 450 MHz Pentium-II CPU is used to test the algorithms.

is about 6 times slower than APC-P.

In the following discussion, JPEG-LS is used as a reference to demonstrate what has been achieved and the associated cost. This involves the comparison of both performance and complexity. The comparison of computational complexity is not as straight forward as the comparison of performance. The running time for an algorithm can only be regarded as an indication of the complexity. However, this is not an accurate measure because a number of factors related to the running time are ignored. A major factor is the implementation of the algorithm. A highly optimized implementation will run faster than an experimental implementation which is not optimized for speed. Other important factors include the memory requirement and the hardware architecture. The memory requirement for the proposed APC algorithm depends mostly on the size of the training window. For example, for the experimental setting of the algorithm APC-2, four rows of pixels must be stored in memory. On the other hand, if the proposed APC algorithm is run in a hardware architecture that permits parallel processing, then the WHT, the combination coefficients, the prediction by each predictor can be implemented by using the parallel processing functionality. Finally, we note that the speed of an algorithm is different for different setting of parameters. In our experiments, default settings for CALIC, HBB and JPEG-LS are used.

In general, the proposed algorithm is more complex than CALIC<sup>2</sup> and JPEG-LS<sup>3</sup>. The algorithms APC-0 and APC-P are faster than HBB<sup>4</sup> running in its default settings. Table 5 presents the relative performance and relative complexity of a number of algorithms. The results shown in this table are obtained by running those algorithms on the TMW<sup>5</sup> test image set. The interpretation of the relative performance is that if the average bit rate for JPEG-LS is 1 bit/pixel, then the average bit rate for an algorithm such as CALIC, is 0.971 bit/pixel. The interpretation of the relative complexity is that if JPEG-LS requires 1 time unit to encode

---

<sup>2</sup>As available from [ftp://ftp.csd.uwo.ca/pub/from\\_wu/v.arith/](ftp://ftp.csd.uwo.ca/pub/from_wu/v.arith/). An improved version of CALIC was described in [15].

<sup>3</sup>As available from [ftp://dspftp.ece.ubc.ca/pub/jpeg-ls/ver-2.2/jpeg\\_ls\\_v2.2.tar.gz](ftp://dspftp.ece.ubc.ca/pub/jpeg-ls/ver-2.2/jpeg_ls_v2.2.tar.gz)

<sup>4</sup>The binary executable of HBB is provided by Mr T. Seemann of Monash University, Australia.

<sup>5</sup>Binary executable and test images are available from <http://www.csse.monash.edu.au/~bmeyer/tmw/index.html>

	JPEG-LS	CALIC	HBB	APC-0	APC-1	APC-2	APC-A	APC-P	TMW
RP	1	0.971	0.964	0.952	0.945	0.938	0.939	0.948	0.933
RC	1	1.94	15.5	8.4	17.7	93.5	76.6	11.9	325.3*

Table 5: Relative performance and complexity for different algorithms. RP represents relative performance measured by dividing the average bit rate for an algorithm by that of JPEG-LS. RC represents the relative complexity measured by dividing the running time required by an algorithm to encode the whole set of images by that required by JPEG-LS. \*This shows only the encoding time of the TMW algorithm, the time used for image analysis is not known.

whole set of images, then an algorithm such as CALIC requires 1.94 time units. This table shows what has been achieved in performance improvements over that of JPEG-LS and what the associated cost is. It is clear from this table that a significant computational complexity increase only results in a very moderate improvement in compression performance.

## 5 Conclusion

This paper is concerned with an alternative adaptive prediction technique for lossless image coding. The proposed adaptive predictor combination (APC) algorithm has two distinctive properties. First, by adaptively combining a set of predictors (models) the resulting predictor outperforms any one of the individual predictors. This property is directly related to the Bayesian principle by which the costs (the prediction error) due to the uncertainty of the individual predictors is considered and minimized. Second, the TDLMS predictor with block-based training is different from the traditional LMS prediction filter in that the TDLMS algorithm is used to design a good predictor for the current pixel rather than aimed at minimizing the global mean square errors. It is clear that APC can be regarded as a data modelling technique that could be applied to other signal processing problems.

