Parameter reduction and context selection for compression of gray-scale images

by Stephen Todd
Glen G. Langdon, Jr.
Jorma Rissanen

In the compression of multilevel (color or gray) image data, effective compression is obtained economically by judicial selection of the predictor and the conditioning states or contexts which determine what probability distribution to use for the prediction error. We provide a cost-effective approach to the following two problems: (1) to reduce the number of coding parameters to describe a distribution when several contexts are involved, and (2) to choose contexts for which variations in prediction error distributions are expected. We solve Problem 1 (distribution description) by a *partition* of the range of values of the outcomes into *equivalence classes*, called *buckets*. The result is a special *decomposition* of the error range. Cost-effectiveness is achieved by using the many contexts only to predict the bucket (equivalence class) probabilities. The probabilities of the value within the bucket are assumed to be independent of the context, thus enormously reducing the number of coding parameters involved. We solve Problem 2 (economical contexts) by using the *buckets* of the surrounding pixels as components of the conditioning class. The bucket values have the desirable properties needed for the error distributions.

1. Introduction

In this paper we describe an approach to the lossless compression of gray-scale images. The approach is similar to prior black/white image compression techniques [1]. In [1], to encode each bilevel pixel (picture element) value z we look at the values in its previously encoded neighbors, and use these values as a conditioning state, or *context* w. For each context w we collect statistics on the probability of the value of the pixel to be encoded, and use these probabilities as expectations to drive an arithmetic encoder. We use the term *parameter* to denote a coding probability or other quantity similarly related to the coding probability. (For example, the length of a Huffman codeword is related to minus the log of the coding probability of the event which the codeword represents.)

For black/white images, the use of the m neighboring pixel values to define each context yields 2^m distinct contexts. (In [1], m was 7 or 10.) For a straightforward extension of [1] to gray-scale images with $|z|$ levels, we would have $|z|^m$ contexts, each context requiring $|z|$ coding probabilities. The value $|z| \times z^m$ is quite large for 8-bit pixels ($|z| = 256$) and three neighboring pixels as contexts; this yields too large a number of parameters to be conveniently stored and/or transmitted.

Most of the work on compressing images deals with lossy techniques; see Netravali and Limb [2]. Both lossy and lossless image compression can benefit from prediction techniques (see Mussman [3]). In medical imaging, the application requires lossless techniques. For example, see Lehmann and Macovski [4], who use a predictor and adaptively select which of four Huffman codes to use on the basis of the mean prediction error of neighboring pixels. In a sense, the Lehmann and Macovski approach has four *contexts*, and a code for each context; they reduce the number of codewords (coding parameters) needed by using 81 codewords for the most common error values, and a "copy codeword" for a fixed-length error value to follow.

The approach in this paper reduces the number of coding parameters while using a large number of contexts. Although we study images of 8-bit pixels, i.e., illumination values from 0 through 255, the approach is also valid for 12-bit pixels or any other resolution.

The ordinary prediction methods need only as many coding parameters as are required to describe the prediction error. A linear combination of the surrounding pixels provides a prediction of the current pixel value, and a single distribution is needed to encode the difference. In general, the difference may be any value from +255 to −255. However, for most images the value 0 should be the most popular error value. The predictive method is a powerful way to take into account the values of the neighbors and generate a single distribution instead of, say, 256 × 256 × 256 distributions. However, increased compression can be achieved by the assignment of an independent distribution to each context w_i. For example, if the current pixel is in a "smooth" area, context w_1, the error distribution $e(w_1)$ should be concentrated near "0," with a high probability for error "0." On the other hand, if the current pixel is in context w_2, an area of the picture which has "edges" (sharply changed values), the prediction error $e(w_2)$ is expected to be large (relatively small probability for error of "0"). Clearly, we are dealing with different distributions.

Consider the case where we have 625 contexts, hence 625 distributions. Each distribution deals with 511 (+255 to −255) coding probabilities. We are now dealing with 625 × 510 parameters. We further reduce this number by

1. Reducing the number of parameters needed to characterize the error distributions $e(w_i)$, regardless of how the contexts w_i are chosen.
2. Proposing a simple method to determine powerful contexts w_i.

The next section describes two plane predictors. The third section describes alternatives predictors, and the fourth section discusses parameter reduction and context selection. The fifth section describes an experiment, with results in the sixth section and conclusions in the last section.

