Context-Based, Adaptive, Lossless Image Coding

Xiaolin Wu, *Senior Member, IEEE,* and Nasir Memon, *Member, IEEE*

***Abstract*—We propose a context-based, adaptive, lossless image codec (CALIC). The codec obtains higher lossless compression of continuous-tone images than other lossless image coding techniques in the literature. This high coding efficiency is accomplished with relatively low time and space complexities. CALIC puts heavy emphasis on image data modeling. A unique feature of CALIC is the use of a large number of modeling contexts (states) to condition a nonlinear predictor and adapt the predictor to varying source statistics. The nonlinear predictor can correct itself via an error feedback mechanism by learning from its mistakes under a given context in the past. In this learning process, CALIC estimates only the expectation of prediction errors conditioned on a large number of different contexts rather than estimating a large number of conditional error probabilities. The former estimation technique can afford a large number of modeling contexts without suffering from the context dilution problem of insufficient counting statistics as in the latter approach, nor from excessive memory use. The low time and space complexities are also attributed to efficient techniques for forming and quantizing modeling contexts.**

***Index Terms*—Adaptive prediction, entropy coding, image compression, statistical context modeling.**

I. Introduction

RECENT years have seen an increased level of research in image compression. Most of the effort, however, has focused on the development of lossy compression techniques. Certain applications, such as medical imaging, image archiving, and remote sensing, require or desire lossless compression. Furthermore, lossless compression is often a necessary last step in many lossy image compression systems. Hence, the development of efficient and effective lossless image compression is an important research topic that has many potential applications.

Statistical modeling of the source being compressed plays a central role in any data compression system. Fitting a given source with statistical models is a difficult endeavor pursued by researchers in a wide range of disciplines including source coding, machine learning, game theory, and estimation. It is not the intention of this paper to give this subject a thorough theoretical treatment (the reader is referred to [6] and [9] for concepts and background on universal statistical data modeling). Instead, we study practical algorithmic techniques to use the paradigm of statistical modeling efficiently and effectively for the purpose of lossless compression of continuous-tone images. Such a study on the computational aspect of lossless image coding is warranted because the problem has some unique operational difficulties [4], [9]. In this paper, we illustrate these difficulties, and provide practical solutions that were incorporated in the design of an efficient and effective context-based lossless image coding scheme—CALIC.

CALIC was designed in response to the ISO/IEC JTC 1/SC 29/WG 1 (JPEG) call soliciting proposals for a new international standard for lossless compression of continuous-tone images. In the initial evaluation of the nine proposals submitted at the JPEG meeting in Epernay, France, July 1995, CALIC had the lowest lossless bit rates in six of seven image classes: medical, aerial, prepress, scanned, video, and compound document, and the third lowest bit rate in the class of computer-generated images. CALIC gave an average lossless bit rate of 2.99 b/pixel on the 18 8-b test images selected by JPEG for proposal evaluation, compared with an average bit rate of 3.98 b/pixel for lossless JPEG[1] on the same set of test images. Even more encouragingly, CALIC obtained a 2% lower bit rate than the recent UCM (Universal Context Modeling) method developed by Weinberger *et al.* [9]. The latter is a principled but complex context-based image coding technique that is considered to be indicative of the lower bound on lossless bit rates achievable by other more practical methods.

Unlike UCM, CALIC is intended to be a practical lossless image codec. In order to attain this goal of practicality, new efficient algorithmic techniques have been developed for context formation, quantization, and modeling. Although conceptually more elaborate than many existing lossless image coders, CALIC is algorithmically quite simple, involving mostly integer arithmetic and simple logic. The prediction and modeling components of the encoder and decoder require only 1.52 CPU s on 512×512 continuous-tone images and 2.28 CPU s on 720×576 continuous-tone images when being executed on a SUN SPARC10 workstation. Both the encoding and decoding algorithms are suitable for parallel and pipelined hardware implementation while supporting sequential build-up. Furthermore, CALIC is symmetric, meaning that the encoder and decoder have similar time and space complexities. The binary executables of CALIC are available for common hardware platforms by anonymous ftp.[2]

This paper is structured as follows. In the next section, we give a schematic description of CALIC to provide an overview of the entire coding system. Then, in Sections III–VIII, we elaborate on the individual components of the CALIC coding system, and present the algorithms involved

Paper approved by M. R. Civanlar, the Editor for Image Processing of the IEEE Communications Society. Manuscript received February 20, 1996; revised July 8, 1996. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada.

X. Wu is with the Department of Computer Science, University of Western Ontario, London, Ont., N6A 5B7 Canada.

N. Memon is with the Department of Computer Science, Northern Illinois University, De Kalb, IL 60115 USA.

Publisher Item Identifier S 0090-6778(97)02784-0.

[1] Implementations for lossless JPEG on >8 b images are not publicly available.

[2] ftp://ftp.csd.uwo.edu/pub/from_wu.

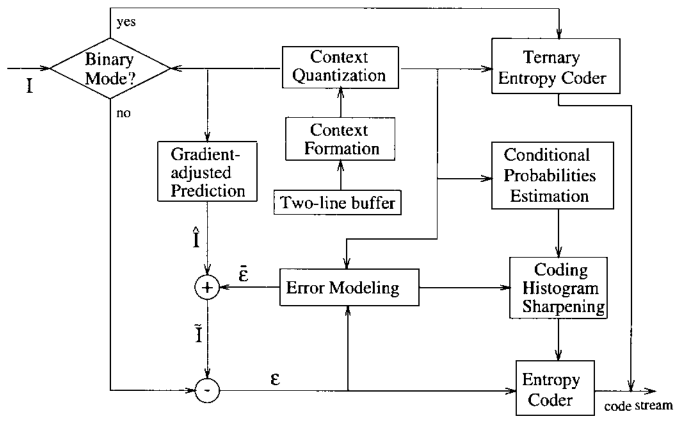

Fig. 1. Schematic description of CALIC's encoder.

and the rationales and design decisions behind the algorithms. Finally, in Sections IX and X, we present entropy results and actual bit rates that CALIC obtained on the ISO test images, and comparisons between CALIC and some popular existing lossless image coders.

