Context-based Lossless Image Compression

P. E. TISCHER, R. T. WORLEY, A. J. MAEDER AND M. GOODWIN

Victorian Centre for Image Processing and Graphics, Department of Computer Science, Monash University, Clayton, 3168, Victoria, Australia

The two-dimensional method of Langdon and Rissanen for compression of black and white images is extended to handle the exact lossless compression of grey-scale images. Neighbouring pixel values are used to define contexts and probabilities associated with these contexts are used to compress the image. The problem of restricting the number of contexts, both to limit the storage requirements and to be able to obtain sufficient data to generate meaningful probabilities, is addressed. Investigations on a variety of images are carried out using the JPEG lossless mode predictors. Results indicate that a one pass arithmetic encoder using restricted contexts can lead to effective lossless image compression algorithms.

Received March 1992; accepted October 1992

1. INTRODUCTION

Recent interest in arithmetic encoding has encouraged a clear separation between the *model* for representing data and the *encoding* of information with respect to that model and has led to a new paradigm for the development of data compression algorithms [10]. With this paradigm, new algorithms are developed by concentrating on effective ways of modelling the data. In the area of text compression and universal data compression, variable-order Markov models usually form the basis of state-of-the-art data compression schemes [2]. We have applied this paradigm to the problem of exact compression of digital image data and in this paper we present a technique which makes it possible to build very complex models without requiring exorbitant amounts of memory.

Since image data is often voluminous, compression schemes which seek to reduce the volume of data considerably at the expense of absolute picture fidelity have been popular. These schemes are called *lossy*, *inexact* or *noisy* schemes as opposed to *lossless* or *exact* compression schemes which allow the original data to be recovered without errors. The main advantage of *lossy* schemes is that they provide better compression ratios by discarding some picture information, while at the same time retaining enough picture information to reconstruct a picture which is similar visually to the original. A great deal of time and effort has been invested in such schemes, particularly for applications which are bandwidth-limited such as teleconferencing and faxsimile transmission.

In 1990 a number of standards for *lossy* image compression emerged. The JPEG standard was proposed for encoding still-frame multilevel colour digital images [9]. The MPEG standard covers the transmission of moving images and also addresses the problem of synchronization of audio information with video information [4]. In December 1990 the px64 standard was adopted for transmission of video services using ISDN (Integrated Services Digital Network) channels [5]. The potential value of these video compression standards can be seen by the fact that some companies were prepared to implement the techniques in the JPEG standard in hardware, while the standard was still in the draft proposal stage.

We have chosen to concentrate on *lossless* schemes since we are primarily interested in modelling the data accurately. In the case of *lossy* schemes, such as JPEG, the image data is transformed so that it is easier to identify information which can be discarded without greatly affecting image quality. A lossy scheme may achieve further compaction by representing the remaining information inexactly. However, every lossy scheme must at some point reduce the image data to a sequence of symbols which is coded exactly. At this stage techniques developed for lossless schemes can be applied to lossy schemes. Our interest in exact schemes has also been motivated by two main concerns:

1. Research into the structured information content of digital pictures offers potential application to image processing techniques such as noise-reduction, edge detection and segmentation;
2. Archival storage of images in application areas where exact reproducibility of the data is important. If it is extremely expensive to capture the images in the first place, or if the analysis of the image data is a critical process, we should expect to be concerned about preserving that data exactly when storing it in compressed form. Possible examples are satellite images, reconnaissance images, medical images and images from deep space probes.

Our work has concentrated initially on grey-scale images but we believe it generalizes naturally to multi-spectral data, video sequences and colour images.

2. EXACT GREY-SCALE IMAGE COMPRESSION

We wish to compress grey-scale digital images for which the data values can be represented as a two dimensional

array of values, consisting of r rows of c columns of picture elements (pixels). The pixel values are regarded as small integers in the range 0 to 2^b, $4 \leqslant b \leqslant 16$, i.e. 16 to 65,536 intensity levels. $b = 8$, i.e. 256 intensity levels, is common and has been assumed in many of the examples in this paper. Grey-scale images generally range in size from a quarter million to many million pixels: we are particularly concerned with the larger images. Images are most commonly stored by rows. If a compressed image has been stored by rows, then when that image is compressed, information from previous rows and from previous columns in the current row is available to both the encoder and decoder to help in effective compression.

Differential Pulse Code Modulation (DPCM) uses information from these pixels to predict a value for the next pixel to be encoded [6]. Since the decoder has the same information available, it can reconstruct the predicted value, so the encoder needs only to transmit the difference between the actual and predicted picture values. The distribution for the prediction errors is often a highly skewed distribution. Variable length codewords or arithmetic coding can thus be used to encode the prediction errors efficiently.

We find it convenient to refer to the location of pixels relative to the current pixel by using compass notation as illustrated by Figure 1. **CP** denotes the current pixel. Thus, the **W** pixel is immediately to the left in the current row, the **N** pixel is immediately above, and so on.

The JPEG standard specifies a baseline method for the lossy compression of images. This baseline method is based on the Discrete Cosine Transform (DCT). The DCT can be used as the basis of a method for lossless compression but the JPEG committee chose not to do this, as the DCT involves floating point arithmetic. This means that a guarantee that images could be recovered without any errors while using the DCT requires the nature of the floating point arithmetic used by both the encoder and decoder to be strictly defined. To accommodate lossless compression the JPEG standard specifies methods which use DPCM. Table 1 lists the predictors specified by the JPEG standard.

