Generalized Locally Adaptive DPCM

Torsten Seemann, Peter Tischer

Department of Computer Science, Monash University
Clayton, Victoria, Australia, 3168
http://www.cs.monash.edu.au/

Abstract

Image and audio data are examples of domains in which DPCM (Differential Pulse Coded Modulation) is an effective method for removing much of the linear correlation between data samples. However, most simple DPCM schemes cannot simultaneously cater for smooth signals, noisy signals, and discontinuities such as edges in images. To overcome this, many adaptive DPCM schemes have been proposed, including median predictors, gradient-based switching predictors and gradient-based blending predictors. In this paper we generalize the idea of blending predictors, and describe a powerful technique for creating locally adaptive compound predictors. The result is a prediction scheme which works well over a range of data types froms smooth to noisy, and requires very few tunable parameters. We apply this scheme to greyscale image data, colour image data, and audio data, and compare results with some of the current best adaptive DPCM predictors.

1 Introduction

Image and audio data are examples of domains in which (a) we typically have a large amount of data, (b) the data samples belong to a large alphabet (≥ 256), and (c) the data samples are usually highly correlated. Features (a) and (c) make images and audio ideal candidates for data compression, however the memory requirements of (b) preclude the naïve use of Markov models.

A common approach is to first "decorrelate" the samples so that lower order Markov models will be more effective. In DPCM, we make a prediction $\hat{f}_i$ of the current sample f_i, and only encode the prediction errors $e_i = \hat{f}_i - f_i$, which will usually have a lower zeroth order entropy. In forming the prediction, the encoder can use all the previous samples $f_0 \ldots f_{i-1}$, but by using only a small *local* subset of previous samples, there is typically little loss in compression performance. This approach has the advantages of being parallelizable and simplifying the prediction models.

The simplest prediction model is a linear one in which our prediction $\hat{f}_i$ is a linear combination of samples from some local causal neighbourhood Ω, with weights a_j.

$$\hat{f}_i = \sum_{j \in \Omega,\, j<i} a_j \cdot f_j \tag{1}$$

For one dimensional data such as audio, there is an obvious encoding order, and the local neighbourhood Ω would consists of the n previous samples. For image data, we typically encode using a raster scan order[1], in which our local neighbourhood does not necessarily consist of the most recently encoded pixels. Figure 1 shows the position and names given to the nearest neighbours relative to the current pixel f_i.

Figure 1: Compass Point Notation For Neighbouring Pixels

NWNW	NNW	NN	NNE	NENE
WNW	NW	N	NE	ENE
WW	W	f_i		

A *static* predictor is one for which the weights a_j are constant for each sample in the data. For example, the Delta predictor in audio compression has the formula $\hat{f}_i = f_{i-1}$. Table 1 lists various static predictors used throughout the literature for images, some of which are part of the existing lossless JPEG standard [1]. It is also possible to construct optimal[2] static linear predictors for a given set of data.

Table 1: Common Static Linear Predictors for Images

Predictor	Formula
Null	0
West	$\mathtt{W}$
North	$\mathtt{N}$
NorthWest	$\mathtt{NW}$
NorthEast	$\mathtt{NE}$
Plane	$\mathtt{N} + \mathtt{W} - \mathtt{NW}$
Plane-2	$\mathtt{W} + \mathtt{NE} - \mathtt{N}$
JPEG Mode 5	$\mathtt{W} + (\mathtt{N} - \mathtt{NW})/2$
JPEG Mode 6	$\mathtt{N} + (\mathtt{W} - \mathtt{NW})/2$
GradWest	$2\mathtt{N} - \mathtt{NN}$
GradNorth	$2\mathtt{W} - \mathtt{WW}$
Mean	$(\mathtt{W} + \mathtt{N})/2$
Average-4	$(\mathtt{W} + \mathtt{NW} + \mathtt{N} + \mathtt{NE})/4$
Pirsch	$(2\mathtt{W} + \mathtt{N} + \mathtt{NE})/4$

An *adaptive* predictor is one for which the coefficients a_j are varied for each prediction event based on some assessment of the local neighbourhood. Adaptive predictors usually perform better than static ones as properties such as smoothness, discontinuities, and noise content often differ throughout the data. One of the earliest examples of an adaptive image predictor is Graham's switching predictor [3]

$$\hat{f}_i = \begin{cases} \mathtt{N} & \text{if} \quad |\mathtt{NW} - \mathtt{N}| > |\mathtt{NW} - \mathtt{W}| \\ \mathtt{W} & \text{otherwise} \end{cases} \tag{2}$$

which will choose between two fixed sub-predictors $\mathtt{N}$ and $\mathtt{W}$ depending on whether it detects a horizontal or vertical edge around the current pixel. A more advanced form of this is the Gradient Adjusted Predictor [4] which computes two gradient measures $dh = |\mathtt{W} - \mathtt{WW}| + |\mathtt{N} - \mathtt{NW}| + |\mathtt{NE} - \mathtt{N}|$ and $dv = |\mathtt{W} - \mathtt{NW}| + |\mathtt{N} - \mathtt{NN}| + |\mathtt{NE} - \mathtt{NNE}|$ to choose between seven static linear sub-predictors.

[1] In a raster scan we encode one row at a time, top to bottom, left to right.

