Image Codec by Noncausal Prediction, Residual Mean Removal, and Cascaded VQ

Amir Asif and José M. F. Moura, *Fellow, IEEE*

***Abstract*—We describe a technique for still image compression that combines i) noncausal optimal recursive prediction, ii) residual quadtree mean removal, and iii) a modification of cascaded vector quantization. We refer to this image codec as noncausal prediction with residual mean removal, and cascaded vector quantization (NRQ/CVQ). We provide examples that illustrate the performance of NRQ/CVQ up to compression ratios of 42.5 : 1. We show that NRQ/CVQ outperforms alternative algorithms that we tested including the conventional causal prediction differential pulse code modulation (DPCM) with quadtree mean removal cascaded vector quantization and the Joint Photographic Experts Group (JPEG) baseline standard algorithm.**

I. Introduction

DIGITAL representation of images allows visual information to be easily manipulated on a computer, stored on a compact disc, or communicated over a digital channel. These advantages, however, are at the expense of the large volume of data required to represent raw digital images. Images contain a high degree of spatial redundancy, with the pixel values being highly correlated. Image compression exploits this redundancy to reduce the number of bits required to represent the image, providing significant savings in resources for storage and transmission.

Compression image techniques may be divided into two classes: lossless (or noiseless) and lossy (or noisy) compression. Lossless compression allows perfect reconstruction of the original image from the compressed image. Perfect reconstruction is ideal since no information is lost but it limits the achievable compression ratio. In contrast, lossy compression provides higher levels of compression, with a trade-off between compression ratio and reconstruction error. Lossy compression usually consists of three basic components:

- Decomposition or transformation of the original image to find an equivalent representation which, for example, has a reduced dynamic range or leads to an otherwise more concise description of the image.
- Quantization to reduce the bit rate.
- Lossless coding to achieve rates close to the entropy of the quantized symbol source.

Reconstruction of the image from the coded data stream consists of applying the inverse of each of the operations in reverse order.

Manuscript received January 20, 1995; revised October 31, 1995. This paper was recommended by Associate Editor C. A. Gonzales. This work was supported in part by a grant of Bellcore through INI.

The authors are with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213-3890 USA.

Publisher Item Identifier S 1051-8215(96)01323-7.

This paper describes a lossy compression scheme that we refer to as noncausal codec with residual quadtree mean removal and cascaded vector quantization (NRQ/CVQ). The method uses noncausal prediction with residual quadtree mean removal and cascaded vector quantization (QCVQ). Noncausal prediction has also been used by Gray *et al.* [1], in which they subsample a vector, code the downsampled vector, and then predict the remaining samples. They call the scheme a "Checkerboard Tile." In our case, noncausal prediction is based on modeling the image as a Gauss Markov random field (GMRF) and is used to generate an error image considerably less correlated than the original image. QCVQ is then used to compress the error image. QCVQ is composed of two steps. The first step removes the means of each block by a quadtree technique applied to the error image that was obtained in the noncausal prediction stage. On the second step of QCVQ, the residuals are vector quantized by a multiple stage vector quantizer—cascaded VQ. In our implementation of QCVQ, we introduce in between the stages of the vector quantizer a selector that discards from further stages of quantization those residuals that have been adequately quantized by the previous stages of quantization. QCVQ offers several advantages over conventional vector quantization, namely, lower storage requirements for storing and transmitting the codebooks, reduced search, hence, lesser complexity, and faster speed for the generation of the codebook and quantization. We stress that even though QCVQ is explained in terms of NRQ/CVQ and of the error image resulting from the noncausal prediction step, the ideas are fairly general and applicable to different types of vector quantizers.

The paper is structured in eight sections. Section II reviews the noncausal prediction algorithm based on Gauss Markov random fields (GMRF) which is used to generate the error image. Section III summarizes the standard algorithm for vector quantization and discusses some of the associated limitations. Section IV introduces quadtree mean removal, while Section V describes a modification to cascaded vector quantization. Section VI combines quadtree mean removal and cascaded vector quantization. Section VII designs in detail the full codec and compares its performance with that of other competing algorithms, namely other versions of vector quantization alone, conventional predictive coding, and transform coding. Finally Section VIII concludes the paper.