In our work we aim at achieving the best possible compression and so we are prepared to consider schemes which might be quite computationally expensive. If highly efficient implementations are required, some algorithm complexity and compression efficiency could be sacrificed to achieve this. Most work in DPCM has concentrated on predictors which are easy to compute, such as predictors which use linear combinations of past pixel values, since the emphasis seems to have been on algorithms which could be implemented efficiently in hardware. Our approach has been influenced by developments in text compression where knowledge of the most recently encountered symbols is used to derive probability estimates for the likelihood that the current symbol will assume a particular value.

	NNW	**NN**	**NNE**
WNW	**NW**	**N**	**NE**
WW	**W**	**CP**	

FIGURE 1. Referring to pixels relative to the current pixel (**CP**).

TABLE 1. Predictors used in JPEG's lossless mode

Mode		*Mode*	
0	**Null**	4	**N + W − NW**
1	**W**	5	**W + (N − NW)/2**
2	**N**	6	**N + (W − NW)/2**
3	**NW**	7	**(N + W)/2**

Langdon and Rissanen have adopted a similar approach in the compression of bilevel images (images whose pixels take only two values), either black or white [3]. Their methods are context-based in that they classify the environment of the current pixel depending on the values of pixels which have already been encountered. Langdon and Rissanen showed their approach could compress the eight CCITT test documents 20–30% better than existing algorithms. Arps *et al.* describe a VLSI implementation of Langdon and Rissanen's method for compressing bilevel images and report that their algorithm achieves about 20% better compression than the CCITT algorithm on the CCITT document test set, for which the CCITT algorithm had been explicitly optimized [1]. On bilevel image data the new algorithm performed 2.0 to 4.5 times better than the CCITT algorithm.

The Langdon and Rissanen approach to compressing binary images has been extended by Todd *et al.* to multilevel images [8]. Values from neighbouring pixels are used to predict the value of the current pixel. The prediction error is then encoded. For convenience we term this algorithm as *TLR* (for Todd, Langdon and Rissanen). *TLR* is a two pass algorithm. In the first pass the distribution of prediction errors for a given predictor is determined. Error values are partitioned into a number of *classes* having approximately the same number of observations. The error classes of the neighbours of the current pixel define a *context* which is used to select a model for encoding the prediction error for the current pixel. If there are n classes and m neighbouring pixels then there are n^m contexts. A prediction error is encoded by specifying which error class the prediction error was in, and what value within that error class the prediction error took. Depending on the neighbouring pixels, one of n^m contexts is used to provide a model for encoding the error class for the prediction error of the current

pixel. To save on memory space, it is assumed that a single model can be used to describe the value for a prediction error within an error class, irrespective of the context.

If the pixel values are in the range 0 to 255 inclusive, the prediction errors are in the range -255 to 255. If we use five error classes and three neighbours then we have 5^3 contexts with five probabilities required per context, plus 511 probabilities to determine the error within the error class, so we need to maintain $n^{(m+1)} + 511 = 1136$ parameters for the model. Todd *et al.* also considered 11 error classes. In this case the model requires 15,152 parameters. They presented results on six images for five choices of predictor and for both five and 11 error classes. These results suggested that *TLR* was an effective image compression scheme and that the most suitable number of error classes was 11. It is difficult to draw conclusions as to how effective *TLR* is from this work because of the fact that there is no agreed set of standard test images on which to base comparisons with different algorithms.

3. CONTEXT MODELLING FOR IMAGE COMPRESSION

Our intention was to generalize the work of Todd *et al.* [8]. They had been particularly concerned with reducing the number of parameters required for a context-based model. With the progress of technology, memory requirements are becoming less important constraints on algorithms. We are interested in investigating improvements in compression obtained when very large numbers of contexts are used.

Contexts are patterns of neighbouring pixels, where these pixels are in that part of the image which has already been encoded and would therefore be known to the decoder at the time the current pixel needs to be decoded. The context approach is intuitively appealing. If large prediction errors have occurred in the neighbouring pixels then it is natural to expect that large errors are more likely to occur in the predicted value of the current pixel. On the other hand, when we have been able to predict the neighbouring pixel values either exactly or with small errors, we should assume that we are in some region of the picture where the intensity is changing in a regular manner and we shall be able to predict the value of the current pixel accurately. Todd *et al.* observed that large prediction errors are often related to the presence of edges in the picture, abrupt changes in intensity that invalidate the assumptions of smoothly changing values that were used to derive the predictor formulas. It was this observation that motivated their choice of the prediction error in neighbouring pixels as the means of characterizing the context of the current pixel.

There are two main drawbacks in using a large number of contexts. The first is the amount of storage required for the large number of contexts. If we wish to distinguish n contexts as well as a different probability model for each context, we need n times as much storage as for a single model. Todd *et al.* reduce the amount of storage by assuming that the probability model associated with a context can be broken into two parts, one part being shared amongst all contexts [5]. We have developed a different means to reduce the amount of storage required for a probability model.