II. System Overview

CALIC is a sequential coding scheme that encodes and decodes in raster scan order with a single pass through the image. The coding process uses prediction and context templates that involve only the two previous scan lines of coded pixels. Consequently, the encoding and decoding algorithms require a simple two-line buffer that holds the two rows of pixels that immediately precede the current pixel. A schematic description of CALIC's encoder is given in Fig. 1. Decoding is just the reverse process.

CALIC operates in two modes: binary and continuous tone. Binary mode is for situations in which the current locality of the input image has no more than two distinct intensity values, hence it is designed for a more general class of images than the traditional class of black and white images. The system selects one of the two modes on the fly during the coding process, depending on the context of the current pixel. Mode selection is automatic, and completely transparent to the user. In binary mode, a context-based adaptive ternary arithmetic coder is used to code three symbols, including an escape symbol which triggers an exit from binary mode.

In continuous-tone mode, the system has four major integrated components: gradient-adjusted prediction (GAP), context selection and quantization, context modeling of prediction errors, and entropy coding of prediction errors. First, the gradient of the intensity function at the current pixel I is estimated. A gradient-adjusted prediction $\hat{I}$ of I is made. A detailed description of GAP is presented in Section III. As illustrated by Fig. 1, the predicted value $\hat{I}$ is further adjusted via an error feedback loop of one-step delay. This results in an adaptive, context-based, nonlinear prediction $\tilde{I}$, as we will explain shortly.

The residue of the predictor $\tilde{I}$ is entropy-coded based on eight estimated conditional probabilities in eight different contexts. These eight contexts, called coding contexts, are formed by quantizing an error energy estimator Δ, a random variable, into eight bins. These quantizer bins partition prediction error terms into eight classes by the expected error magnitude. The described procedures in relation to the system are identified by the blocks labeled "Context Quantization" and "Conditional Probabilities Estimation" in Fig. 1. Details of this context quantization scheme in association with entropy coding are given in Section IV.

Performance of the GAP predictor can be significantly improved via context modeling. Hence, to further improve coding efficiency, we combine 144 texture contexts with four error energy contexts $(\Delta/2)$ to form a total of 576 compound contexts. Texture contexts are formed by a quantization of a local neighborhood of pixel values to a binary vector. The expectation of prediction error conditioned on each compound context is estimated. This estimate $\bar{\epsilon}$, which is the sample mean conditioned on the current compound context, is added to the GAP predictor $\hat{I}$ to correct its bias in the given compound context, generating an improved context-sensitive prediction $\tilde{I} = \hat{I} + \bar{\epsilon}$. In Fig. 1, these steps are conceptually related to the whole system by the blocks labeled "Context Quantization" and "Error Modeling" and the error feedback loop. In addition, the sign of $\bar{\epsilon}$ is utilized to sharpen the estimated conditional probabilities (hence reduce the underlying conditional entropies) that drive the entropy coder via a technique of sign flipping [10]. This corresponds to the block of "Coding Histogram Sharpening" in Fig. 1. The modeling of prediction errors conditioned on compound contexts is discussed in Section V.

The sign-flipped prediction error is then encoded using the eight coding contexts Δ described above. In the binary mode, we use a simple ternary adaptive arithmetic coder as described in Section VI. In the continuous-tone mode, we use an adaptive m-ary arithmetic coder. In order to improve coding and space efficiency, some preprocessing of errors is done prior to entropy coding. Details are given in Section VII.

III. GAP—Gradient-Adjusted Prediction

The simplest and also most common form of statistical redundancy in image data is the smoothness of the intensity function. A universal statistical model, while being able to learn this feature of smoothness, may require more parameters than a prediction scheme that utilizes *a priori* knowledge of image smoothness. Moreover, this learning process may take time, causing a delay in model adaptation. On the other hand, a simple and heuristic data fitting or prediction technique such as DPCM can decorrelate image data in smooth areas with very few fixed coefficients (no learning involved) and at very low computational cost. Therefore, prediction can be viewed as a context modeling technique of very low model cost that is highly effective under an assumption of smoothness. Hence, in CALIC, we chose to employ predictive coding in the continuous-tone mode.

The GAP predictor employed by CALIC is a simple, adaptive, nonlinear one that can adapt itself to the intensity gradients near the predicted pixel. Hence, it is more robust than traditional DPCM-like linear predictors, particularly in areas of strong edges. GAP differs from existing linear predictors in that it weights the neighboring pixels of $I[i,j]$ according to the estimated gradients of the image. For simple denotation

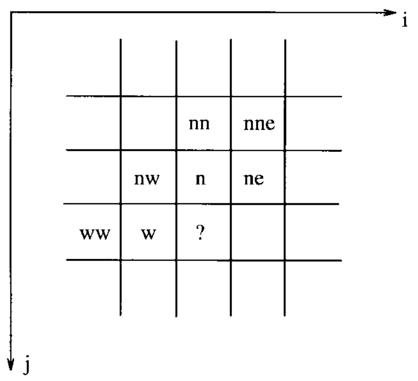

Fig. 2. Labeling of neighboring pixels used in prediction and modeling.

in the sequel, we let

$$
\begin{aligned}
&I_n = I[i,j-1],\quad I_w = I[i-1,j],\quad I_{ne} = I[i+1,j-1]\\
&I_{nw} = I[i-1,j-1],\quad I_{nn} = I[i,j-2],\quad I_{ww} = I[i-2,j]\\
&I_{nne} = I[i+1,j-2] \qquad (1)
\end{aligned}
$$

meaning north, west, northeast, northwest, north–north, west–west, and north–northeast, respectively. The locations of these pixels with respect to $I[i,j]$ are given in Fig. 2.