In comparing the reconstructed picture quality, we make use of the mean square error (MSE) and peak signal to reconstruction noise ratio (PSNR) which for an $(N \times N)$ 8-bit

image are defined as follows:

$$\text{MSE} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}(x(i,j) - \hat{x}(i,j))^2 \tag{1}$$

$$\text{PSNR} = 10\log_{10}\left(\frac{255^2}{\text{MSE}}\right) \tag{2}$$

where the original image is denoted by $x(i,j)$ and the reconstructed image by $\hat{x}(i,j)$. A higher PSNR does not necessarily imply a higher quality reconstructed image, because the perceived picture quality is highly dependent upon the human visual system (HVS). To enable subjective comparison of the different techniques, we include in the paper examples of images as reconstructed by the several methods considered.

II. Noncausal Prediction

In the following sections, a new approach to image compression based on Gauss Markov random fields (GMRF), [2], [3], is discussed. The approach resembles differential pulse code modulation (DPCM), [4], with one major difference—the prediction is based on pixels from along all directions, rather than pixels on one side of the pixel of concern. Before discussing the main algorithm, the required background is briefly presented. We assume that the field is zero mean.

In noncausal prediction, we use a noncausal neighborhood of pixels to make a linear prediction of the current pixel value. Fig. 1 illustrates the noncausal neighborhood sets of order up to six for the pixel labeled 'o'. The first order neighborhood set involves pixels marked '1', the second order set involves the pixels marked '1' and '2,' and similarly for the higher order sets. For a first order prediction model, the prediction of the image intensity value at pixel (i,j) (say the pixel represented by 'o' in Fig. 1) given the image intensity values at all other pixels is, [5]

$$\hat{x}(i,j) = \beta_v x(i-1,j) + \beta_h x(i,j-1) + \beta_h x(i,j+1) + \beta_v x(i+1,j) \tag{3}$$

where β_v and β_h are the vertical and the horizontal field interactions, $x(i-1,j)$ represents the intensity value of the pixel in the row above the reference pixel, row $i-1$, and in the same column as the reference pixel, column j, and likewise for the remaining quantities in (3). The error $e(i,j)$ involved in the prediction is given by

$$e(i,j) = x(i,j) - \hat{x}(i,j). \tag{4}$$

For simplicity, consider the image to be square, say $N \times N$. Following standard procedure, we collect the intensity values for the pixels of row i into the $N \times 1$ random vector $\vec{X}_i = [x_{i1} \quad x_{i2} \quad \cdots \quad x_{iN}]^T$, and stack the vectors corresponding to all rows of the image $i = 1, \cdots, N$ to form the $N^2 \times 1$ vector $\vec{X} = [\vec{X}_1^T \quad \vec{X}_2^T \quad \cdots \quad \vec{X}_N^T]^T$. Equation (4), after substituting for $\hat{x}(i,j)$ as given by (3), is expressed in a matrix form by stacking the field values $x(i,j)$, and the input noise $e(i,j)$, into $N^2 \times 1$ column vectors $\vec{X}$ and $\vec{e}$, respectively. Taking into consideration appropriate boundary conditions, we obtain

$$A\vec{X} = \vec{e}. \tag{5}$$

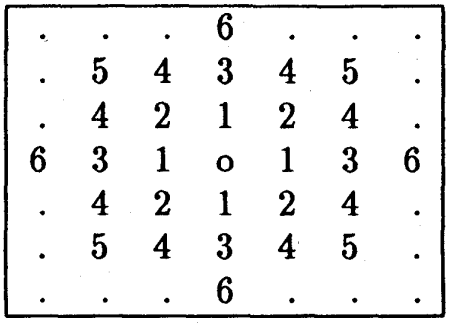


Fig. 1. First to sixth order neighborhood noncausal prediction models.