$z(i-1, j-1)$	$z(i-1, j)$	$z(i-1, j+1)$
$z(i, j-1)$	$z(i, j)$	

Figure 1

Diagram of two-dimensional pixel coordinate system.

2. Two plane predictors for gray-scale images

Figure 1 illustrates a portion of the two-dimensional matrix of the illumination levels $z(i, j)$ of an image, where value i denotes the row and value j denotes the column. [We can view levels $z(i, j)$ as a third dimension.] Relative to pixel $z(i, j)$ at the origin, the figure shows a number of its neighboring elements with their coordinates.

Following Harrison [5], we can pass a unique plane in 3-space through the location (i, j) of the pixel to be encoded. Let the pixel value $f(i, j)$ be the height of the plane.

Let us pass the plane P_1 through the three already encoded pixel points $(i-1, j-1)$, $(i-1, j)$, and $(i, j-1)$ of respective heights $z(i-1, j-1)$, $z(i-1, j)$, and $z(i, j-1)$. Plane P_1 can be used as function f to estimate or "predict" the pixel value $z(i, j)$ as follows. Define

$$C = z(i, j-1) + z(i-1, j) - z(i-1, j-1). \tag{1}$$

In terms of C,

$$f(i, j) = C \text{ for } 0 \le C \le 255,$$

and

$$f(i, j) = 0 \text{ for } C \le 0 \text{ and } f(i, j) = 255 \text{ for } C \ge 255. \tag{2}$$

We can convert the two-dimensional array $\{z(i, j)\}$ to a linear one $\{z(t)\}$, $t = 1, 2, \cdots$, obtained by scanning the picture row by row (raster-scan order), starting in the uppermost row and progressing from left to right. Use index 0 for the first row and column, and let there be M columns. If the "current" pixel $z(i, j)$ is $z(t)$, the *prediction error sequence* $\{e(t)\}$ for $0 \le f(0, 0) \le 255$ can be calculated from the equation

$$\begin{aligned} e(t) &= z(t) - f(i, j) \\ &= z(t) - z(t-1) - z(t-M) + z(t-M-1), \end{aligned} \tag{3}$$

where M denotes the number of pixels in a row. In (3), put $z(t) = 0$ for $t < 1$. Importantly, the transformation (3), taking the scanned sequence to the error sequence, is one-to-one.

If the plane predictor P_1 above does its job properly, the associated error sequence $e(t)$ is purely random. To test this we displayed the absolute values of the errors for an image of a face using model P_1. It turned out that the errors looked quite random except for the contours, from which the image still could be recognized as a face. Consequently, there must have been information left in the error sequence, which by suitable processing could be taken advantage of to improve compression.

A second plane predictor, called P_2, when passed through the three pixels $z(i-1, j-1)$, $z(i-1, j+1)$, and $z(i, j-1)$, gives the estimate $f'(i, j)$ as follows. Let

$$C = z(i, j-1) + [z(i-1, j+1) - z(i-1, j-1)/2],$$

so that in terms of C

$$f'(i, j) = C \text{ for } 0 \le C \le 255,$$

$$f'(i, j) = 0 \text{ for } C \le 0,$$

and

$$f'(i, j) = 255 \text{ for } C \ge 255,$$

which yields the error sequence

$$\begin{aligned} e(t) &= z(t) - f'(0, 0) \\ &= z(t) - z(t-1) - [z(t-M+1) - z(t-M-1)]/2. \end{aligned}$$

3. Three simple predictors

In this section we describe three predictors for which the error is very simple to compute. We have the trivial predictor

$$P_0\text{: } e(t) = z(t).$$

The remaining two predictors respectively use for the prediction the previous pixel (horizontal predictor P_h) and the pixel on the scan line above (vertical predictor P_v):

$$P_h\text{: } e(t) = z(t) - z(t-1),$$

$$P_v\text{: } e(t) = z(t) - z(t-M).$$

4. Parameter reduction and context selection

When the current pixel is in the vicinity of an edge, the error associated with the surrounding pixels tends to be larger. Thus, by combining *contexts* with the prediction model, we can improve the compression.

In this section, we introduce "buckets" of predicted values to reduce the number of parameters for the probability distributions when contexts are employed. We also show a second use for the "bucket": as a component in the context itself.