[2] Optimality with regards to a particular criterion, typically least sum of squared errors, least absolute deviations, or least-entropy. See [2] for an examination of the performance of these three criteria.

The MED predictor used in the LOCO-I [5] scheme is a nonlinear switching predictor, where the switching rule is to choose the median from a set of values produced by its sub-predictors, namely $\{\mathtt{W},\ \mathtt{N},\ \mathtt{W}+\mathtt{N}-\mathtt{NW}\}$. MED cannot choose a predicted value outside the range $\min(\mathtt{N},\mathtt{W})$ to $\max(\mathtt{N},\mathtt{W})$.

Roos *et al* [6] describe a more elaborate blending scheme by Heiß [7] which first calculates four gradients in the `N`, `W`, `NW` and `NE` directions. For example, the `NW` gradient $G(\mathtt{NW}) = |\mathtt{NWNW}-\mathtt{NW}| + |\mathtt{NNW}-\mathtt{N}| + |\mathtt{NN}-\mathtt{NE}| + |\mathtt{WNW}-\mathtt{W}|$. A large gradient value suggests the presence of a contour perpendicular to that gradient direction, hence the pixel in that gradient's direction should be given less weight in the prediction. Heiß computed the weights $a(x)$ as

$$a(x) = \frac{1}{G(x)^\gamma + \delta} \qquad \text{where} \qquad x \in \{\mathtt{W},\mathtt{NW},\mathtt{N},\mathtt{NE}\} \tag{3}$$

which are then normalized so that $\sum a(\cdot) = 1$. His scheme does not take into account the fact that his $G(\cdot)$s have different numbers of terms in them. Roos *et al* used $\gamma = 3$ and $\delta = 20$ for a set of 9 bit medical images in their paper. We point out that the coefficients $a(\cdot)$s in this scheme are always positive.

In Section 2, we describe a blending algorithm which generalizes some of the schemes just described. In the next three sections we then apply this new algorithm to greyscale image data, colour image data, and audio data. The final section then gives conclusions and future directions for the algorithm.

2 A Generalized Blending Algorithm

Consider Heiß's scheme. He blends the four neighbouring *pixel* values based on the four *gradients* aligned in the direction of each of those four pixels. To generalize switching and blending prediction schemes, we suggest an alternative interpretation. The four pixels can be thought of as the four local *sub-predictors* W, N, NE, NW, and the gradient measure as the *sum of absolute prediction errors* or the *penalty term* for each sub-predictor, evaluated at various positions within the local neighbourhood.

By moving away from the pixel–gradient view, we are no longer limited to the four simple neighbour predictors W, N, NW, NE. We can use *any* sub-predictors we choose, including nonlinear ones. We now describe a general blending algorithm which only requires a suitable set of sub-predictors, and positions near the current sample to evaluate them at, so that the gradients or *penalty terms* can be computed.

2.1 Notation

Let us assume we have a data stream f, and we are trying to to make a prediction $\hat{f}_i$ of sample f_i using only some subset Ω of the previously encoded data $f_0 \ldots f_{i-1}$. We also have a set $\mathcal{P}$ of n suitable sub-predictors P_j with $j = 1 \ldots n$. Let $h(P_j, f_k)$ denote P_j's predicted value for position f_k in the data stream.

2.2 Algorithm Blend

1. Compute a penalty term G_j for each sub-predictor P_j using the sum of absolute prediction errors it produces in the current neighbourhood Ω.

$$G_j = \sum_{f_k \in \Omega} |h(P_j, f_k) - f_k| \qquad \forall\ P_j \in \mathcal{P} \tag{4}$$

2. For each sub-predictor, compute its prediction PV_j for the current sample f_i, and clip it to the legal range of pixel values[3].

$$PV_j = h(P_j, f_i) \qquad \forall\ P_j \in \mathcal{P} \tag{5}$$

3. Each PV is then weighted inversely proportionally to its corresponding penalty term G, with the resulting blend normalized so $\sum_i (1/G_i) = 1$.

$$\hat{f}_i = \left(\sum_{j=1}^{n} \frac{1}{G_j} \cdot PV_j \right) \div \left(\sum_{j=1}^{n} \frac{1}{G_j} \right) \tag{6}$$

4. The final predicted value is taken to be the integer $\left\lfloor \hat{f}_i + 0.5 \right\rfloor$.

If exactly one G_j happens to be zero (meaning that sub-predictor P_j predicted eactly throughout the local neighbourhood) then we "switch" to that sub-predictor, taking $\hat{f}_i = P_j$. If two or more G_js are zero then we can either blend only those P_js with $G_j = 0$, or we can arbitrarily choose one P_j. Our experiments have shown that this situation does not happen often, and choosing just one of the P_js has little effect on the final entropy.

This blending approach can be easily modified to behave as a switching predictor, where we simply choose the sub-predictor with the *minimum* penalty term G. However, if $\mathcal{P}$ is chosen well, the performance of the blend should never be much worse than switching, and in many cases, it will be better than switching.

2.3 Choice of Parameters

Although Algorithm Blend is general in nature, it does require (a) a set $\mathcal{P}$ of sub-predictors which cover the range of expected behaviour in the data to be compressed, and (b) a suitable local region Ω from which meaningful penalty terms can be computed. In the following sections, we will apply Algorithm Blend to the lossless compression of greyscale image data, colour image data, and audio. In each section we will discuss the issues involved in choosing an appropriate $\mathcal{P}$ and Ω, and compare the performance of each prediction scheme to others in the literature.