We estimate the gradient of the intensity function at the current pixel $I[i,j]$ by the following quantities[3]:

$$
\begin{aligned}
d_h &= |I_w - I_{ww}| + |I_n - I_{nw}| + |I_n - I_{ne}|\\
d_v &= |I_w - I_{nw}| + |I_n - I_{nn}| + |I_{ne} - I_{nne}|.
\end{aligned} \qquad (2)
$$

Clearly, d_h and d_v are estimates, within a scaling factor, of the gradients of the intensity function near pixel $I[i,j]$ in the horizontal and vertical directions. The values of d_h and d_v are used to detect the magnitude and orientation of edges in the input image, and make necessary adjustments in the prediction in order to get improved performance in the presence of local edges. Three absolute differences are used for d in each direction. This gave the best compression results in our experiments. Two or one absolute differences can be used here for lower complexity with a small loss in performance. The reason for using absolute rather than algebraic differences in (2) is to prevent cancellation of values of opposite signs. Efficient incremental and/or parallel schemes for evaluating d_h and d_v are straightforward. For instance, to avoid unnecessary repeated computations, one can store the values of $|I[\cdot,\cdot] - I[\cdot,\cdot]|$ associated with preceding pixels for future reference. This only requires an array of the size W, the width of the image.

[3]For the sake of clarity, we only consider sequential coding of interior pixels. Many possible treatments of boundary pixels are possible, and they do not make a significant difference in the final compression ratio.

Having computed d_h and d_v, a gradient-adjusted prediction (GAP) $\hat{I}[i,j]$ of $I[i,j]$ is made by the procedure, shown at the vottom of the page..

The predictor coefficients and thresholds given above were empirically chosen. A major criterion in choosing these coefficients is ease of computation. For instance, most coefficients are powers of 2 so that multiplications/divisions can be performed by shifting. The thresholds given above are for 8-b data, and are adapted on the fly for higher intensity resolution images as explained in Section VIII. It is possible, *albeit* quite expensive, to optimize the coefficients and thresholds for an image or a class of images, so that a norm of the expected prediction error $\{E||I-\hat{I}||\}$ is minimized. We do not recommend that such an optimization process be carried out on an image-by-image basis. However, it is important to point out that the coefficients and thresholds in computing $\hat{I}[i,j]$ can be set by a user who knows optimal or nearly optimal coefficients for target images.

IV. Coding Context Selection and Quantization

The prediction step does not completely remove the statistical redundancy in the image. It is observed that the variance of prediction errors $e = I - \hat{I}$ strongly correlates to the smoothness of the image around the predicted pixel $I[i,j]$. To model this correlation at a small computational cost, we define an error energy estimator

$$\Delta = ad_h + bd_v + c|e_w| \qquad (3)$$

where d_h and d_v are defined in (2) and $e_w = I[i-1,j] - \hat{I}[i-1,j]$ is the previous prediction error. $|e_w|$ is included in Δ because large errors tend to occur consecutively. One can, of course, include $e_n = I[i,j-1] - \hat{I}[i,j-2]$ in Δ. This makes Δ a slightly better error energy estimator, but requires buffering of an extra row of errors. We decided not to use e_n. Note that (3) has only two independent parameters since Δ can be scaled arbitrarily. The coefficients a, b, and c can be determined in an off-line design process by standard linear regression technique so that Δ is the least-squares estimator of the error strength $|e|$ based on $d_h, d_v, |e_w|$. For algorithm efficiency, we recommend $a = b = 1$ and $c = 2$ in (3).

By conditioning the error distribution on Δ, we can separate the prediction errors into classes of different variances. Thus, entropy coding of errors using estimated conditional probability $p(e|\Delta)$ improves coding efficiency over using $p(e)$. For time and space efficiency, we quantize Δ to L levels.

```
IF $(d_v - d_h > 80)$ {sharp horizontal edge} $\hat{I}[i,j] = I_w$
ELSE IF $(d_v - d_h < -80)$ {sharp vertical edge} $\hat{I}[i,j] = I_n$
ELSE {
    $\hat{I}[i,j] = (I_w + I_n)/2 + (I_{ne} - I_{nw})/4;$
    IF $(d_v - d_h > 32)$ {horizontal edge} $\hat{I}[i,j] = (\hat{I}[i,j] + I_w)/2$
    ELSE IF $(d_v - d_h > 8)$ {weak horizontal edge} $\hat{I}[i,j] = (3\hat{I}[i,j] + I_w)/4$
    ELSE IF $(d_v - d_h < -32)$ {vertical edge} $\hat{I}[i,j] = (\hat{I}[i,j] + I_n)/2$
    ELSE IF $(d_v - d_h < -8)$ {weak vertical edge} $\hat{I}[i,j] = (3\hat{I}[i,j] + I_n)/4$
}
```

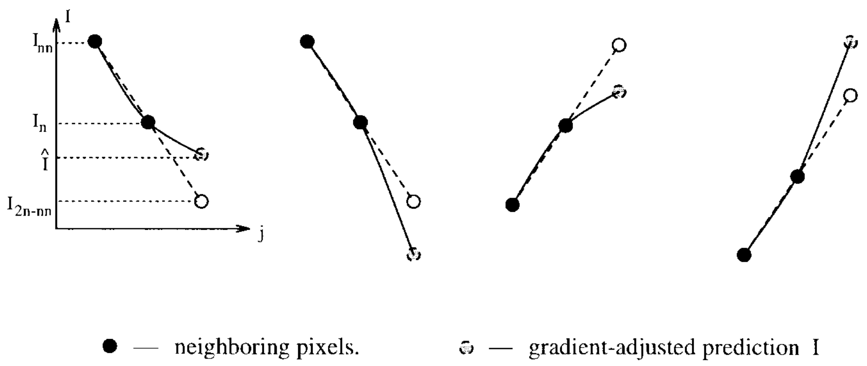

Fig. 3. Convexity relationship between GAP predictor $\hat{I}$ and the neighboring pixels.