Problem 1—too many parameters to store

If we consider 100 contexts, for example, with 511 parameters per context, then we must deal with 51 100 parameters. We can reduce the number of parameters needed by partitioning the error range into equivalence classes called *error buckets*, and only predicting the bucket value. Consider, for example, dividing the range $+255 \cdots 0 \cdots -255$ into five buckets. For 100 contexts, we need five coding probabilities for each of the 100 distributions on the buckets, so we need 500 error bucket parameters. Suppose now that we assume that the error value within the bucket is independent of the context. If we have approximately 103 values in each of the five buckets, we need an additional 5×103 or 515 parameters for values within the error buckets. We have thus retained 100 contexts, but reduced the parameter-handling problem from 51 100 parameters to $500 + 515 = 1015$ parameters and effected a large cost reduction (greater than 50 to 1) in parameter storage.

Problem 2—too many contexts

The single-context prediction model showed distinct sensitivity to the presence of edges. In these areas, the prediction errors are large, giving rise to a flatter distribution about error value 0. We therefore suggest that the bucket values, employed in Problem 1 as equivalence classes for the predicted values, also be used as the components of the context for selecting the probability distributions. Thus, for a five-bucket range, let the error buckets of the three surrounding pixels $(i-1, j-1)$, $(i-1, j)$, and $(i, j-1)$ define a $5 \times 5 \times 5 = 625$ context model for conditioning the error distribution. The ideas discussed were tested, and the results are given in the next section.

We may add additional components to the context. For example, the range of intensity values may itself be partitioned into equivalence classes for use as a context component. Moreover, the equivalence classes employed in the solution to Problem 1 need not be the same as those employed in the solution to Problem 2.

- *Performance calculation*

When the picture is scanned under a simple predictor, let the number of times $e(t) = k$, $k = -255, \cdots, 255$, be denoted $n(k)$. Then the entropy of the N-pixel image, I, as an approximation to the ideal code length of the picture relative to these linear models, is given by

$$I = N \log N - \sum [n(k) \log n(k)],$$

where the sum is taken over all values k of the prediction error, and N is the sum of the $n(k)$, which also gives the number of the pixels in the image. This formula is derived in [6]. With arithmetic coding the picture can be encoded without any loss of information with a code length exceeding the ideal by a fraction of a percent. Hence, we can estimate the potential compression from the entropy calculation. Given I, the *per-pixel* entropy H is

$$H = (1/N) \times I.$$

Figure 2

Image of the face on an IEEE test chart.

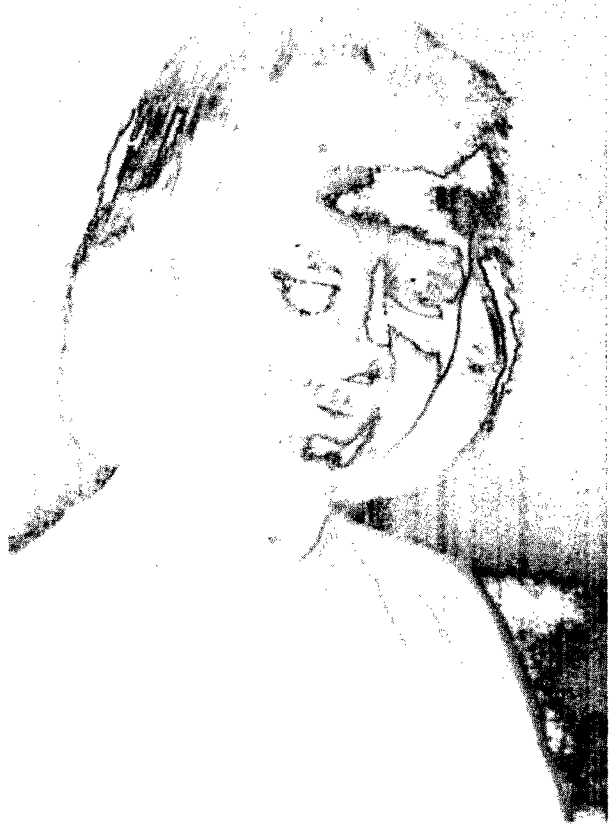

Figure 3

Error image of the face using predictor P_2.