3 Greyscale Image Data

3.1 Parameters

Most image data is not homogeneous, and typically consists of edges, textures, regions where the intensity varies smoothly, and a variable amount of noise.

3.1.1 Smooth Regions

A large proportion of a typical continuous-tone image would contain areas where the intensity varies in a linear or smooth manner, and these are usually easy to predict. Let us assume we are trying to predict the current pixel $\hat{f}_i$ using a linear combination of our nearest four neighbours.

$$\hat{f}_i = a_N\texttt{N} + a_W\texttt{W} + a_{NW}\texttt{NW} + a_{NE}\texttt{NE} \tag{7}$$

[3] For an image with α bits per pixel, the legal range of pixel values is 0 to $2^\alpha - 1$.

If our four neighbours have the same value, we would like to predict that same value. This requires that $\sum_i a_i = 1$. The next useful situation is to ensure that predicted value is exact if all the pixels lie in the same plane. This introduces two additional constraints for the x and y directions respectively, with the Plane and Plane-2 predictors of Table 1 being just two possible solutions. Note that these three constraints require that the solution contain at least one *negative* coefficient a_i, in order to estimate a gradient. The one dimensional analogues to the Plane-like predictors are GradWest and GradNorth (Table 1), which do simple linear extrapolation in the horizontal and vertical directions respectively.

3.1.2 Noise

Most images are not perfectly smooth, and may contain noise which is inherently unpredictable. We can model an image as an underlying smooth function with a random noise term. If we assume we can predict the smooth component exactly, then the error in the predicted value will depend only on a linear combination of the noise terms in the four neighbouring pixels.

$$e_i = a_N noise_N + a_{NW} noise_{NW} + a_N noise_N + a_{NE} noise_{NE} \tag{8}$$

If we also assume that the *noise* terms are bounded in magnitude by ϵ then the noise in the predicted value will be bounded in magnitude by

$$worst\ case\ |e_i| = \epsilon \cdot (|a_N| + |a_{NW}| + |a_N| + |a_{NE}|) \tag{9}$$

Thus, we would like to keep the sum of the absolute coefficient magnitudes as small as possible. To also maintain $\sum_i a_i = 1$, this means we also require all the coefficients to be non-negative. Hence, in the presence of noise, it is usually best to take an averaging approach like the Mean, Average-4 and Pirsch [8] predictors. Not taking this approach risks amplifying and introducing noise into the predicted value, which can never be removed. Unfortunately, the constraints on predictor coefficients for noisy and smooth regions are in contradiction.

3.1.3 Edges and Texture

Edges are the most visually important parts of an image, but they typically comprise only a small percentage of the pixel data. Thus, for lossless image coding, efforts to predict strong edges only give small improvements in compression. The four neighbour predictors W, N, NW, NE are good for predicting their corresponding edge directions, and the Plane predictor has the additional property of also being exact for ideal horizontal and vertical edges. An effective model for texture has yet to be devised, but for the purposes of this paper we may consider it to be some combination of edges and noise.

3.1.4 Final Choice

For image data we would like to use sub-predictors which cater for all the above situations. In our experiments we decided to compare three schemes, each using a small set of static linear predictors, which are listed in Table 2. All three incorporate the four neighbour predictors — it is hoped these will "detect" stronger edges and have an averaging approach in noisy areas. Blend-5 and Blend-7 additionally include some predictors with negative coefficients, which should predict better in the presence of smooth gradients.

Table 2: Blending Sub-Predictors used for Greyscale Images

Name	Predictors Used ($\mathcal{P}$)
Blend-4	W, N, NW, NE
Blend-5	W, N, NW, NE, Plane
Blend-7	W, N, NW, NE, Plane, GradWest, GradNorth

The choice of position and number of penalty terms is somewhat data dependent. We found that using 3 terms in the current pixel's immediate neighbourhood gave the best results over a range of image types. Table 3 lists the penalty terms with $e(x)$ denoting the prediction error produced by that sub-predictor at pixel x. For example, $e(\mathtt{N})$ for Plane would equal $|\mathtt{N} - (\mathtt{NN} + \mathtt{NW} - \mathtt{NNW})|$.

Table 3: Penalty terms used for Greyscale Images

Predictor	Penalty Term
W	$\lvert e(\mathtt{N})\rvert + \lvert e(\mathtt{W})\rvert + \lvert e(\mathtt{NE})\rvert$
N	$\lvert e(\mathtt{N})\rvert + \lvert e(\mathtt{W})\rvert + \lvert e(\mathtt{NE})\rvert$
NE	$\lvert e(\mathtt{N})\rvert + \lvert e(\mathtt{W})\rvert + \lvert e(\mathtt{NE})\rvert$
NW	$\lvert e(\mathtt{N})\rvert + \lvert e(\mathtt{W})\rvert + \lvert e(\mathtt{NW})\rvert$
Plane	$\lvert e(\mathtt{N})\rvert + \lvert e(\mathtt{W})\rvert + \lvert e(\mathtt{NE})\rvert$
GradWest	$\lvert e(\mathtt{N})\rvert + \lvert e(\mathtt{W})\rvert + \lvert e(\mathtt{NE})\rvert$
GradNorth	$\lvert e(\mathtt{N})\rvert + \lvert e(\mathtt{W})\rvert + \lvert e(\mathtt{WW})\rvert$

3.2 Results

We present results for a set of 8 and 12 bpp (bits per pixel) images. The 8 bpp images are the original 720×576 greyscale ISO JPEG test set, plus the standard 512×512 `lenna` and `peppers` image [9]. Three 12 bpp medical images are also included. `ref12b-0` is a smooth 512×512 CAT-scan slice from Siemens, `chestxray` is a smooth 1024×1024 chest x-ray, and `c00156` is a noisy 1755×1463 cervical x-ray.