In practice, $L = 8$ is found to be sufficient. Denote the Δ quantizer by Q, i.e., $Q\colon \Delta \to \{0, 1, \cdots, 7\}$. The quantization criterion is to minimize the conditional entropy of the errors. In an off-line design process, we get a training set of (e, Δ) pairs from test images, and use dynamic programming to choose $0 = q_0 < q_1 < \cdots < q_{L-1} < q_L = \infty$ to partition Δ into L intervals such that

$$-\sum_{e} p(e) \log p(e \mid q_d \le \Delta < q_{d+1}) \tag{4}$$

is minimized. We designed an optimal Δ quantizer using a training set consisting of all ISO test images and with $a = b = 1$ and $c = 2$ to minimize (4). This quantizer, whose bins are

$$\begin{aligned} &q_1 = 5, \quad q_2 = 15, \quad q_3 = 25, \quad q_4 = 42 \\ &q_5 = 60, \quad q_6 = 85, \quad q_7 = 140 \end{aligned} \tag{5}$$

worked almost as well as the optimal individual-image-dependent Δ quantizer. Quantizing the error energy estimator or the activity level in error modeling was used earlier by many authors, for example, in [1], [8]. But we optimized the quantizer under a conditional entropy criterion.

Estimating $L = 8$ conditional error probabilities $p(e|Q(\Delta))$ requires only a modest amount of memory during entropy coding. Furthermore, the small number of conditional error probabilities involved means that even small images will provide enough samples for context modeling to learn $p(e \mid Q(\Delta))$ quickly in adaptive entropy coding.

V. Context Modeling of Prediction Errors

Gradients alone cannot adequately characterize some of the more complex relationships between the predicted pixel $I[i, j]$ and its surroundings. Context modeling of the prediction error $e = I - \hat{I}$ can exploit higher order structures such as texture patterns and local activity in the image for further compression gains. However, the large number of possible contexts can lead to the sparse context or high model cost problem. In recent work [10], Wu has proposed a novel and effective solution to this problem. Instead of estimating $p(e|C)$ within each context C, he estimates only the conditional expectations $E\{e|C\}$ using the corresponding sample means $\bar{e}(C)$ within each context. These estimates are then used to further refine the prediction prior to entropy coding by an error feedback mechanism. In CALIC, we adopt this approach. Although the general principles that we have adopted are from [10], the specifics of applying these principles are different. In addition to refining some of the techniques in [10], we have also kept in mind the complexity of software and hardware implementations. We describe below the details by which we form and quantize contexts and arrive at an adaptive, context-based, nonlinear prediction.

A. Formation and Quantization of Contexts

In CALIC, contexts for error modeling are formed by combining 144 texture contexts with four error energy contexts by uniform quantization scheme $\Delta/2$ to form a total of 576 compound contexts. Texture context C is formed by a local neighborhood of pixel values

$$\begin{aligned} C &= \{x_0, \cdots, x_6, x_7\} \\ &= \{I_n, I_w, I_{nw}, I_{ne}, I_{nn}, I_{ww}, 2I_n - I_{nn}, 2I_w - I_{ww}\}. \end{aligned} \tag{6}$$

C is then quantized to an 8-b binary number $B = b_7 b_6 \cdots b_0$ using the prediction value $\hat{I}$ as the threshold, namely,

$$b_k = \begin{cases} 0, & \text{if } x_k \ge \hat{I}[i,j] \\ 1, & \text{if } x_k < \hat{I}[i,j] \end{cases} \qquad 0 \le k < K = 8. \tag{7}$$

Clearly, B captures the texture patterns in the modeling context which are indicative of the behavior of e. Also note that an event x_i in a modeling context need not be a neighboring pixel to $I[i, j]$. It can be a function of some neighboring pixels. x_6 and x_7, for example, represent the events whether the prediction value $\hat{I}[i, j]$ forms a convex or concave waveform with respect to the neighboring pixels in the vertical and horizontal directions, as depicted by Fig. 3. These convexity indicators are useful in the context modeling of e. The fact that $2I_n - I_{nn}$ and $2I_w - I_{ww}$ bring new information to the modeling context even though I_n, I_{nn}, I_w, and I_{ww} are already present is only because of the context quantization.

Since the variability of neighboring pixels also influences the error distribution, we combine the quantized error energy $0 \le \lfloor Q(\Delta)/2 \rfloor < L/2$ with the quantized texture pattern $0 \le B < 2^K$ to form compound modeling contexts, denoted by $C(\delta, \beta)$. This scheme can be viewed as a product quantization of two independently treated image features: spatial texture patterns and the energy of prediction errors.

B. Counting the Number of Contexts

At a glance, we would seemingly use $4 \cdot 2^8 = 1024$ different compound contexts since $0 \le \lfloor Q(\Delta)/2 \rfloor < L/2 = 4$ and

$0 \leq B < 2^K = 2^8$. However, not all 2^8 binary codewords of B quantizer defined by (7) are possible. If the prediction value $\hat{I}[i,j]$ lies between I_n and I_{nn}, then the condition $2I_n - I_{nn} < \hat{I}[i,j]$ is either always true or always false. In other words, the outcome of the test $2I_n - I_{nn} < \hat{I}[i,j]$ in (7) is uniquely determined if $I_n \geq \hat{I}[i,j]$ and $I_{nn} < \hat{I}[i,j]$, or if $I_n < \hat{I}[i,j]$ and $I_{nn} \geq \hat{I}[i,j]$. This should be apparent from Fig. 3. Also, by referring to (6) and (7), we see that in B, b_6 is fixed if b_2 and b_5 are different from each other. Likewise, b_7 is fixed if b_0 and b_4 are different from each other. Because b_6, b_2, and b_5 are not independent, they can only yield $2^3 - 2 = 6$ combinations. The same holds for b_7, b_0, and b_4. Thus, B represents only $6 \cdot 6 \cdot 4 = 144$ rather than 256 valid texture patterns. Therefore, the total number of valid compound contexts in error modeling is only $4 \cdot 144 = 576$.