5. The experiment

The 511 values of possible error were partitioned into five equivalence classes such that each had roughly speaking the same number of occurrences. A first pass of each image determined the buckets. Because most errors have a small absolute magnitude and cluster about 0, bucket 0 includes small errors of both signs, bucket 1 includes medium-sized positive errors, and bucket 2 large positive errors, while buckets 3 and 4 are the negative counterparts of buckets 1 and 2, respectively. Three context components, each with five values, yield 5^3 contexts w, each needing 511 coding parameters $P(e/w)$. Following the solution to Problem 1 (reduce parameter per error distribution), we use five buckets b as the equivalence class for the range of $e = e(t)$. If $b(e)$ (the bucket corresponding to the equivalence class for e) is a variable ranging over these buckets for e, we can write $P(e/w)$ as follows:

$$P(e/w) = P[e/b(e), w] \times P[b(e)/w].$$

As in Problem 1, we reduce parameters with an approximation and assume that the distribution within each bucket is independent of the context, and so replace $P[e/b(e), w]$ with $P[e/b(e)]$.

Now we need only collect the 3125 (625×5) bucket probabilities, plus 511 probabilities to determine the error within the bucket, or 3636 parameters. As part of the experiment, we tested the dependence on the number of error buckets by running an experiment with 11 buckets as well.

We give the formula for the entropy of the image in terms of the various occurrence counts. Let $n(e/b, w)$ denote the number of times value $e(t) = e$ occurs in the bucket b at the conditioning class w. First we calculate the contribution to the N-pixel image entropy from the error within the bucket, $I(e/b)$. Letting

$$n(e/b) = \sum_{w} n(e/b, w), \; n(b) = \sum_{e \, \mathrm{in} \, b} n(e/b),$$

Table 1 Compression results in bits per pixel for six images.

Model	No conditioning	Buckets		Image
		5	11	
P_0	6.80	4.77	3.94	
P_1	3.10	2.87	2.80	
P_2	3.74	2.89	2.65	face
P_h	4.09	3.10	2.78	
P_v	3.28	2.82	2.70	
P_0	6.62	5.01	4.47	
P_1	5.79	4.89	4.67	
P_2	5.56	4.76	4.54	face with noisy background
P_h	5.50	4.69	4.48	
P_v	5.37	4.61	4.41	
P_0	7.44	5.71	4.98	
P_1	5.05	4.71	4.19	
P_2	5.22	4.74	4.40	house
P_h	5.34	4.79	4.22	
P_v	5.57	4.91	4.35	
P_0	6.05	4.40	3.86	
P_1	4.32	4.13	3.52	
P_2	4.13	3.98	3.68	landscape (satellite photograph)
P_h	4.15	3.98	3.68	
P_v	4.17	3.96	3.35	
P_0	5.01	3.03	2.27	
P_1	1.98	1.77	1.69	
P_2	1.61	1.47	1.37	landscape (satellite photograph)
P_h	2.05	1.71	1.54	
P_v	2.09	1.73	1.56	
P_0	6.80	5.12	4.42	
P_1	5.07	4.50	4.23	
P_2	4.82	4.30	4.02	liver (ultrasound image)
P_h	5.06	4.31	4.00	
P_v	5.54	4.60	4.17	

then

$$I(e/b) = \sum_b n(b) \log n(b) - \sum_{b,e} n(e/b) \log n(e/b).$$

Next the contribution to the N-pixel image entropy from determining the error bucket, denoted $I(b/w)$, is calculated as follows. Letting

$$n(b/w) = \sum_e n(e/b,\ w),\ n(w) = \sum_b n(b/w),$$

then

$$I(b/w) = \sum_w n(w) \log n(w) - \sum_{b,w} n(b/w) \log n(b/w).$$

The per-pixel entropy is

$$H = (1/N) \times [I(e/b) + I(b/w)].$$

6. Results

We calculated the entropy of six images without a predictor (P_0), and relative to the four linear predictors P_1, P_2, P_h, and P_v (see **Table 1**). The first column identifies the predictor, and the second column shows the result (i.e., per-pixel entropy in bits) with only the prediction error (a single context). The third and fourth columns give the result using respectively three five-bucket context components (625 contexts) and three eleven-bucket error distributions (1331 contexts), with respectively five and eleven buckets (equivalence classes) for the predicted error.

Two of the images are photographs of a face which forms part of an IEEE test chart (see **Figure 2**). The background of the original scan is heavily dotted with noise, and Fig. 2 is a cleaned-up version. See **Figure 3** for the error image. The third image, shown in **Figure 4**, is a photograph of Hursley House, where the IBM United Kingdom Development Laboratory is located. The fourth and fifth images in Table 1 are satellite pictures of landscape. The last test image is an ultrasound scan of a human liver typically used for medical diagnosis.