The second drawback is that we expect a start-up overhead to be associated with any probability model for a context. This start-up overhead will become more significant if we have more contexts. For example, if we have 1000 observations for one context, then the statistics associated with the model for that context may have reached a reasonably steady state. If those observations are spread over a large number of contexts, say 250, then we may, on average, have only four observations per context and that model may not have changed much from its initial state. The single context model should therefore be capable of better performance on smaller numbers of observations whereas the model with many contexts would require many more observations before enough have accumulated for each context. The same problem occurs with the use of high-order Markov models in universal compression. Cleary and Witten use overlapping models to overcome problems associated with the start-up phase [2]. Low order models are more effective for small number of observations than higher order models, so low order models are used in preference to higher order models until enough observations have accumulated. We have used a similar approach with context-based image compression.

3.1. Reducing storage requirements

We have investigated how context-based image compression is affected by using a large number of contexts. We considered as many as tens or hundreds of thousands of contexts. If the model associated with each context was stored in full it would require 256 probability estimates and each context would require from at least 0.25 kbytes to many kbytes.

One way to reduce storage requirements is to maintain a cache of the most commonly occurring contexts. Infrequently occurring contexts can be cached out and the storage space re-used. One such approach is described in Tischer [7].

Another approach is to approximate the prediction error distribution with some distribution which can be described using a small number of parameters. There is a trade-off involved: an approximation to the distribution may lead to less compression but reduced storage costs may mean that more contexts can be used and the greater number of contexts might lead to better compression performance overall. Possible choices for such distributions are the normal distribution and Laplacian distribution [6]. Although this approach may greatly reduce the amount of storage, a disadvantage is the

computational effort required to generate probability estimates from the parameters of a distribution in a way which is compatible with the requirements of an arithmetic encoder.

The approach we have been exploring is extremely easy to implement and to interface with an arithmetic encoder. It also allows more efficient decompression. Given an n bit prediction error we can choose to regard this as n independent observations of a binary variable and to encode each bit separately. For a given context, we can maintain an overall frequency count for that context and counts for how often each of the n bits is set. Thus, $n+1$ frequency counts are required. The probability that a particular bit is set can be approximated by the observed frequency that the bit was set divided by the observed frequency for the context. The frequency counts themselves may be held in byte quantities. The probability that the prediction error will take a particular prediction value will be the product of the probabilities the model predicts for each of the bits in the representation. For 8 bit data, this product of probabilities can be the ratio of two 64 bit numbers, so using a byte to represent the frequency counts can store probabilities as accurately as if a 64 bit frequency count was kept for each possible prediction error value.

We refer to this way of economizing storage for a context as the *bit plane approach*. It has the advantage of requiring an amount of storage which is related to the logarithm of the number of values we might wish to encode. This approach is therefore particularly well suited to applications where pixels have more than 8 bits. A bit plane representation may not provide as accurate an approximation to the probability distribution of the prediction errors in a given context as a method which retains individual frequency counts, but it requires so little storage per context (9 bytes typically) that we can use many more contexts. A further advantage of the bit-plane approach is that the arithmetic decoder need only make one decision in determining whether the encoded symbol was a 1 or a 0. In the general case where the encoding alphabet takes 2^b possibilities, the decoder might need to make as many as b decisions in determining the identity of the current symbol. Thus encoding and decoding on a bit-by-bit basis should lead to simpler and quicker arithmetic decoders which might be easier to implement in hardware.

3.2. Updating the probability model

Suppose our picture has 8 bit pixels so the prediction errors are in the range $-255, \ldots, 255$. In the absence of any prior information to the contrary a model might be initialized by assuming each prediction error can occur with probability 1/511. An arithmetic encoder can encode a symbol which is expected to occur with probability p in very close to $-\log_2 p_i$ bits. A common scheme for allowing the probability estimates to change in response to the stream of symbols which have been encoded would reflect this initial assignment of probabilities by giving each symbol a frequency count of 1, so the total for all frequency counts would be 511. Each time a prediction error is encoded, its frequency count would be incremented and the overall frequency count would be incremented. Whenever the symbol occurs, the probability used to encode that symbol would be the ratio of its frequency count to the overall frequency count.

This method of generating probability estimates adapts to the nature of the source of symbols and is widely used in universal data compression schemes. When modelling grey-scale image data, we have prior knowledge about the source of the data which enables us to develop probability update schemes which adapt more quickly to the nature of the prediction errors. With grey-scale images we expect that the intensities will generally change smoothly. Thus if we observe a prediction error of -1, we would expect prediction errors which are close to -1 would be more likely to occur than prediction errors very different from -1. The general probability estimate update scheme outlined earlier increases the probability estimate for the symbol which has just occurred but decreases the probability estimates for all other symbols by $n/(n+1)$ where n is the total frequency count, not including the current symbol. Thus if the current prediction error is 0, the general updating scheme would decrease its probability estimates for -1 and 1 by the same ratio as it would decrease the probability estimates for very large prediction errors like -255 and 255. A desirable property for probability estimate update schemes for grey-scale image compression is that when a particular prediction error value occurs, values which are close to the observed value have their probability estimates raised depending on their nearness to the observed value and values which are distant from the observed value have their probabilities lowered. We term this the *locality* of the update process.