Results are presented for three sets of the images — the originals, noisy versions and smooth versions. The noisy versions were created by adding Gaussian noise with $\mu = 0$ and $\sigma^2 = 32$ to each pixel in the original. The smooth versions are the result of convolving the original image with the 3×3 smoothing filter $\frac{1}{16}\{1, 2, 1,\ 2, 4, 2,\ 1, 2, 1\}$. The pixels were encoded in a raster scan order, and border pixels were encoded using a simple DPCM scheme.

The results are the zeroth order entropy of the prediction errors in bits per pixel. We compare our results to the simple linear predictors Pirsch (good for noisy images) and Plane (good for smooth images), as well as to GAP and MED, two of the leading DPCM-based contenders for the next lossless JPEG standard [10]. We defer inclusion of the MAP (Mean Adjusted Prediction) device and coding contexts until later, as we wish to initially restrict the prediction schemes to purely local information (MAP is a history-based technique). However, to be fair, we do adaptively scale the GAP thresholds based on the standard deviation of the prediction errors of the previous row, as described in [4].

Table 4: Zeroth Order Entropy (bpp) — Original Images

Image	Blend-4	Blend-5	Blend-7	GAP	MED	Pirsch	Plane
`balloon`	3.04	2.94	**2.93**	3.04	3.12	3.20	3.34
`barb`	5.19	5.06	**4.90**	5.13	5.20	5.44	5.49
`barb2`	5.21	5.10	**5.02**	5.05	5.18	5.43	5.51
`board`	3.93	**3.80**	3.81	3.90	3.95	4.24	4.27
`boats`	4.39	4.21	**4.16**	4.28	4.31	4.61	4.54
`girl`	4.11	3.97	**3.91**	4.11	4.21	4.35	4.39
`gold`	4.77	4.66	**4.65**	4.68	4.72	4.84	5.00
`hotel`	4.72	4.61	**4.58**	4.65	4.73	4.96	5.03
`zelda`	3.95	**3.88**	3.90	3.95	4.11	4.10	4.43
`lenna`	4.39	4.33	**4.32**	4.39	4.55	4.50	4.80
`peppers`	**4.34**	**4.34**	4.42	4.53	4.72	4.48	5.13
`c00156`	**5.90**	**5.90**	6.01	6.01	6.22	5.99	6.61
`chestxray`	5.80	5.56	**5.46**	5.67	5.72	5.92	5.75
`ref12b-0`	3.44	3.08	**2.98**	3.48	3.20	3.62	3.11

The performance of the static predictors Pirsch and Plane can often give a good indication of the noise content of an image. For example, Pirsch performs much better than Plane on the `zelda`, `lenna`, `peppers`, and `c00156` images. This is likely to be due to Plane's negative `NW` coefficient which has the effect of amplifying noise in the predicted value as described in Section 3.1.2. Images such as `chestxray`, `ref12b-0`, and `boats` all do better with Plane, as it performs well on the large smooth (and relatively noise free) areas they contain. The rest of the JPEG test set has Pirsch beating Plane by smaller amounts, so we can consider them moderately noisy.

For the original images, Table 4 shows that Blend-7, followed closely by Blend-5, did best on the 8 bpp and the smooth 12 bpp images. Those images for which Blend-4 and Blend-5 did better were the same ones previously classified as noisy. The relative failure of Blend-7 in those cases is probably due to the extra negative coefficients introduced by the GradNorth and GradWest predictors. This suggests that the absolute sum of prediction errors may not be a strong enough penalty function to separate the Blend components properly. However, it was found that squaring the weights occasionally helped for 12 bpp images, and often made it worse for the 8 bpp ones. Overall, GAP and MED were not far behind, with exceptions being the noisy `peppers` and `c00156`, the smooth `chestxray` and `ref12b-0`, and `barb` and `girl` which contain large regions of edgy textures.