The matrix A is a $N^2 \times N^2$ matrix containing the field interactions β_v and β_h and is referred to as the potential matrix or potential operator. For first order fields, A has the following structure

$$A = \begin{bmatrix} \boldsymbol{B}_1 & \boldsymbol{C}_1 & 0 & 0 & 0 & \cdot & \cdot \\ \boldsymbol{C}_1^T & \boldsymbol{B} & \boldsymbol{C} & 0 & 0 & \cdot & \cdot \\ 0 & \boldsymbol{C}^T & \boldsymbol{B} & \boldsymbol{C} & 0 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 0 & 0 & \boldsymbol{C}^T & \boldsymbol{B} & \boldsymbol{C}_1 \\ \cdot & \cdot & 0 & 0 & 0 & \boldsymbol{C}_1^T & \boldsymbol{B}_1 \end{bmatrix} \tag{6}$$

where $\boldsymbol{B}_1, \boldsymbol{C}_1, \boldsymbol{B}$, and $\boldsymbol{C}$ are $N \times N$ blocks. The blocks $\boldsymbol{B}_1$ and $\boldsymbol{C}_1$ are slight modifications of blocks $\boldsymbol{B}$ and $\boldsymbol{C}$ accounting for the boundary conditions, and $\boldsymbol{B}$ and $\boldsymbol{C}$ have the special structure

$$\boldsymbol{B} = \begin{bmatrix} 1-\beta_h & -\beta_h & 0 & 0 & 0 & \cdot & \cdot \\ -\beta_h & 1 & -\beta_h & 0 & 0 & \cdot & \cdot \\ 0 & -\beta_h & 1 & -\beta_h & 0 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 0 & 0 & -\beta_h & 1 & -\beta_h \\ \cdot & \cdot & 0 & 0 & 0 & -\beta_h & 1-\beta_h \end{bmatrix}$$

and

$$\boldsymbol{C} = -\beta_v I_N \tag{7}$$

where I_N is the identity matrix of order N. The parameters β_v and β_h satisfy a constraint such that A is symmetric and positive definite.

For the experiments in Section VII, we use an asymmetric Neumann boundary condition [2], i.e., we assume that the off lattice neighbors of a boundary pixel have the same value as the boundary pixel. Then, the blocks $\boldsymbol{B}_1$ and $\boldsymbol{C}_1$ are

$$\boldsymbol{B}_1 = \boldsymbol{B} + \boldsymbol{C} \quad \text{and} \quad \boldsymbol{C}_1 = \boldsymbol{C}. \tag{8}$$

The structure of the potential matrix A includes both past (left and up) and future (right and down) pixels for prediction of a pixel value. Such a representation precludes recursive computations as Kalman–Bucy filtering. We derive one-sided-backward representations by upper Cholesky factorization of A

$$A = U^T U. \tag{9}$$

The matrix U is an upper triangular $N^2 \times N^2$ sparse block matrix

$$U = \begin{bmatrix} \boldsymbol{U}_1 & \boldsymbol{\Theta}_1 & 0 & 0 & 0 & \cdot & \\ 0 & \boldsymbol{U}_2 & \boldsymbol{\Theta}_2 & 0 & 0 & \cdot & \cdot \\ 0 & 0 & \boldsymbol{U}_3 & \boldsymbol{\Theta}_3 & 0 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 0 & 0 & 0 & \boldsymbol{U}_{N-1} & \boldsymbol{\Theta}_{N-1} \\ \cdot & \cdot & 0 & 0 & 0 & 0 & \boldsymbol{U}_N \end{bmatrix} \tag{10}$$

where $\boldsymbol{U}_i$ and $\boldsymbol{\Theta}_i$ are $N \times N$ upper and lower triangular blocks, respectively. Substituting in (5), and left multiplying throughout by U^{-T}, we get

$$\boldsymbol{U}_i \vec{X}_i + \boldsymbol{\Theta}_i \vec{X}_{i+1} = \vec{w}_i \quad \text{for} \quad 1 \le i \le N-1 \tag{11}$$

$$\boldsymbol{U}_N \vec{X}_N = \vec{w}_N \tag{12}$$

where $\vec{w} = U^{-T}\vec{e}$. The vector $\vec{w} = [\vec{w}_1^T \quad \vec{w}_2^T \quad \cdots \quad \vec{w}_N^T]^T$ is white; see [2].