3.3. Gray-coded bit plane approach

As outlined earlier the bit plane approach enables us to save a lot of space in storing a model for a context. A question which arises is how well the bit plane approach performs with respect to updating the probability estimates. In encoding prediction errors for 8 bit pixels the prediction error will be in the range $-255, \ldots, 255$, which requires 9 bits. However, if we regard the 256 prediction values as being arranged in a circle with 255 adjacent to 0 then any value is at most 255 positions away from any other value and the prediction error can actually be encoded in 8 bits. Thus in our discussions on the locality of updating the bit plane approach we shall be assuming the use of 8 bit prediction errors.

Suppose our model has been initialized with the assumption that all values are equally likely and occur with probability 1/256. Suppose also that the first predic-

tion error is 0. Using the bit plane approach the initial state for the model would be that each bit has a 1/2 chance of being set. If 0 has occurred first, we can update the probability of a particular bit's being set to 1/3. Thus if the next prediction error is also zero, that value will be encoded in $8 \times -\log_2(2/3) = 4.680$ bits. Values with exactly 1 bit set like 1, 2, 4, etc. will be encoded in $7 \times -\log_2(2/3) - \log_2(1/3) = 5.680$ bits. Values with few bits set have their probabilities raised and are encoded in fewer bits while those with more than 4 bits set have their probability estimates lowered and take more bits to encode.

If the conventional updating scheme used in universal data compression were applied, the encoding for a prediction error of 0 would require $-\log_2(2/257) = 7.006$ bits while every other value would be encoded in $-\log_2(1/257) = 8.006$ bits. Thus, the effect of the update strategy on the bit plane case is to increase the probability estimates of other values which are close to the value which was updated while the conventional scheme lowers the probability estimates for all intensity values for the context other than the one which occurred.

The locality properties of updating the bit plane representation are not optimal. For example, if we represent -1 as a two's complement number, its representation will have all bits set and its probability in the situation we have been considering will be changed to $8 \times -\log_2(1/3) = 12.68$ bits. Thus we have the undesirable property that a prediction error of 0 will cause us to believe that the probability of a prediction error of 1, 2, 3 or 4 will also rise but that at the same time the likelihood of a prediction error of -4, -3, -2 or -1 will fall dramatically. In order to improve the grey-scale locality properties, we need to represent values in such a way that neighbouring values will agree in the greatest number of bits possible. Gray codes have this property since consecutive Gray-coded integers have bit representations that differ in only one position. In all our work using the bit-plane approach, we use Gray-coded representations of the prediction error. We have found that this can lead to increases in compression of up to 1 bit per pixel over the normal two's complement representation of the prediction error.

4. NUMERICAL EXPERIMENTS

In our experiments we have assumed that an arithmetic encoder will be used to compress image data according to the probability estimates provided by a particular model. In most cases it is sufficient to estimate the size of the compressed data from the probability estimates without actually producing the compressed data file. In some cases we have verified the compressed file estimates by producing compressed output, which was subsequently successfully decompressed to retrieve the original data. The compressed file size estimates underestimate the actual file size but we observed that the estimate was generally accurate to better than 1%.

4.1. Test images

A difficulty with work in image compression is that there are few widely recognized test image sets. Two images which we had at our disposal which are widely used are MANDRILL and LENNA. Both images were at 512 × 512 resolution and were originally 24 bit colour images. We chose to produce test images from these by taking the luminance component of the original images and treating this as an 8 bit grey-scale image.

NECKXRAY is an image which comes from a radiology application where X-ray pictures with some accompanying information are digitized for archival and transmission. As such there are large constant regions in the background and since the X-ray is of a person's neck, there are large constant background regions in the X-ray itself. The image has a resolution of 850 rows × 579 columns.

NOAA0 and NOAA2 are two bands from a multispectral image taken from a NOAA satellite. NOAA0 is taken in the visible part of the electromagnetic spectrum while NOAA2 is in the infra-red part of the spectrum. The data in these images was originally 10 bit but was reduced to 8 bits. Both images consist of 1600 rows by 2048 columns and thus take up 3.125 mbytes.

CLOUDS is a 2048 × 2048 subset of a GMS infra-red image. This image is stored and treated as having 8-bit data. In fact, the pixel values could have been stored in 7 bits but none of our programs makes explicit use of this fact. CLOUDS is the largest image at 4 mbytes.

The six test images are illustrated in Figure 2. The images in clockwise order from the top left corner are CLOUDS, LENNA, MANDRILL, NECKXRAY, NOAA2 and NOAA0. Note that the relative size of the images reflects the relative size of the test images in terms of numbers of pixels.

4.2. Single context compression

We expect the Gray-encoded bitplane approach to involve a trade-off. It will use less memory space than the approach of storing frequency counts for each of the 256 possible prediction error values but this may be at the cost of approximating the prediction error distribution less accurately. This approximation might be reflected in reduced compression efficiency. To explore the effects of this trade-off we ran a series of tests using the six test images and each of the JPEG predictors in Table 1. For each combination of predictor and image we ran two tests: one using 256 frequency counts to model the prediction error distribution and another using the Gray-encoded bitplane approach. The results are shown in Table 2.