The results in Table 5 show that the addition of artificial noise to the test images has an interesting effect on predictor performance. As expected, Pirsch did much better than Plane on every image, generally beating MED (except on the originally smooth images) and sometimes beating GAP. MED cannot cope well with noise as it has no mechanism to to take an averaging approach, whereas GAP is probably successfully switching to its central predictor[4].We also observe that Blend-7 has now fallen behind the first two Blends, which achieved similar results, especially on the images we originally classified as the noisiest. This suggests that the Plane sub-predictor in Blend-5 is successfully

[4] GAP is a switching predictor, with seven sub-predictors. The central predictor is for "smooth" regions, and the other six are for weak, moderate, and strong vertical and horizontal edges. The central predictor has the formula $0.5\mathtt{W} + 0.5\mathtt{N} - 0.25\mathtt{NW} + 0.25\mathtt{NE}$, which is similar to Pirsch, except it has a small negative `NW` coefficient

Table 5: Zeroth Order Entropy (bpp) — Noisy Images

Image	Blend-4	Blend-5	Blend-7	GAP	MED	Pirsch	Plane
`balloon`	4.87	4.88	4.98	4.99	5.17	4.93	5.59
`barb`	5.76	5.70	**5.66**	5.76	5.88	5.90	6.24
`barb2`	5.72	5.65	5.65	**5.64**	5.79	5.87	6.15
`board`	5.10	**5.07**	5.14	5.15	5.30	5.22	5.69
`boats`	5.28	**5.22**	5.26	5.27	5.40	5.37	5.74
`girl`	5.10	**5.08**	5.14	5.17	5.33	5.17	5.69
`gold`	5.37	**5.33**	5.39	5.38	5.51	5.39	5.87
`hotel`	5.44	**5.38**	5.41	5.42	5.55	5.54	5.88
`zelda`	5.00	**5.01**	5.09	5.10	5.28	5.07	5.69
`lenna`	5.21	**5.20**	5.26	5.29	5.47	5.27	5.82
`peppers`	**5.15**	5.16	5.25	5.31	5.47	5.23	5.88
`c00156`	**6.06**	6.07	6.17	6.18	6.38	6.15	6.77
`chestxray`	6.06	5.90	**5.86**	5.98	6.08	6.16	6.23
`ref12b-0`	5.01	**5.00**	5.07	5.12	5.23	5.09	5.64

being blended out, otherwise we would expect it to amplify the noise, resulting in a higher entropy. However, we can not say the same for the GradNorth and GradWest components in Blend-7. This could be due to the fact that the penalty terms for these two predictors need to use more *remote* pixels than the others.

Table 6: Zeroth Order Entropy (bpp) — Smooth Images

Image	Blend-4	Blend-5	Blend-7	GAP	MED	Pirsch	Plane
`balloon`	2.14	1.89	**1.79**	2.25	2.09	2.43	2.00
`barb`	4.01	3.69	**3.36**	3.95	3.86	4.27	3.83
`barb2`	4.00	3.63	**3.37**	3.81	3.79	4.26	3.71
`board`	2.68	2.32	**2.18**	2.77	2.36	3.17	2.36
`boats`	3.32	2.92	**2.71**	3.27	3.00	3.61	2.91
`girl`	3.23	2.82	**2.57**	3.29	3.21	3.59	3.00
`gold`	3.60	3.19	**2.98**	3.50	3.23	3.80	3.14
`hotel`	3.57	3.18	**2.93**	3.48	3.23	3.94	3.21
`zelda`	2.72	2.35	**2.18**	2.71	2.56	2.96	2.35
`lenna`	3.25	2.95	**2.77**	3.24	3.25	3.51	3.11
`peppers`	3.04	2.68	**2.49**	3.02	2.92	3.25	2.74
`c00156`	4.14	3.84	**3.67**	4.09	4.01	4.28	3.93
`chestxray`	5.04	4.38	**4.02**	4.99	4.78	5.42	4.29
`ref12b-0`	3.02	2.37	**2.15**	3.27	2.71	3.42	2.22

Smoothing the images has the desired effect of averaging away some of the noise, as well as smoothing edges into gradients. As expected, Table 6 reveals that Plane did better than Pirsch on all of the images. The presence of gradient-style predictors Plane, GradWest and GradNorth in Blend-7 helped it to beat all others on the smoothed images. MED also performed much better, even beating GAP on every image except `lenna`, suggesting it is better suited to smoother data. GAP and Blend-4 suffered somewhat with this smoothed data, being beaten by Blend-7 by about 0.5–0.6 bpp for the 8 bpp images, and up to 1.0 bpp for the 12 bpp ones. We suggest that this is due to Blend-4's complete inability to obtain any negative coefficients, and GAP only being able to reach -0.25`NW`. MED however, can switch to the Plane sub-predictor.

3.3 Prediction Error Modeling

Although the Blend predictors typically performed better than GAP, MED, and the static predictors in the previous section, we must take certain facts into consideration. GAP and MED are only relatively simple predictors, existing as one part of a larger overall compression scheme (CALIC and LOCO-I respectively), and have been designed with efficient implementation in mind. On the other hand, Blend is more complex. It uses more neighbouring pixels and computes more functions of those pixels.

3.3.1 Error Feedback

An important feature of CALIC and LOCO-I is the error feedback scheme (which CALIC calls MAP or "Mean Adjusted Prediction") designed to correct systematic errors within a context. To see how this affects performance, we implemented a form of MAP similar to that in [4], using $(\Delta/2)\cdot 256$ contexts, except that Δ was set to be the integer log of the standard deviation of the six nearest neighbours. The expectation counters within a given MAP context were also halved every 256 events. Table 7 has results for the prediction error entropy after MAP was applied.