In this paper, we have demonstrated that while combining a set of simple and fixed predictors has led to good compression performance, a further combination with the TDLMS predictor which employ more causal pixels has resulted in even better performance. We

have studied several training methods for the TDLMS predictor. These include: pixel based and block based. In the block based method, we have shown that adaptively changing the block size has led to a significant reduction in computation time while maintaining the performance. We have also shown that the performance of proposed algorithm is consistently better than or comparable to those of published lossless image compression algorithms.

Using JPEG-LS as a reference to compare both the performance and the complexity of different algorithms, we have demonstrated that the compression performance seems to have reached such a level that further improvement requires a significant increase in computational cost.

## References

- [1] C. Christopoulos, A. Skodras and T. Ebrahimi, "The JPEG 2000 still image coding system: An overview," *IEEE Trans. Consumer Electronics*, v. 46, pp. 1103-1127, 2000.
- [2] G. Deng and H. Ye, "Lossless image compression using adaptive predictor combination, symbol mapping and context filtering," *Proc. IEEE Int. Conf. Image Processing*, v.4, pp. 63-67, 1999.
- [3] G. Deng, H. Ye and L. W. Cahill "Adaptive combination of linear predictors for lossless image compression," *IEE Proc. -Sci. Meas. Technol.*, v. 147, pp. 414-419, 2000.
- [4] G. K. Wallace, "The JPEG still picture compression standard," *Comm. ACM*, v.34, pp. 30-44, 1991.
- [5] S. Haykin, *Adaptive Filter Theory*, Prentice Hall, 3rd ed., 1996
- [6] W. S. Lee, "Edge adaptive prediction for lossless image coding," *Proc. Data Compression Conference*, pp. 483-490, 1999.

- [7] X. Li and M. T. Orchard, "Edge directed linear prediction for lossless compression of natural images," *Proc. IEEE Int. Conf. Image Processing*, CDROM (1999)
- [8] B. Meyer and P. E. Tischer, "TMW-a new method for near lossless compression of greyscale images," *Proc. Picture Coding Symposium*, pp. 533–538, 1997.
- [9] N. D. Memon and K. Sayood, "Lossless image compression: A comparative study," *Still image compression, Proc. SPIE*, v. 2148, pp. 8-20, 1995.
- [10] N. Memon and X. L. Wu, "Recent developments in context-based predictive techniques for lossless image compression," *The Computer Journal*, v. 40, pp. 127–136, 1997.
- [11] D. Santa-Cruz and T. Ebrahimi, "An analytical study of JPEG 2000 functionalities," *Proc. IEEE Int. Conf. Image Processing*, CDROM, 2000.
- [12] T. Seemann, P. E. Tischer and B. Meyer, "History-based blending of image sub-predictors," *Proc. Picture Coding Symposium*, pp. 147–151, 1997.
- [13] I. Tabus, J. Rissanen and J. Astola, "Adaptive L-predictors based on finite state machine context selection," *Proc. IEEE International Conf. on Image Processing*, v.1, pp. 401–404, 1997.
- [14] M. J. Weinberger, G. Seroussi and G. Shapiro, "LOCO-I: A low complexity, context-based, lossless image compression algorithm," *Proc. Data Compression Conference*, pp. 141–150, 1996.
- [15] X. L. Wu, "An algorithmic study on lossless image compression," in *Proc. Data Compression Conference*, pp. 150–159, 1996.
- [16] X. L. Wu and K. U. Barthel, "Piecewise 2D autoregression for predictive image coding," *IEEE Int. Conf. Image Processing*, pp. 901–904, 1998.

- [17] H. Ye, G. Deng and J. C. Devlin, “Least squares approach for lossless image coding,” *Proc. Fifth Int. Symposium on Signal Processing and Its Applications*, v.1, pp. 63–66, Brisbane, Australia, 1999.