In running these tests, we chose to ignore effects caused by picture boundaries. For example, JPEG predictor **Null** can be used for every pixel whereas other predictors cannot be used in the first row if they require values from a previous row or cannot be used in the

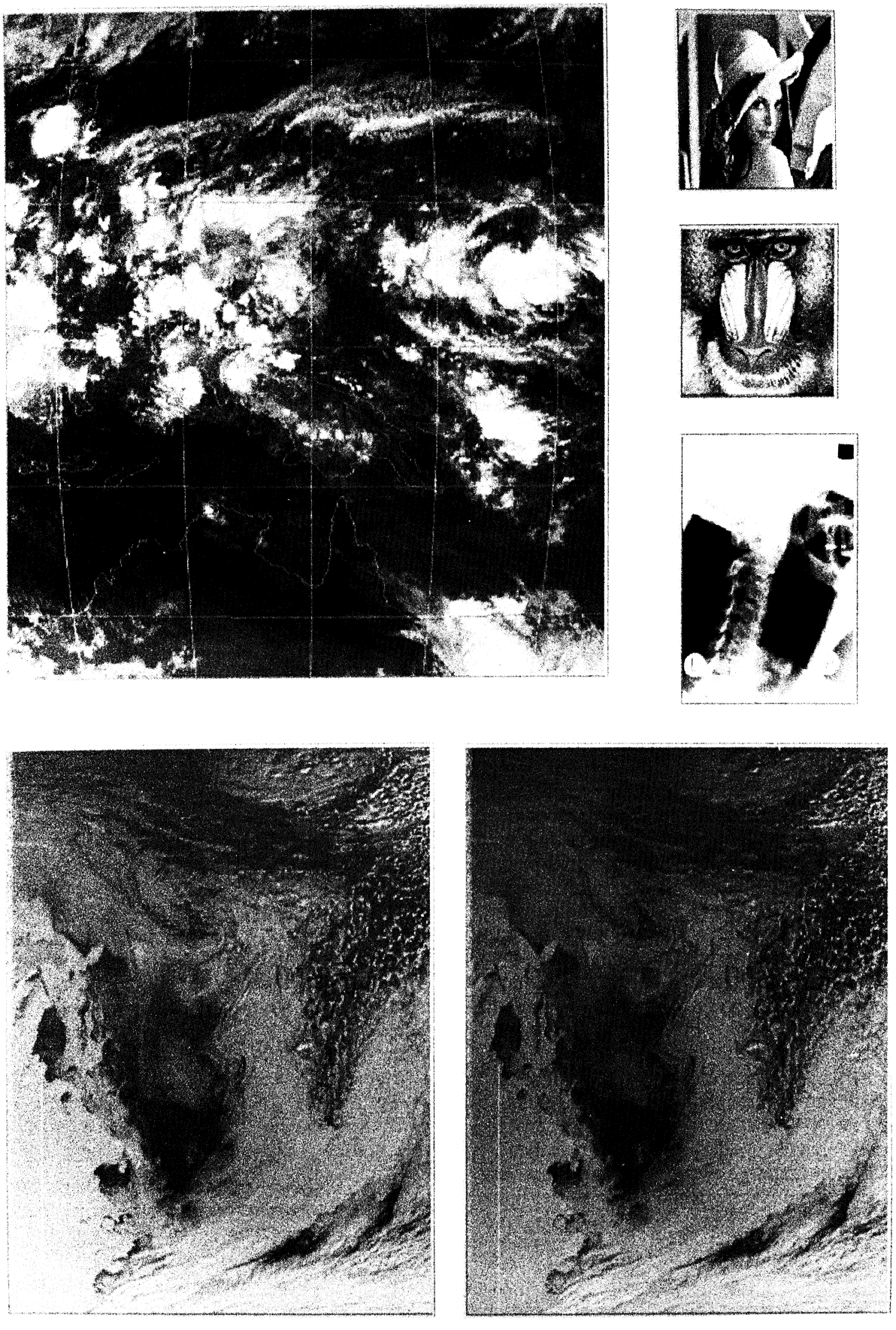

FIGURE 2. Single context compression.

TABLE 2. Performance of the JPEG predictors
Top line—compression results in bits per pixel for the Gray-encoded bitplane approach
Bottom line—compression results in bits per pixel for 256 frequency counts

	0: **Null**	**1**: **W**	**2**: **N**	**3**: **NW**	**4**: **N + W − NW**	**5**: **W + (N − NW)/2**	**6**: **N + (W − NW)/2**	**7**: **(N + W)/2**
LENNA	7.61	5.22	4.80	5.42	4.92	4.89	4.68	4.77
	7.28	5.03	4.63	5.23	4.77	4.72	4.53	4.60
MANDRILL	7.38	6.75	6.43	6.89	6.67	6.40	6.52	6.32
	7.27	6.30	6.63	6.78	6.55	6.27	6.39	6.20
NECKXRAY	6.70	3.36	3.51	3.86	3.02	3.12	3.02	3.12
	5.99	3.17	3.31	3.63	2.87	2.94	2.86	2.94
NOAA0	5.07	2.82	2.86	3.04	2.70	2.66	2.61	2.47
	5.49	2.72	2.74	2.92	2.60	2.55	2.51	2.37
NOAA2	5.24	3.86	3.60	4.03	3.54	3.41	3.54	3.38
	5.54	3.76	3.50	3.92	3.43	3.31	3.44	3.26
CLOUDS	5.22	2.93	3.06	3.63	2.59	2.77	2.65	2.96
	5.25	2.80	2.83	3.43	2.40	2.55	2.52	2.75

first column if they require values from a previous column. These pixels never constitute more than 0.4% of the pixels in the 512 × 512 images and less as the image size increases.