Table 7: Effect of MAP (Mean Adjusted Prediction)

Image	Blend-4	Blend-5	Blend-7	GAP	MED	Pirsch	Plane
`balloon`	2.95	2.90	**2.89**	2.96	3.00	3.06	3.09
`barb`	4.92	4.84	**4.73**	4.91	4.97	5.08	5.13
`barb2`	5.00	4.94	**4.89**	4.92	4.98	5.15	5.14
`board`	3.76	3.69	**3.68**	3.75	3.78	3.96	3.93
`boats`	4.20	4.10	**4.07**	4.15	4.16	4.34	4.26
`girl`	3.95	3.87	**3.84**	3.96	4.01	4.10	4.09
`gold`	4.67	**4.61**	**4.61**	4.63	4.64	4.72	4.76
`hotel`	4.58	4.52	**4.49**	4.55	4.58	4.73	4.72
`zelda`	3.85	**3.81**	3.82	3.86	3.93	3.93	4.08
`lenna`	4.30	4.27	**4.26**	4.31	4.39	4.37	4.50
`peppers`	**4.28**	**4.28**	4.31	4.39	4.47	4.38	4.69
`c00156`	5.84	**5.83**	5.88	5.88	5.99	5.89	6.19
`chestxray`	5.57	5.45	**5.42**	5.51	5.57	5.65	5.56
`ref12b-0`	3.16	2.97	**2.94**	3.18	3.08	3.27	3.03

We can see that MAP has helped GAP and MED improve by up to 0.3 bpp compared to Table 4. However, the Blends also improved by up to 0.2 bpp, which kept them ahead of GAP and MED. This tells us that the Blend algorithm is capturing *some* features of the data that MAP was designed for, but not all of them. For all the images except `barb` and `barb2`, MAP has brought Blend-5 within 0.05 bpp of Blend-7. This is probably due to the GradWest and GradNorth-like elements within the texture component used in the determination of MAP contexts.

If we consider Blend *without* MAP compared to GAP *with* MAP, we find Blend to be much better still on the 12 bpp images, but about equal in performance on the 8 bpp images. The poor performance on 12 bpp images for GAP may be due to an inappropriate choice of switching thresholds. It is interesting to see that MAP reduces the margin between the adaptive and static predictors. For example, Pirsch+MAP beat MED on the original noisy images, and Plane+MAP beat it on the smooth ones. Before MAP, Pirsch was ahead of Plane on most of the ISO JPEG images, but after MAP, it is slightly ahead. This suggests MAP does a good job of removing systematic errors introduced by an inappropriate predictor.

3.3.2 Context Coding of Prediction Errors

Most modern DPCM schemes in the literature try to encode prediction errors based on the context of the current pixel, where the context is usually some estimate of the expected size or variance of the error. To see the effect of this context coding, we implemented a simple scheme using Δ (as defined in 3.3.1) Sign-Magnitude Bit-plane (SMB) [11] contexts. These were chosen as they adapt quickly, are easily extended to 12 bpp data, and work well with the centered (zero mean) prediction errors produced after the MAP stage. The SMB counters were halved every 256 events.

Table 8: Effect of both MAP and Coding Contexts

Image	Blend-4	Blend-5	Blend-7	GAP	MED	Pirsch	Plane
balloon	2.87	**2.83**	2.84	2.87	2.92	2.90	3.02
barb	4.55	4.50	**4.43**	4.55	4.61	4.66	4.77
barb2	4.64	4.60	**4.57**	4.60	4.66	4.71	4.83
board	3.60	**3.56**	3.57	3.60	3.65	3.68	3.82
boats	3.92	3.86	**3.84**	3.89	3.93	3.98	4.06
girl	3.83	3.78	**3.76**	3.84	3.90	3.92	4.00
gold	4.46	**4.41**	4.42	4.43	4.45	4.47	4.57
hotel	4.33	**4.29**	**4.29**	4.32	4.37	4.39	4.53
zelda	3.79	**3.77**	3.79	3.81	3.89	3.84	4.05
lenna	4.13	**4.12**	**4.12**	4.16	4.21	4.18	4.34
peppers	**4.17**	**4.17**	4.21	4.27	4.35	4.22	4.57
c00156	**5.79**	5.80	5.85	5.84	5.94	5.81	6.13
chestxray	5.38	5.29	**5.27**	5.31	5.38	5.38	5.40
ref12b-0	2.89	2.76	**2.74**	2.87	2.84	2.90	2.82

Table 8 gives the bit rates after the MAP transformed prediction errors have been context-coded. We see that once again, GAP and MED have improved slightly more than the three Blends did, reducing the difference to less than 0.1 bpp. Blend-5 has also gained on Blend-7, leaving very little between the two. Even the static predictors improved, often getting within a few percent of the best predictor. We found similar behaviour to occur for the artificially noisy versions of the images. The Blend approach however, remained up to 0.3 bpp ahead of the others on the smoothed versions, particularly the 12 bpp data.

4 Colour Image Data

4.1 Parameters

Colour image data consists of three colour bands, typically red (R), green (G), and blue (B), which can be thought of as three separate images. In lossy (non-reversible) compression, these bands are usually converted to another colour space such as Y/Cr/Cb to try to remove the inter-band correlation which exists in R/G/B space. However, the conversion itself is non-reversible unless the samples are stored at a higher precision than the original R/G/B ones, which for lossless compression, can cancel out the gain from the actual compression process. By remaining in R/G/B space, we can try to exploit the correlation to improve our predictions.