7. Conclusions

We can draw a number of conclusions from the results shown in Table 1. The most striking result is that it is nearly irrelevant which of the four last linear models is used if the error sequence is *conditioned* under the bucket context components: The end result is virtually the same with them all. Second, conditioning appears to be quite effective, above all when eleven buckets are used. An increase of the bucket number from eleven improves the compression only slightly.

There is a great difference in the results between the first and the second picture of a girl's face. The reason is that the background in the latter picture is dotted with high-frequency noise, which has a lot of entropy, and which, as expected, causes the plane models to be quite poor—in fact, worse than no model at all.

Figure 4

Image of Hursley House.

References

1. Glen G. Langdon, Jr. and Jorma Rissanen, "Compression of Black-White Images with Arithmetic Coding," *IEEE Trans. Commun.* **COM-29,** No. 6, 858–867 (June 1981).
2. A. N. Netravali and J. O. Limb, "Picture Coding: A Review," *IEEE Spectrum* **68,** No. 3, 366–407 (March 1980).
3. H. G. Mussman, "Predictive Image Coding," *Advances in Electronics and Electron Physics, Suppl. 12,* Academic Press, Inc., New York, 1979.
4. L. A. Lehmann and A. Macovski, "Data Compression of X-ray Images by Adaptive DPCM Coding," *SPIE Conference on Digital Radiography,* Stanford University, CA, September 1981, pp. 396–404.
5. C. W. Harrison, "Experiments with Linear Prediction in Television," *Bell Syst. Tech. J.* **31,** 764–783 (July 1952).
6. J. Rissanen and G. G. Langdon, Jr., "Universal Modeling and Coding," *IEEE Trans. Info. Theory* **IT-27,** No. 1, 12–23 (January 1981).

Received August 9, 1984; revised October 27, 1984

Glen G. Langdon, Jr. *IBM Research Division, 5600 Cottle Road, San Jose, California 95193.*

Dr. Langdon received the B.S. from Washington State University, Pullman, in 1957, the M.S. from the University of Pittsburgh, Pennsylvania, in 1963, and the Ph.D. from Syracuse University, New York, in 1968, all in electrical engineering. He worked for Westinghouse on instrumentation and data logging from 1961 to 1962 and was an application programmer for the PRODAC computer for process control for most of 1963. In 1963 he joined IBM at the Endicott, New York, development laboratory, where he did logic design on small computers. In 1965 he received an IBM Resident Study Fellowship. On his return from Syracuse University, he was involved in future system architectures and storage subsystem design. During 1971, 1972, and part of 1973, he was a Visiting Professor at the University of São Paulo, Brazil, where he developed graduate courses on computer design, design automation, microprogramming, operating systems, and MOS technology. The first Brazilian computer, called Patinho Feio (Ugly Duckling), was developed by his students at the University of São Paulo during his stay. He is author of *Logic Design: A Review of Theory and Practice*, an ACM monograph, and coauthor of the Brazilian text *Projecto de Sistemas Digitais*; he has recently published *Computer Design.* He joined the IBM Research laboratory in 1974 to work on distributed systems and later on stand-alone color graphic systems. He has taught graduate courses on logic and computer design at the University of Santa Clara, California. He is currently working in data compression. Dr. Langdon received an IBM Outstanding Innovation Award for his contributions to arithmetic coding compression techniques. He holds ten patents.

Jorma J. Rissanen *IBM Research Division, 5600 Cottle Road, San Jose, California 95193.*

Dr. Rissanen is a member of a computer science department at the IBM Research laboratory in San Jose. He joined IBM in 1960 at the IBM Nordic Laboratories, Stockholm, Sweden, and transferred to the Research laboratory in San Jose in 1965. He spent the year 1973–74 at Linköping University, Sweden, as Professor of Control Engineering. His current research interests are in combinatorial information theory, algebraic system theory, and estimation problems in time series. He received the Doctor of Technology Degree in control theory and mathematics in 1965 from Helsinki Institute of Technology, Finland.

Stephen J. P. Todd *IBM United Kingdom Scientific Centre, Athelstan House, St. Clement Street, Winchester, Hants SO23 9DR, England.*

Mr. Todd joined the IBM Scientific Center in 1971 and has worked there since, except for a two-year assignment at the Research Division in San Jose, California. During this assignment, he researched text processing and coding theory. Most of his other work with IBM has been on relational data base systems, with some work on automated offices and image processing. He is currently working on graphics. Mr. Todd received a B.A. in mathematics from Oxford University, England, in 1968, and an M.A. in 1969.