The backward regressors, $\boldsymbol{U}_i$ and $\boldsymbol{\Theta}_i$, required for generating the error signal $\vec{w}$, are evaluated by solving a Riccati type equation that follows by equating the diagonal block entries of (9)

$$S_1 = \boldsymbol{B}_1 \tag{13}$$

$$S_2 = \boldsymbol{B} - \boldsymbol{C}_1^T \boldsymbol{B}_1^{-1} \boldsymbol{C}_1 \tag{14}$$

$$S_i = \boldsymbol{B} - \boldsymbol{C}^T S_{i-1}^{-1} \boldsymbol{C} \quad \text{for} \quad 3 \le i \le N-1 \tag{15}$$

$$S_N = J\boldsymbol{B}_1 J - \boldsymbol{C}_1^T S_{N-1}^T \boldsymbol{C}_1 \tag{16}$$

where

$$S_i = \boldsymbol{U}_i^T \boldsymbol{U}_i \quad \text{and} \quad J = \begin{bmatrix} & & & & 1 \\ & & & \cdot & \\ & & 0 & & \\ & \cdot & & & \\ 1 & & & & \end{bmatrix}. \tag{17}$$

The regressor block coefficient $\boldsymbol{\Theta}_i$ follows by equating the off-diagonal entries of the matrices in (9), for example,

$$\boldsymbol{U}_i^T \boldsymbol{\Theta}_i = C \quad \text{for} \quad 2 \le i \le N-2. \tag{18}$$

With real data fields and in actual applications, evaluation of the regressors $\boldsymbol{U}_i$ and $\boldsymbol{\Theta}_i$ for all $1 \le i \le N$ is not required. In fact, the regressors converge asymptotically to a steady state solution at a geometric rate, see [3]; hence, only a few of the regressors need to be computed. For a first order field, the steady state solution is given by

$$S_\infty = \boldsymbol{B}/2 + \sqrt{(\boldsymbol{B}/2)^2 - \beta_v^2 I} \tag{19}$$

$$\boldsymbol{U}_\infty^T \boldsymbol{U}_\infty = S_\infty \quad \text{and} \quad \boldsymbol{\Theta}_\infty = \boldsymbol{U}_\infty^{-T} C. \tag{20}$$

To save on the computational complexity, the regressors $\{\boldsymbol{U}\}_{i=1}^N$ and $\{\boldsymbol{\Theta}\}_{i=1}^N$ were approximated by their respective steady state values $\{\boldsymbol{U}\}_\infty$ and $\{\boldsymbol{\Theta}\}_\infty$. See [3] for details. To use the above model, we need estimates of β_v and β_h.

The evaluation of β_v and β_h is based on maximizing the likelihood function; see [3]. We may use instead the following approximate parameter estimates for β_v and β_h

$$\beta_v = \frac{\xi \chi_v}{|\chi_h| + |\chi_v|} \tag{21}$$

$$\beta_h = \frac{\xi \chi_h}{|\chi_h| + |\chi_v|} \tag{22}$$

where χ_h and χ_v are the horizontal and vertical sample correlations given by

$$\chi_v = \frac{1}{N^2} \sum_{i=1}^{N-1} \sum_{j=1}^{N} x(i,j)x(i+1,j) \tag{23}$$

$$\chi_h = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N-1} x(i,j)x(i,j+1) \tag{24}$$

and ξ is a positive tolerance which for an $N \times N$ image is bounded by

$$\xi < \frac{1}{2\cos\dfrac{\pi}{N+1}}. \tag{25}$$

Summarizing the above results, Fig. 2 shows how to compress and reconstruct an image based on noncausal prediction. Prior to VQ, the compression procedure requires three stages to generate the uncorrelated error field. The main steps involved are:

1) *Parameter Estimation:* After subtracting the global mean, the horizontal field interaction β_h, the vertical field interaction β_v, and the driving noise power σ^2 are estimated for the zero mean image. These parameters are needed for reconstruction and are transmitted to the receiver as overhead.
2) *Backward Representation:* Based on the value of the horizontal and the vertical field interactions, the regressors ($\boldsymbol{U}_i$ and $\boldsymbol{\Theta}_i$) are computed using (14)–(18) or the asymptotic counterparts using (19)–(20).
3) *Whitening:* Using the backward representation, the uncorrelated error image field $\vec{w}$ is evaluated from (11)–(12).

To achieve high compression rates, the error image $\vec{w}$ is vector quantized, the details are discussed in the next section. The image is reconstructed by reversing the steps of the encoder as shown in Fig. 2.

III. Vector Quantization

We briefly discuss the basic idea of vector quantization (VQ) (Fig. 3) for the sake of completeness and to introduce notation. The $N \times N$ image, W, to be vector quantized is partitioned into contiguous, nonoverlapping, square $n \times n$ subblocks. Each $n \times n$ subblock is then rearranged in lexicographic order to form $n^2 \times 1$ vectors W_i. The number of input vectors W_i is $K = N^2/n^2$. The encoder computes the distortion $d(W_i, Y_j)$ between the input vector W_i and each code vector $Y_j, j = 1, 2, \cdots, p$, from a codebook C with p codevectors. The optimum encoding rule is the nearest neighbor rule, in which the index J is transmitted to the decoder if code vector

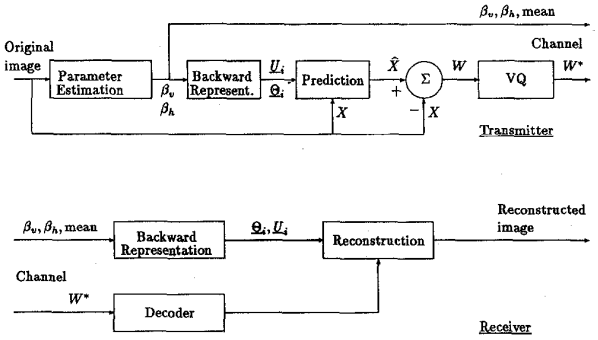


Fig. 2. Noncausal DPCM system.

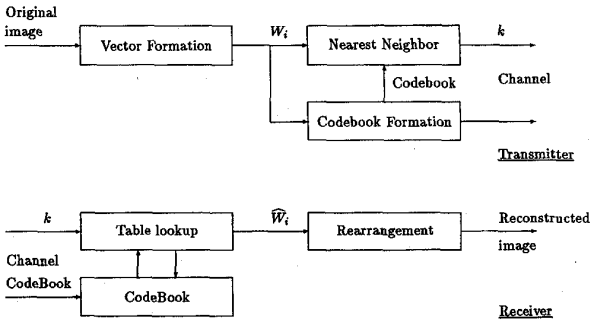


Fig. 3. Vector quantizer.

Y_J yields the least distortion. The decoder simply looks up the Jth code vector Y_J from a copy of the codebook C and the output $\hat{W}_i$ is Y_J. The decoded vectors are then rearranged to form the reconstructed image.

An optimal vector quantizer is one which uses a codebook C^* that yields the least average distortion D^* for encoding the input vectors W_i among all possible codebooks. The design algorithm for an optimal codebook is difficult. However, a clustering algorithm, commonly referred to as the LBG algorithm or generalized Lloyd algorithm is normally used to generate the codebook. The algorithm employs a large set of input vectors derived from a training sequence, selects or designs an initial codebook and then optimizes it. See [6] or [7] for details.