In our experiments on colour images, we took $\mathcal{P}$ to be the same as Blend-5. We also assume that the image is encoded in an interleaved fashion — the first pixel's G component, R component, B component, and then onto the second pixel[5]. We considered three methods for computing the penalty terms:

- **Method A:** This is equivalent to treating the bands as three separate greyscale images, using the penalty terms of Table 3 ("intra"-band)
- **Method B:** The 1st band was processed as in Method A. The 2nd band can now use the knowledge of the 1st band's current pixel, as it has been decoded. We set the penalty terms to be equal to be the *actual* e_is of each P_j in the 1st band, and the 3rd band used the average of the e_is in the first two bands ("inter"-band).
- **Method C:** The 1st band was also processed as before. The other two bands used the average of the intra-band (Method A) and inter-band (Method B) information that they had available.

4.2 Results

We present results in Table 9 for the 512×512 24 bpp colour images `lenna`, `mandrill`, `peppers`, and `yacht` [9]. The entries labelled with an asterisk are the best result for that colour band out of the three methods. We see that we can always do better by using some or only previously decoded inter-band information, as the penalty terms are based on actual errors, not estimated ones. However, within a given method, some bands may fare worse than just compressing them separately (eg. the R band of `peppers` and `yacht` for Method B).

Overall, Method C performed best based on total bit rates, except in the case of `mandrill`, a very textured image with some large clearly defined red and blue segments. In this case, the band information was probably more useful, and averaging in local information only corrupted the penalty terms. For the other three, it was really only a small improvement. This could be due to the inter-band information being most helpful around edges, but which only make up a small percentage of the data.

We can conclude by saying that inter-band penalty terms are cheap to incorporate, but improve results by only a small percentage when combined with the intra-band terms. One possibility may be to use *only* the inter-band terms, trading it off for a small decrease in compression but with improved throughput.

[5] There are 6 possible orderings of the colour components. The optimum ordering for a given image can not be known without running the algorithm with all six combinations and seeing which does best. We found that there was about a 1–3% variation between results for different orderings, with the optimum being image-dependent, so we only considered the single ordering G,R,B.

Table 9: Results for Colour Image Data (bits per pixel)

Band	`lenna`	`mandrill`	`peppers`	`yacht`
Method A				
G	4.75	6.40	4.85	5.58
R	4.22	6.24	4.98	5.25
B	4.97	6.37	4.87	5.43
Total	13.94	19.01	14.70	16.26
Method B				
G	4.75	6.40	4.85	5.58
R	4.19	5.99*	5.10	5.29
B	4.81*	6.15*	4.86	5.35*
Total	13.75	**18.55**	14.82	16.22
Method C				
G	4.75	6.40	4.85	5.58
R	4.12*	6.05	4.95*	5.20*
B	4.86	6.23	4.82*	5.37
Total	**13.72**	18.69	**14.63**	**16.14**

5 Audio Data

5.1 Parameters

Audio data is a 1-D signal, and is typically stored at 8 or 16 bits per sample. The sampling rate determines the quality of the audio, and common rates are between 8 and 44 kHz (CD quality). The higher the sampling rate and bits per sample, the smoother and more correlated the samples are.

Table 10: Polynomial Extrapolating Predictors for Audio

Order	Formula for $\hat{f}$
0	0
1	f_{i-1}
2	$2f_{i-1} - f_{i-2}$
3	$3f_{i-1} - 3f_{i-2} + f_{i-3}$
4	$4f_{i-1} - 6f_{i-2} + 4f_{i-3} - f_{i-4}$

If we assume we do have smooth waveform data, then polynomial extrapolation predictors like those of Table 10 may be good candidates for $\mathcal{P}$. The order $[0, 1, 2]$ predictors are like the [Null, W, GradWest] predictors used for images, with the predictor weights being the binomial coefficients.

5.2 Results

Table 11 gives results for five audio files. The first four are music (as used in [12]) and the last one is speech, and they cover a range of features. For our experiments we tried $\mathcal{P} = \{1, 2\}$, $\mathcal{P} = \{1, 2, 3\}$, and $\mathcal{P} = \{1, 2, 3, 4\}$, where "1" represents the order 1 predictor from Table 10. We defined the penalty terms to be the sum of absolute errors for each sub-predictor over the last $|\Omega|$ samples. We tried $|\Omega| = 4$, 8, and 32.

Table 11: Entropy of Audio Predictors (bits per sample)

File	CU8	DS16	DS8	MF16	SC16
Samples	1,662,976	542,071	135,517	617,400	168,000
Bits/Sample	8	16	8	16	16
Freq. (kHz)	8	44	11	22	22

Order	*Static Predictors*				
0	4.87	13.85	5.90	12.80	13.32
1	**4.15**	11.83	**5.26**	11.19	10.43
2	4.56	11.60	5.85	10.97	7.62
3	5.28	11.82	6.60	11.24	5.82
4	6.10	12.28	7.28	11.68	6.27

$\|\Omega\|$	*Algorithm Blend with* $\mathcal{P} = \{1, 2\}$				
4	4.23	11.49	5.44	10.83	7.98
8	4.22	11.47	5.43	10.82	8.17
32	4.21	11.46	5.42	10.80	7.98