C. Estimation of Error Magnitude Within a Context

Conditional expectations $E\{e|C(\delta,\beta)\}$ within each compound context are estimated by using the corresponding sample means $\bar{e}(\delta,\beta)$ for different compound contexts. Computing $\bar{e}(\delta,\beta)$ involves only accumulating the error terms in each compound context and maintaining a count on the occurrence of each context. For each compound context $C(\delta,\beta)$, $0 \leq \delta < 4$, $0 \leq \beta < 144$, we count the number $N(\delta,\beta)$ of occurrences of $C(\delta,\beta)$. We only use 1 byte to store the occurrence count. Whenever the count reaches some value $n_{\max} < 256$, we scale down the count, but leave the corresponding $\bar{\epsilon}(\delta,\beta)$ intact. We recommend $n_{\max} = 128$. In order to update $\bar{\epsilon}(\delta,\beta)$, we use an accumulator $S(\delta,\beta)$ of $\epsilon(\delta,\beta)$, and compute

$$\bar{\epsilon}(\delta,\beta) = S(\delta,\beta)/N(\delta,\beta) \tag{8}$$

whenever a compound context $C(\delta,\beta)$ occurs. We use a 16-b signed integer to store $S(\delta,\beta)$ whose range is more than sufficient in practice. It takes more than 128 consecutive $\epsilon(\delta,\beta) = -255$ or $\epsilon(\delta,\beta) = 255$ to violate the bounds $-2^{15} < S(\delta,\beta) < 2^{15}$, but we have $n_{\max} = 128$. Once $N(\delta,\beta)$ reaches 128, we rescale the variables by setting

$$S(\delta,\beta) = S(\delta,\beta)/2; \quad N(\delta,\beta) = 64. \tag{9}$$

The modeling system needs 1 byte for $N(\delta,\beta)$ and 2 bytes for $S(\delta,\beta)$ for each of 576 compound contexts. Thus, the total size of the memory required for context modeling of prediction errors is 1728 bytes. Besides being a technique to reduce the memory use, rescaling also has the widely recognized beneficial side effect of aging the observed data. It is an inexpensive way of adapting the context error model to time-varying sources. Indeed, the scaling technique slightly improved compression in our experiments.

D. Error Feedback and Sign Flipping

Since the conditional mean $\bar{e}(\delta,\beta)$ is the most likely prediction error in a given compound context $C(\delta,\beta)$, we correct the bias in prediction by a feedback $\bar{e}(\delta,\beta)$ to get $\tilde{I} = \hat{I} + \bar{e}(\delta,\beta)$. In order not to overadjust the GAP predictor, we actually consider the new prediction error $\epsilon = I - \tilde{I}$ rather than $e = I - \hat{I}$ in context-based error modeling described above. Context modeling of ϵ leads to an improved predictor for I: $\tilde{I} = \hat{I} + \bar{\epsilon}(\delta,\beta)$, where $\bar{\epsilon}(\delta,\beta)$ is the sample mean of ϵ conditioned on compound context $C(\delta,\beta)$. Conceptually, we ultimately have a two-stage adaptive prediction scheme via context modeling of errors and error feedback. A block diagram of the scheme is given in Fig. 4.

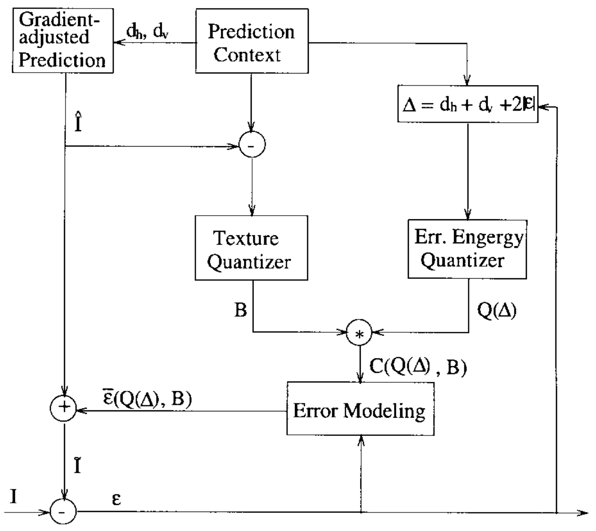

Fig. 4. Two-stage adaptive prediction scheme via context modeling of errors and error feedback.

Besides aiding in obtaining an improved prediction, the estimated prediction error $\bar{\epsilon}(\delta,\beta)$ within each context is also utilized to sharpen the estimated conditional probabilities (hence reduce the underlying conditional entropy) that drive the entropy coder via a novel technique of sign flipping as reported in [10]. This technique works as follows. Suppose the current context is $C(\delta,\beta)$. Before encoding $\epsilon = I - \tilde{I}$, the encoder checks whether $\bar{\epsilon}(\delta,\beta) < 0$. If yes, $-\epsilon$; otherwise, ϵ is encoded. Since the decoder also knows $C(\delta,\beta)$ and $\bar{\epsilon}(\delta,\beta)$, it can reverse the sign, if necessary, to reconstruct ϵ. Coding efficiency can be improved by the sign flipping technique because the error modeling based on compound contexts $C(\delta,\beta)$ captures some of the statistical redundancy in the sign of ϵ. In essence, by preserving or reversing the sign of ϵ, we make a prediction on the sign of ϵ.

VI. Binary Mode

The above adaptive, predictive coding scheme via context modeling of prediction errors is designed for compressing natural continuous-tone images. But predictive coding can have poor performance on images or subimages which contain very few widely separated gray levels compared with a coder based on explicit Markov modeling. This is because the image data no longer satisfy the assumption of smoothness for predictive coding. However, for a discrete type of images and subimages, direct context modeling of source symbols becomes feasible precisely because of the very small size of the symbol set. In this situation, it is more effective to code pixel values directly rather than prediction errors.

In order to effectively compress uniform or nearly uniform image areas, graphics, rasterized documents, and any combination of natural images with one or more of these types, we propose a binary coder as a separate module of the compression system. The system now has two modes: continuous tone and binary. Before $I[i,j]$ is to be coded, the algorithm first checks six neighboring pixels: I_{ww}, I_w, I_{nw}, I_n, I_{ne}, I_{nn}. If these six pixels have no more than two different values, the binary mode is implicitly and automatically triggered; otherwise, the system remains in the continuous-tone mode.