In addition, the frequency counts in both approaches were constrained to be in the range [1, ..., 255] so that they might be stored as bytes. When the frequency counts exceeded 255 they were re-scaled by halving. If the re-scaled count underflowed to 0 it was adjusted to 1. This re-scaling can have beneficial effects as it allows the influence of any particular observation to progressively fade and thus lets the prediction error distributions adapt as the nature of the image changes.

The results in Table 2 indicate that the Gray-encoded bitplane approach leads to less compression than the 256 frequency count approach. With the exception of four cases, three involving the **Null** predictor, the 256 frequency approach always leads to better compression—on average, 4.26% better. For the **Null** predictor, the Gray-encoded bitplane sometimes leads to better compression. We conjecture that the Gray-encoded bitplane approach may sometimes lead to better compression because it can adapt quicker to changes in the nature of the image.

In general it is difficult to say which JPEG predictor will give the best results in a given situation. The **Null** predictor will almost certainly give the worst results. In our results the **NW** predictor also consistently gave the second worst results. Predictors that involve two or more pixels constantly gave better compression. It is hard to separate the performances of the remaining predictors and to propose criteria for selecting a predictor to use with a given image.

4.3. Multiple context compression

Since the use of Gray-encoded bitplanes trades off compression efficiency against storage requirements, it is natural to consider using a number of Gray-encoded bitplane contexts. The 256 frequency count approach requires at least 258 bytes per context if 1 byte counts are maintained per frequency (the overall frequency needs at least two bytes if individual counts can approach 255). We can use as many as 29 Gray-encoded bitplanes while still needing less storage than the 256 frequency count approach. We chose to use 25 contexts as 25 is a perfect square.

We conducted a number of trials using one or two neighbouring pixels in a context. For one neighbour contexts we used 25 classes. For two neighbour contexts we used five classes per neighbour. We conducted a number of trials using either the actual pixel value of a neighbour in determining a context or using the prediction error. In the case of the prediction error contexts, the prediction errors were partitioned into five classes by using a number of partition values which had been determined by partitioning the prediction error distributions for a number of predictors applied to a suite of test images. These values were constants in our algorithms and in some cases were an inadequate partitioning of the prediction error distribution. Since many results were generated we list the best results in Table 3. The compression results reported in this table were obtained under similar conditions to those for Table 2, in that we estimated the compressed file size and ignored the effect of edge pixels. In Table 3 the second line of the entry for each row describes the context which gave the best compression. The contexts were:

- N: the value of the northern neighbour;
- W: the value of the western neighbour;
- N&W: the values of the northern and western neighbours;
- PeN: the prediction error value of the northern neighbour;
- PeW: the prediction error value of the western neighbour;
- PeN&W: the prediction error values of both the

TABLE 3. Performance of JPEG predictors with 25 Gray-encoded bitplane contexts
Top line: compression in bits per pixel
Bottom line: best context models

	0: **Null**	**1:** **W**	**2:** **N**	**3:** **NW**	**4:** **N + W − NW**	**5:** **W + (N − NW)/2**	**6:** **N + (W − NW)/2**	**7:** **(N + W)/2**
LENNA	4.95 N	4.71 PeN	4.57 PeW	4.90 PeN	4.78 PeW	4.70 PeN	4.55 PeN	4.50 PeN
MANDRILL	6.35 W	6.30 N&W	6.45 PeW	6.60 PeW	6.53 PeN	6.31 PeW	6.38 N&W	6.20 PeW
NECKXRAY	3.70 N	2.74 PeN&W	2.80 PeN	3.13 PeN&W	2.73 PeN&W	2.78 PeN&W	2.74 PeN&W	2.70 PeN&W
NOAA0	3.81 PeW	2.43 PeN&W	2.44 PeN&W	2.61 PeN&W	2.57 PeN&W	2.44 PeN&W	2.41 PeN&W	2.26 PeN&W
NOAA2	4.19 W	3.41 PeW	3.30 PeN	3.72 PeW	3.43 PeN&W	3.31 PeN&W	3.36 PeN&W	3.21 PeN&W
CLOUDS	3.44 PeW	2.33 PeW	2.38 PeN	2.85 PeW	2.36 PeN&W	2.40 PeN&W	2.30 PeW	2.47 PeW

northern and western neighbours.

The use of 25 contexts leads to a great improvement in the performance of **Null**. The improvement was always at least one bit per pixel. On the MANDRILL image the **Null** predictor actually comes within 0.15 bpp of the best performing predictor/context combination. Excluding the performance improvement for the **Null** predictor, the use of 25 Gray-encoded bitplane contexts as opposed to the use of one context leads to an average lowering by 9.32% of the bits per pixel. The improvement was most marked for the predictors which use a single pixel, i.e. **N**, **W** and **NW**.