VQ for image compression has two important limitations. The first is its computational complexity which grows exponentially as the size of the code vector or the size of the image increases. The second is degradation especially at high compression ratios due to the low signal to quantization error ratio (SQR). We describe quadtree cascaded VQ (QCVQ) designed to improve on both of these problems. To improve SQR, we introduce a mean removal technique based on quadtrees. Computational complexity is reduced by implementing VQ in stages—cascaded VQ. We discuss these two issues separately in the two subsequent sections.

IV. Quadtree Mean Removal

The basic idea of block mean removal (see Baker and Gray [8]) is to divide the image in blocks, to compute the mean of each block (arranged as a vector), and subtract the mean of each block from the intensity of each pixel in the block. The resultant vector is referred to as the residual vector. The residual vectors and the means are coded using separate codebooks. Thus, we decompose the original vector into separate features, a mean representing the general background level and a residual vector representing the departure of the vector about its mean.

Most regions of the image consist of highly homogeneous information. Rather than dividing the image into equal size blocks, it is preferable to segment the image into homogeneous blocks of varying sizes and shapes and then transmit their means. This requires additional bits to describe the size, shape, and location of each region. It represents a trade-off between

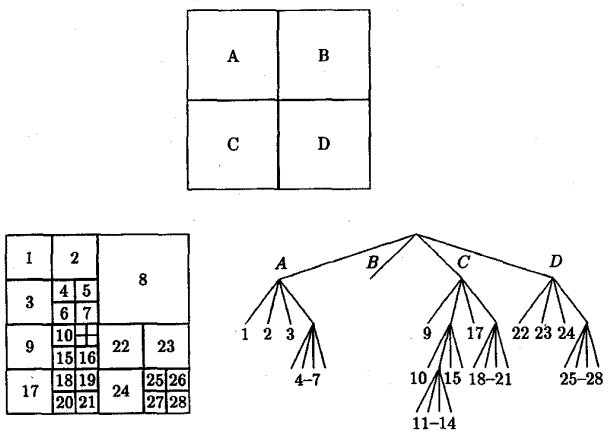


Fig. 4. Quadtree partition.

better compression due to more homogeneous regions versus the additional overhead required for the description of each individual block. Quadtrees represent a good compromise because the partition is readily specified with a modest overhead in bits per pixel. A variable size partition based on a quadtree is illustrated in Fig. 4.

We apply the quadtree mean removal scheme discussed above, not to the original image, but to the error image that results from the initial noncausal prediction step. This is an important distinction of our work.

A. Quadtree Mean Removal Algorithm

The initial step in the quad tree mean removal is to select the starting block size. As first step, we partition the error image into four equal sized blocks, say of 128 × 128 pixels. The mean of each 128 × 128 block is computed, quantized, and subtracted from each pixel in the block to form the residual error image. At this stage, a decision is made whether to subdivide each block further. For a given block, if the decision is positive, the block is again divided into four subblocks corresponding to the generation of four children from the appropriate node. Means are again evaluated, quantized, and subtracted from each block. The procedure is repeated now with each of the new subblocks until either the decision turns out to be negative or we reach the lowest block size which, in our experiments in Section VII, we choose to be 4 × 4.

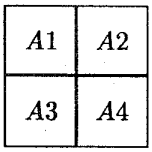


Fig. 5. Order of traversal.

The decision to further subdivide the block is based on the variance of the block. We start by looking at four subblocks of the original error image and determine if they are inactive, by comparing the subblock variance with a preselected threshold value. If inactive, that is, if the subblock variance is less than the threshold, we remove the large block mean; otherwise, we segment the subblock into four equal parts, yielding the four children of the subblock and continue with the same technique. As stated earlier, the procedure is repeated until either the 4 $\times$ 4 subblock size is reached or the variance drops below the threshold value.

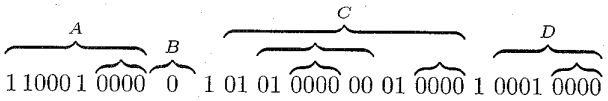
In our procedure, we set the threshold as a fraction, γ, of the overall variance of the error image. We have experimented successfully with values of γ between 0.25 and 0.50. A higher value of the threshold implies larger blocks or smaller number of children, whereas a lower value implies more block subdivisions.