$\|\Omega\|$	*Algorithm Blend with* $\mathcal{P} = \{1, 2, 3\}$				
4	4.40	11.29	5.67	10.67	6.04
8	4.38	11.26	5.65	10.65	6.12
32	4.36	11.24	5.63	10.63	5.96

$\|\Omega\|$	*Algorithm Blend with* $\mathcal{P} = \{1, 2, 3, 4\}$				
4	4.60	11.21	5.90	10.59	5.84
8	4.58	11.18	5.87	10.56	5.96
32	4.55	**11.16**	5.83	**10.54**	**5.80**

We see that the simple order 1 Delta predictor is the best performer for the two 8 bit files. As more predictors were introduced into the blend, the entropy actually increased, meaning the penalty function was not successful enough to zero out the poor performing sub-predictors. Perhaps a switching approach could work better in this situation. The 8 bit files have a lower sampling rate, and could also be very noisy, which means the higher order (≥ 2) sub-predictors may not be suitable as they contain large negative coefficients. We note that the 8 bit results were relatively insensitive to the increase in the size of Ω.

The smoother 16 bit data showed the opposite behaviour.The best results all occurred for the largest Ω and $\mathcal{P}$, and the Blend's entropy was never much worse than that of each sub-predictor's alone. We also found that increasing Ω made tiny improvements, and increasing $\mathcal{P}$ gave small decreases in performance.

6 Conclusion

In this paper we have described an algorithm for producing locally adaptive compound predictors, which only requires a small set of suitable sub-predictors, and an appropriate local region from which to compute penalty terms from. In our examples, we blended sets of linear predictors for greyscale images, colour images, and audio data.

For greyscale images, our results show Blend to be a better pure predictor than current schemes such as GAP and MED, especially on low to average noise content. As noise is increased, the difference is less distinctive, but this is expected with any prediction scheme. The main advantage of our scheme is the ability to obtain (some) negative coefficients, which are necessary for successfully predicting gradients. GAP's sub-predictors only have small negative coefficients, which tend not to be large enough in smoother areas, and MED's are too large for noisy regions. The addition of MAP and context coding still had Blend ahead by up to 0.13 bpp, without any effort to tune the sub-predictors, the coding contexts, or the MAP contexts.

This improved performance does come at some computational cost. An n component blend can be implemented using $(n+1)$ divisions and $2(n-1)$ additions, plus the cost of computing each sub-predictor and its penalty terms. In contrast, GAP and MED can be done using only binary shifts and additions.

The general nature of the Blend algorithm algorithm was shown by its ease of application to colour image and audio data. For audio, we took existing predictors known to work well, and obtained good results for 16 bit sound data. With the colour images, we used the same predictors as for the greyscale images, and only modified the penalty terms. The result was a scheme which worked slightly better than compressing the colour bands separately.

As we store more and more digital image data, with a variety of properties, resolutions, and bits per pixel, we can not expect simple prediction schemes to always perform well. A general adaptive approach such as Blend, supplied with a reasonable set of parameters, can adapt to a variety of data types without any training at all. Other possible areas of application include volume imaging, video sequences and multi-spectral data.

7 Acknowledgement

The authors would like to thank Professor W. Zschunke for making the Ph.D thesis of Dr. Heiß available to us.

References

[1] G. K. Wallace, "The JPEG Still Picture Image Compression Standard", *Comm. ACM*, 34, 4, April 1991, pp30–34.

[2] P. E. Tischer, "Optimal Predictors for Image Compression", Technical Report 189, Monash University, 1994.

[3] R. E. Graham, "Predictive Quantizing of Television Signals", *IRE WESCON Conv. Rec.*, Vol. 22, Pt. 4, 1958, pp147–157.

[4] X. Wu, "An Algorithmic Study on Lossless Image Compression", *Proc. Data Compression Conference '96*, 1996, pp150–159.

[5] M. J. Weinberger, G. Seroussi, G. Sapiro, "LOCO-I: A Low Complexity, Context-Based, Lossless Image Compression Algorithm", *Proc. Data Compression Conference '96*, 1996, pp140–149.

[6] P. Roos, M. A. Viergever, M. C. A. Van Dijke, J. H. Peters, "Reversible Intraframe Compression of Medical Images", *IEEE Trans. on Medical Imaging*, Vol. 7, No. 4, December 1988, pp328–336.

[7] R. Heiß, "Adaptives DPCM-System ohne Schaltenschwellen zur Übertragung des Luminanzanteils von Farbfernsehsignalen", Ph.D Thesis, Technical University Darmstadt, 1985.

[8] P. Pirsch, "A New Predictor Design for DPCM Coding of TV Signals", *ICC Conference Report (International Conference on Communications)*, Seattle, WA, 1980, 31.2.1–31.2.5.

[9] URL: `ftp://ipl.rpi.edu/pub/image/still/`.

[10] N. Memon, V. Sippy, X. Wu, "A Comparison of Prediction Schemes for a New Lossless Compression Standard", *1996 International Symposium on Circuits & Systems*, Vol. II, 1996, pp309–312.

[11] R. T. Worley, P. E. Tischer, "An Alternative to Gray Coding for Bit-Plane Compression" *ACSC 17*, Christchurch, New Zealand, 19–21 January 1994.

[12] S. Fox, P. E. Tischer, "Optimal Predictors for Lossless Audio Encoding", Technical Report 263, Monash University, March 1996.