In the binary mode, let $s_1 = I_w$ and let the other value, if any, be s_2. The encoder describes $I[i,j]$ in one of the three states, using a ternary code T

$$T = \begin{cases} 0, & \text{if } I[i,j] = s_1 \\ 1, & \text{if } I[i,j] = s_2 \\ 2, & \text{otherwise.} \end{cases} \tag{10}$$

In the escape case of $T = 2$, the encoder switches from the binary mode to the continuous-tone mode. Encoding in the binary mode is conditioned on a context of six events:

$$\begin{aligned} C &= \{x_0, x_1, \cdots, x_5\} \\ &= \{I_w, I_n, I_{nw}, I_{ne}, I_{ww}, I_{nn}\} \end{aligned} \tag{11}$$

which are quantized to get a 6-b binary number $B = b_5 b_4 \cdots b_0$

$$b_k = \begin{cases} 0, & \text{if } x_k = s_1 \\ 1, & \text{if } x_k = s_2 \end{cases} \quad 0 \le k < 6. \tag{12}$$

As in the continous-tone mode, B represents the texture pattern around $I[i,j]$. Note that, because $s_1 = I_w$, $b_0 \equiv 0$ can be ignored from the context B. Thus, there are only $2^5 = 32$ different modeling contexts in the binary mode. Consequently, we use 32 conditional probabilities $p(T|B)$ to drive an adaptive arithmetic coder of three symbols. For the context modeling in the binary mode, only $32 \times 3 = 96$ frequency counts need to be maintained. Of course, we can use higher order contexts in binary mode like in JBIG for slightly higher compression. For compound-type images of large portions of binary texts, the JBIG's order-10 contexts can yield up to 2% more compression than our order-6 contexts. But for other images, high-order binary contexts can hurt compression because they make the condition for entering the binary mode more restrictive. To balance out the two conflicting situations and for easy implementation, we let the modeling context be the same as the template in which the binary mode is determined.

The incorporation of the binary mode into the compression system adds little to the system complexity, while greatly enhancing the coding performance on bilevel images, compound images, and images with large uniform areas. An important feature of our design is that the switch between continuous-tone and binary modes is context-dependent, automatic, and completely transparent to the user.

VII. Entropy Coding of Prediction Errors

An advantage of CALIC is the clean separation between context-based prediction and modeling of image data and entropy coding of prediction errors. Any entropy coder, be it Huffman or arithmetic, static or adaptive, binary or m-ary, can be easily interfaced with the CALIC system. In this section, we describe the entropy coding techniques used in the version of CALIC that was submitted to ISO as a candidate for standardization.

The entropy coder in the binary mode has an alphabet size of three. Given this small alphabet size, we use a simple ternary adaptive arithmetic coder. Coding is done under 32 different contexts, for each of which we maintain a frequency table consisting of three symbols. We use a 1 byte integer for each count and rescale counts when the cumulative frequency exceeds 2^{14}. We begin with a table in which all counts have been set to 1. For fast adaptation, we set the initial increment to 2^{13}, and this increment is halved every time the counts are rescaled. This halving process stops when the increment reaches a value of 1.

In the continuous-tone mode, we used an adaptive m-ary arithmetic coder, based on the CACM++ package that was developed and made publicly available by Carpinelli and Salamonsen.[4] The software is based on the work by Moffat, *et al.* [5]. In order to improve coding and space efficiency, the following techniques were employed to process the errors prior to entropy coding.

A. Remapping Errors

Although prediction errors can potentially take on 2^{z+1} possible values that range from $-2^z + 1$ to $+2^z - 1$, they can be mapped into the range 0 to $2^z - 1$ using the constraint that the value $\epsilon = I - \tilde{I}$ must fall into the interval $[-\tilde{I}, 2^z - 1 - \tilde{I}]$, where z is the number of bits in intensity resolution. If $\tilde{I} \le 2^{z-1}$, we rearrange the possible prediction errors $-\tilde{I}, \cdots, 0, 1, \cdots, \tilde{I}, \tilde{I}+1, \cdots, 2^z - 1 - \tilde{I}$ in the order $0, +1, -1, \cdots, +\tilde{I}, -\tilde{I}, \tilde{I}+1, \tilde{I}+2, \cdots, 2^z-1-\tilde{I}$. In symmetry to the case $\tilde{I} \le 2^{z-1}$, a similar mapping is performed for the case $\tilde{I} > 2^{z-1}$. Besides reducing the alphabet size for entropy coding, error remapping also orders error values in decreasing probability of occurrence. We use this monotonicity property to reduce the alphabet size prior to entropy coding.

B. Histogram Tail Truncation

Even after error remapping, an alphabet size of 2^z is still unnecessarily large. Large errors occur with diminishing frequency or not at all. But they still occupy spots in code space. For instance, for $\delta = 0$, over 99% of error population is within the range $[-8, 8]$ of the error histogram. If an error histogram of size 2^z is used, we have to set the frequency counts of all empty cells to 1 in practice to guard against the occurrence of an event of small probability, no matter how small. These forced counts of 1 in the error histograms can significantly distort the underlying error statistics, and hence reduce coding efficiency.

An *ad hoc* and yet effective remedy of the problem is to truncate the tails of the error histogram that is used to estimate $p(e|\delta)$, and use an escape mechanism to code the errors beyond the truncated code range, if they occur. Specifically, we limit the size of each conditional error histogram to some value N_δ, $1 \le \delta < L$, such that a large majority of errors to be coded under the coding context δ falls into the range of the N_δ largest entries of the δth error histogram. For a given coding context δ, a symbol (remapped prediction error) $x \ge N_\delta$ is encoded first to be $N_\delta - 1$, followed by the codeword for the new symbol

[4] ftp://ftp.cpsc.ucalgary.ca/pub/projects/arithmetic.coding.