The results in Table 3 show that for each choice of image and predictor there is a way to select 25 contexts when using Gray-encoded bitplane compression which leads to results at least as good as those achieved when using the 256 frequency count approach. In most cases the use of more contexts leads to better compression. The difficulty again arises in determining in advance which combination of predictor and contexts leads to the best compression for a particular image.

4.4. Compression using very large numbers of contexts

To avoid the problems when trying to partition the prediction error distribution into classes with nearly equal numbers of members we considered using 256 contexts. In the 25 context trials the best results were obtained when the prediction error in neighbouring pixels was used to determine the context for the current pixel so we concentrated on using the prediction error in the 256 context cases. This allows each value to determine its own unique class. We conducted trials using either the pixel value at a neighbour or the prediction error value at a neighbour. We again estimated the compressed file size in a similar manner to earlier tests and we ignored the effects of edge pixels.

The results in Table 4 show that greatly increasing the number of contexts does not necessarily lead to large improvements in the compression amount. In fact, for the NOAA0 image, the best result is 2.34 bpp while the best result for this image using 25 contexts was 2.26 bpp. The best results for three images when 25 contexts were used were obtained using prediction errors from both the northern and western neighbours and it is likely that better results for 256 contexts would be obtained if prediction errors from both these neighbours were used.

Since each context requires only 9 bytes of storage it is feasible to consider using very large numbers of contexts. If we let every distinct value of the prediction error in the northern and western neighbour define a

TABLE 4. Performance of JPEG predictors in bits per pixel with 256 Gray-encoded bitplane contexts

	N	**W**	**N + W − NW**		**W + (N − NW)/2**		**N + (W − NW)/2**		**(N + W)/2**	
	PeW	*PeN*	*PeW*	*PeN*	*PeW*	*PeN*	*PeW*	*PeN*	*PeW*	*PeN*
LENNA	4.58	4.72	4.79	4.81	4.77	4.71	4.58	4.56	4.60	4.51
MANDRILL	6.47	6.34	6.63	6.56	6.37	6.34	6.45	6.45	6.22	6.28
NECKXRAY	2.78	2.79	2.84	2.86	2.93	2.86	2.83	2.84	2.82	2.75
NOAA0	2.46	2.47	2.64	2.63	2.58	2.49	2.46	2.55	2.34	2.35
NOAA2	3.41	3.31	3.51	3.45	3.38	3.32	3.40	3.47	3.24	3.32
CLOUDS	2.32	2.37	2.38	2.51	2.47	2.47	2.30	2.66	2.46	2.84

context we have 65,536 contexts but the total memory required is still well under a megabyte.

Each context is initialized to reflect a distribution where each value is equally likely. As observations accumulate, the distributions can more closely approximate the actual distributions for a given context and compression performance improves. This start-up performance overhead is not significant when there are at least a quarter of a million pixels in the image and there are relatively few contexts. However, the overhead does become significant when the number of contexts grows to say 2^{16}. In order to reduce this effect, a back-up model which uses relatively few contexts can be maintained and used until contexts in the main model have accumulated enough observations. The results in Table 4 were generated with the intention of using one of the 256-context models as a back-up model in an algorithm which had very large numbers of contexts in the main model.

It was decided to use the 'mean' predictor, $(\mathbf{N}+\mathbf{W})/2$, with 256 contexts using the prediction error in the western neighbour. This combination of predictor and context seemed to give good compression behaviour frequently on the previous tests and could already be applied in the second row of an image. In addition, it was decided to use the 'north' predictor, **N**, for pixels in the first column and the 'west' predictor, **W**, for pixels in the first row. In other situations where neither the main model nor the back-up could be applied the 'west' predictor was used. For both the 'west' and 'north' predictors, only one context was maintained. The back-up model as well as 'north' and 'west' were updated only on those occasions when they were actually used to encode a pixel.

This algorithm was implemented in a program which generated a compressed file as output. The size of this output file agreed closely with the file size predicted using the assumption of optimal encoding. The correctness of this compression program was verified by producing a decompression program which could restore the original file from the compressed file.

Results were produced for various models using 2^{16} contexts which depended upon either the prediction error or the actual pixel value in either the western or northern neighbours. After experimenting with a number of values, a fixed threshold of 64 was selected as providing good overall results. Thus a particular context from the main model was only used to encode the current pixel if that context had already accumulated 64 observations. If it had not accumulated enough observations then the back-up model was used. The back-up model only has 1/256 as many contexts as the main model and can be expected to provide better results than the main model in the start-up phase. These results appear in Table 5.