$x - (N_\delta - 1)$, but being encoded under the coding context $\delta + 1$. We experimented with a large range of frequency table sizes N_δ. Too small a value of N_δ causes a large number of escape symbols to be coded, whereas a too large N_δ does not cure the zero frequency problem. A good balance between the two conflicting factors is required. We empirically set the frequency table sizes for the eight coding contexts to be

$$\begin{aligned} &N_0 = 18, \;\; N_1 = 26, \;\; N_2 = 34, \;\; N_3 = 50, \;\; N_4 = 66 \\ &N_5 = 82, \;\; N_6 = 114, \;\; N_7 = 256. \end{aligned} \tag{13}$$

Note that the user could be given the flexibility to set the N_δ values differently from the default values suggested.

C. Dynamic Bit Shifting

When the dynamic range of prediction errors is very large, the relatively small histogram sizes used in the proposed system can lead to a large number of escape symbols, resulting in poor coding efficiency. This is more likely to happen with high-resolution images (>10 b/pixel). Therefore, we track for each coding histogram the average of error magnitudes coded. If the average magnitude is significantly higher than $N_\delta/2$, then we decompose the remapped prediction error ϵ' into the most significant $z - k$ bits, ϵ'_1 and the least significant k bits, ϵ'_2. ϵ'_1 is then encoded in the normal manner, and ϵ'_2 can be coded using either a separate set of contexts or more simply sent as is. The value of k is selected such that the current average of ϵ'_1 is less than $N_\delta/2$.

The *ad hoc* technique given above does not add much to the overall complexity of the system. As we pointed earlier in the previous section, computing the average of error magnitudes is simple and fast. However, the gains obtained can be very significant for images that have a very large dynamic range of prediction errors. Also, since the bit decomposition is done adaptively, it leads to improved performance with many 8 bit images by more efficiently coding certain active areas of the image that incur large prediction errors.

VIII. Variable Image Resolutions

The parameters and thresholds of the predictors and the context quantizers were designed and optimized for images of 8-b resolution. For higher resolutions, we devised a very simple scaling mechanism for d_h and d_v that incurs a negligible loss of compression performance. Having scaled d_h and d_v, we proceed in a manner identical to 8-b resolution images. We do this as it is far more involved to adjust parameters and thresholds for the best compression performance for each intensity resolution.

Given an image with intensity resolution of z bits, $1 < z \leq 16$, the simplest scaling factor for d_h and d_v is $\lambda = 2^{8-z}$. But in our experiments, we found that this simple scaling factor can incur quite heavy loss in coding efficiency if the input images are very smooth relatively to the dynamic range. To solve this problem, we scale d_h and d_v according to the standard deviation σ of the errors (the average of error magnitudes) in the previous row. Computationally, the value of σ for the previous row is easy to update during the raster scan, requiring only an accumulator. Based on σ, we use an empirical formula

$$\lambda = 2^{-\lfloor \frac{z-8}{2} \rfloor - \max\{0, \lceil \log_2 \sigma - 5 \rceil\}} \tag{14}$$

to compute the scaling factor for d_h and d_v. Note that scaling requires only binary shifting. After d_h and d_v are scaled by λ, the same GAP predictor $\hat{I}$ as described in Section III, and the same error feedback mechanism to compute $\tilde{I}$, using the same coefficients in Δ and the same Δ quantizer as given in Section IV, are used to code images of more than 8-b intensity resolution.

IX. Compression Performance

In order to demonstrate the compression performance of CALIC, we compare it with some representative lossless image compression techniques. Table I lists compression results achieved with the set of ISO test images that were made available to the proposers. Column 2 lists the actual bit rates obtained by CALIC with the CACM++ implementation of arithmetic coding. Results with a static Huffman code are on an average 3.5% worse. Columns 3 and 4 list actual bit rates obtained by two publicly available lossless image compression codecs: the lossless JPEG implementation of Cornell University (LJPEG) and FELICS, a particularly simple lossless technique developed by Howard and Vitter [2]. The reported results of lossless JPEG were obtained by using the best of the eight JPEG predictors followed by two-pass Huffman coding of the prediction errors. FELICS uses context-based prediction and Rice–Golomb coding of prediction errors. LJPEG and FELICS results on images with more than 8-b per pixel are not given as public domain implementations of these two only handle 8-b/pixel image data. Also, note that the images are mostly color images and have more than one color band. Bit rates were obtained by appropriately averaging over the color bands weighted by size.

Both the current lossless JPEG and FELICS are rather simple techniques that require minimal memory and computation resources. CALIC, on the other hand, is more complex and does require more resources, although the increase in memory and computation resources is modest. Hence, we also compare CALIC with two other arithmetic-coding-based techniques, ALCM and CLARA, that were proposed to ISO by the University of California at Santa Cruz and Mitsubishi, respectively. Both had memory requirements similar to CALIC and were of comparable complexity. On 23 ISO test images, the average compression ratio for CALIC is 26% better than that of Huffman-coded lossless JPEG and 12% better than that of arithmetic-coded lossless JPEG.

X. Computational Complexity

CALIC was designed to be practical, and suitable for both software and hardware implementations. Compared with preceding context-based lossless image compression algorithms [1], [7], [9], CALIC is more efficient, and also simpler in algorithm structure. Even more significantly, the low complexity and algorithm simplicity are achieved with compression performance superior to all existing methods known to us. Since the entropy coding step is independent of source modeling in CALIC, and it is required by all lossless image codecs, in

TABLE I
BIT RATES (BITS PER PIXEL) OF CALC ON ISO TEST SET AND COMPARISON WITH A FEW SELECTED SCHEMES

Image	CALC	JPEG	FELICS	ALCM	CLARA
air2	3.83	4.90	4.49	4.08	4.11
bike	3.50	4.33	4.06	3.69	3.63
cafe	4.69	5.63	5.31	4.99	4.86
tools	4.95	5.69	5.42	5.17	5.06
woman	4.05	4.84	4.58	4.30	4.15
cats	2.51	3.69	3.30	2.67	2.57
water	1.74	2.62	2.36	1.82	1.84
chart	1.28	2.23	2.14	1.27	1.36
graphic	2.26	2.81	2.85	2.41	2.24
faxballs	0.75	1.50	1.74	0.60	0.82
hotel	3.71	4.22	4.20	3.92	3.88
gold	3.83	4.22	4.21	4.02	3.88
finger	5.47	5.85	6.11	5.94	5.46
us	2.34	3.63	3.32	2.32	2.41
cmpnd2	1.24	2.50	2.40	1.34	1.47
cmpnd1	1.24	2.51	2.40	1.29	1.53
bike3	4.23	5.15	4.67	4.43	4.48
chart_s	2.66	3.86	3.44	2.77	2.88
mri	5.73	-	-	6.17	5.73
cr	5.17	-	-	5.43	5.22
xray	5.83	-	-	6.24	5.99
ct	3.63	-	-	4.09	4.08
aerial1	8.31	-	-	8.77	8.36

TABLE II
COUNTS OF DIFFERENT OPERATIONS PER PIXEL INVOLVED IN CONTEXT ERROR MODELING AND CONTEXT-BASED ADAPTIVE PREDICTION

Operations	ADD	MUL/DIV	shift	comparison	ABS
Step 1	6	0	0	0	2
Step 2	<5	0	<2	< 6	0
Step 3	2	0	3	11	1
Step 4	2	1	0	0	0
Total	13	< 1	<5	< 17	3

what follows, we will analyze the time complexity only of the context-based prediction and modeling components of the CALIC system, which determines the difference in execution time between different algorithms.

In CALIC, four major steps are taken by the both encoder and decoder toward context error modeling and context-based adaptive prediction.

Step 1: Computing d_v, d_h as in (2).
Step 2: Computing GAP $\hat{I}$ as in Section III.
Step 3: Context quantization as in (3) and (7).
Step 4: Computing the context-based, adaptive prediction $\tilde{I}$ as in Section VIII.

The arithmetic and logic operations per pixel involved in these steps are counted and tabulated in Table II.

Note that the complexity analysis above only applies to the continuous-tone mode. The binary mode is much simpler. Steps 1 and 3, the two most expensive ones, can be easily executed in parallel and by simple hardware, as self-evident in (2) and (7). For instance, d_v and d_h can be computed simultaneously by two identical circuitries, cutting the computation time in half; in context quantization, all b_k's in a texture pattern of (7) in the continuous-tone mode, or of (12) in the binary mode, can be set simultaneously by a simple circuitry. Because the modeling and prediction algorithms do not require the support of complex data structures (only one-dimensional arrays are used), their hardware implementations should not be difficult. The space complexity of CALIC was already discussed at the end of Section V-C.

REFERENCES

[1] P. Howard and J. S. Vitter, "Error modeling for hierarchical lossless image compression," in *Proc. Data Compression Conf.*, J. A. Storer and M. C. Cohn, Eds.,1992, pp. 269–278.

[2] ——, "Fast and efficient lossless image compression," in *Proc. Data Compression Conf.*, J. A. Storer and M. C. Cohn, Eds., 1993, pp. 351–360.

[3] G. G. Langdon, A. Gulati, and E. Seiler, "On the JPEG model for lossless image compression," in *Proc. Data Compression Conf.*, J. A. Storer and M. C. Cohn, Eds., 1992, pp. 172–180.

[4] N. D. Memon, K. Sayood, and S. S. Magliveras, "Lossless image compression with a codebook of block scans," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 24–31, Jan. 1995.

[5] A. Moffat, R. Neal, and I. Witten, "Arithmetic coding revisited," in *Proc. Data Compression Conf.*, J. A. Storer and M. C. Cohn, Eds., 1995, pp. 202–211.

[6] J. J. Rissanen, "Universal coding, information, prediction and estimation," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 629–636, 1984.

[7] M. J. Slyz and D. L. Neuhoff, "A nonlinear VQ-based predictive lossless image coder," in *Proc. Data Compression Conf.*, J. A. Storer and M. C. Cohn, Eds., 1994, pp. 304–310.

[8] S. Todd, G. G. Langdon, and J. J. Rissanen, "Parameter reduction and context selection for compression of gray scale images," *IBM J. Res. Develop.*, vol. 29, no. 2, pp. 188–193, 1985.

[9] M. J. Weinberger, J. J. Rissanen, and R. B. Arps, "Applications of universal context modeling to lossless compression of gray-scale images," *IEEE Trans. Image Processing*, vol. 5, pp. 575–586, Apr. 1996.

[10] X. Wu, "Lossless compression of continuous-tone images via context selection, quantization, and modeling," *IEEE Trans. Image Processing*, vol. 6, May 1997.

Xiaolin Wu (SM'96) was born in 1958 in Chengdu, Sichuan, China. He received the B.Sc. degree from Wuhan Cehui Technical University, China, in 1982, and Ph.D. degree from the University of Calgary, Calgary, Alta., Canada, in 1988, both in computer science.

He is currently an Associate Professor with the Department of Computer Science, the University of Western Ontario, London, Ont., Canada. His main research interests are visual computing and communications, source coding, and algorithms. He has published frequently in IEEE, ACM, and other international journals on various topics in image coding, computer graphics, quantization theory and algorithms, and image processing. He actively participated in and contributed to the recent development of the new JPEG standard for lossless image coding.

Dr. Wu served as a program committee member of the IEEE Data Compression Conferences in 1994 and 1995.

Nasir Memon (S'91–M'92) received the B.E. degree in chemical engineering and the M.Sc. degree in mathematics from the Birla Institute of Technology, Pilani, India, in 1981 and 1982 respectively. He received his M.S. and Ph.D. degrees from the University of Nebraska, Lincoln, both in Computer Science, in 1989 and 1992 respectively.

He was an Assistant Professor in the Department of Computer Science, Mathematics and Physics at Arkansas State University from 1992 to July 1994. He joined the Computer Science Department at Northern Illinois University, De Kalb, in August 1994, where he is currently an Assistant Professor. His research interests include data compression, data encryption and communication networks.