4.5. General discussion of results

For Tables 2–5 the images are ordered in terms in size. The smallest images are LENNA and MANDRILL and the use of large numbers of contexts does not lead to improvements in the amount of compression for these images. As the size of the images increases there is some benefit in using multiple contexts. This is most clearly

TABLE 5. Performance in bpp of 256 × 256 contexts, Mean with 256 PeW as fallback

	0: **Null**	**1:** **W**	**2:** **N**	**3:** **NW**	**4:** **N + W − NW**	**5:** **W + (N − NW)/2**	**6:** **N + (W − NW)/2**	**7:** **(N + W)/2**
LENNA	4.60	4.60	4.51	4.66	4.65	4.61	4.51	4.53
MANDRILL	6.22	6.22	6.22	6.23	6.23	6.23	6.23	6.21
NECKXRAY	2.68	2.67	2.64	2.88	2.67	2.63	2.63	2.64
NOAA0	2.23	2.23	2.23	2.54	2.54	2.25	2.26	2.23
NOAA2	3.26	3.26	3.26	3.67	3.39	3.23	3.24	3.21
CLOUDS	2.27	2.23	2.16	2.70	2.32	2.19	2.17	2.29

TABLE 6. Detailed performance of best compression with 256 × 256 contexts

	Predictor	*Context*	*Overall*	*Main Model*		*Fallback Model*	
			bpp	*bpp*	*% used*	*bpp*	*% used*
LENNA	**N**	PeN&W	4.51	4.21	73.53%	5.35	26.06%
MANDRILL	**(N + W)/2**	PeN&W	6.21	5.39	29.96%	6.56	69.46%
NECKXRAY	**W + (N − NW)/2**	N&W	2.63	2.06	61.58%	3.54	38.11%
NOAA0	**(N + W)/2, Null**	N&W	2.23	2.10	95.31%	4.28	5.77%
NOAA2	**(N + W)/2**	PeN&W	3.21	3.09	95.31%	5.71	4.53%
CLOUDS	**N**	N&PeW	2.16	2.03	93.52%	4.12	6.38%

seen in the results for NOAA0, NOAA2 and CLOUDS but the improvement is not large: of the order of 0.1 bpp. As the number of contexts grows the influence of the predictor diminishes. If every combination of the values of the northern and western neighbours is used to define a context then any predictor which uses only the value of the northern and/or the western neighbour will predict a constant value for that context. For example, if the northern neighbour takes the value 96 and the western neighbour the value 94 for a particular context, then the averaging predictor, $(\mathbf{N} + \mathbf{W})/2$, will always produce the same predicted value, 95. Thus the prediction error distributions for predictors in such contexts will be shifted versions of the predicted error distributions of the **Null** predictor in the same context.

More detail on the best results is given in Table 6. Of particular interest is the proportion of pixels which were encoded using the main model, the compression efficiency that was achieved when the main model was used and the compression efficiency when the backup model was used to encode the current pixel. For the best result on the 'MANDRILL' image the main model was used only 29.96% of the time, so the performance of the back-up model became important. For the largest three images, the main model could be used for over 94% of the time and the performance of the back-up model has less impact on overall performance.

5. CONCLUSIONS

The Gray-encoded bitplane approach offers a compact way of approximating the prediction error distributions which arise when using DPCM in image compression. The great savings in memory requirements which are possible when using this approach allows the use of context-based techniques which can use large numbers of contexts to improve compression. There are many degrees of freedom in the techniques which remain to be explored and it is not clear what the best choices for these degrees of freedom will be.

The use of the Gray-encoded bitplane approach for image compression is particularly well suited to applications where the pixels may take a large range of values, e.g. 10- or 12-bit data. The results in this paper indicate that large numbers of contexts do not lead automatically to large increases in the amount of compression. This is a useful result in the sense that we can generalize context-based methods for image compression to multispectral images. Multispectral images can arise in remote sensing and RGB colour images can also be regarded as multispectral images in this way. These images can be regarded as 3D grey-scale images and the range of possible contexts for the current pixel in such an image is huge.

REFERENCES

[1] R. B. Arps, T. K. Truong, D. J. Lu, R. C. Pasco and T. D. Friedman, A multipurpose VLSI chip for adaptive data compression of bilevel images, *IBM J. Res. Develop*, **32**(6), pp. 775–795 (1988).

[2] J. G. Cleary and I. H. Witten, Data compression using adaptive coding and partial string matching, *IEEE Trans Commun.*, COM-32(4), pp. 396–402 (1984).

[3] G. G. Langdon and J. Rissanen, Compression of black-white images with arithmetic coding, *IEEE Trans Commun.*, COM-29(6), pp. 858–867 (1981).

[4] D. Le Gall, MPEG: A video compression standard for multimedia applications, *Comm. ACM*, **34**(4), pp. 46–58 (1991).

[5] M. Liou, Overview of the p time 64 Kbit/s video coding standard, *Comm. ACM*, **34**(4), pp. 59–63 (1991).

[6] A. N. Netravali and B. G. Haskell, *Digital Pictures: Representation and Compression*, Plenum Press, New York, 1988.

[7] P. E. Tischer, A modified Lempel-Ziv-Welch data compression scheme, *Aust. Comput. Sci. Commun.*, **9**(1), pp. 262–272 (1987).

[8] S. Todd, G. G. Langdon and J. Rissanen, Parameter reduction and context selection for compression of grey-scale images, *IBM J. Res. Develop*, **29**(2), pp. 188–193 (1985).

[9] G. K. Wallace, The JPEG still picture compression standard, *Comm. ACM*, **34**(4), pp. 30–44 (1991).

[10] I. H. Witten, R. M. Neal and J. G. Cleary, Arithmetic coding for data compression, *Comm. ACM*, **30**(6) pp. 520–540 (1987).