The above technique fine tunes the size of each block, reducing the overhead involved in the transmission of the means. It gives rise to a new problem—the need to transmit to the receiver adequate information about the quadtree structure so that the means are reintroduced at the appropriate pixels. The following subsection addresses this issue.

Quadtree Representation: A number of methods have been proposed in the literature to represent the quadtree structure, see, for example, Samet [9], that uses a pointer-based representation. Since in our application we are concerned only with the segmentation of the error image, we need not pursue this complex pointer-based representation. As an alternative, we use a fairly simple pointerless representation (see [10]) made of the following two rules.

1) *Block Subdivision Coding:* A '0' is transmitted if there is no further subdivision of the block (or subblocks). A '1' is transmitted in case of further subdivision of the block.
2) *Block Scanning:* Each subdivided block is always traversed in the following order: North–West (NW), North–East (NE), South–West (SW) and South–East (SE), or, more specifically, referring to block A in Fig. 5, the order of traversing will be $A1, A2, A3$, and $A4$.

Based on the above approach, the quadtree representation of Fig. 4 is



The quadtree subdivision is represented by a string of 1's and 0's. This string may then be coded using a lossless coding technique such as arithmetic coding to further reduce the overhead involved with quadtree representation. We have not included this in our results.

In summary, we described quadtree mean removal as an efficient procedure for block mean subtraction. The residual error image consists of blocks of different sizes, which are approximately homogeneous. They may be coded using a variable block size VQ. Variable block size VQ raises a further complication, namely, that separate codebooks containing code vectors of unequal dimensions need to be constructed for different block sizes. The optimal allocation of codebook sizes for a given codebook size m is image dependent and extremely difficult to implement. In addition, as we move to higher compression rates, the number of code vectors per codebook decreases and, as such, the image quality deteriorates. We circumvent this problem by using code vectors of the same dimension as explained in the next section.

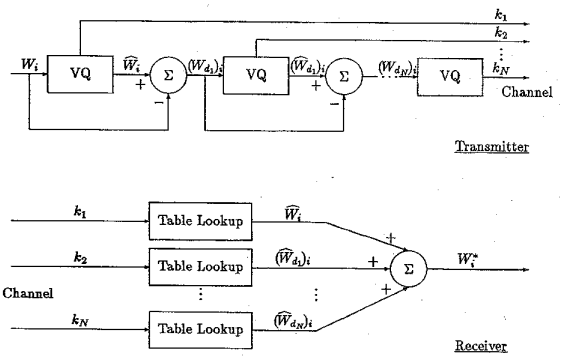


Fig. 6. Cascaded VQ.

V. Cascaded VQ

The mean removed error image, referred to as the residual error image, is next vector quantized. The VQ method that we implement is a modification of cascaded or multistage VQ [11] shown in Fig. 6.

Before presenting the approach, it is worthwhile to comment on some of the main features of cascaded VQ. The basic idea behind cascaded VQ is to perform VQ in stages. The first stage performs VQ of the input vectors $W_i, i = 1, \cdots, K$, leading to $\hat{W}_i = Y_J$ for some codeword Y_J. The second stage vector quantizes the difference vectors $(W_{d_1})_i = W_i - \hat{W}_i$ which are generated by subtracting from the original vectors W_i their respective vector quantized approximations $\hat{W}_i$ of the first VQ stage. If necessary, additional stages are added. Stage $j \geq 2$ quantizes the difference vectors $(W_{d_{j-1}})_i$. The size of the codebook of the last stage is usually taken larger than the size of the previous stages. The reason is that even if the earlier stages perform a relatively crude VQ of the input vectors, the last stage will improve on the accuracy.

The receiver used for reconstructing the error image consists of a number of lookup tables. Each table decodes a corresponding stage of the cascaded VQ algorithm. The output is used to reconstruct the original error image.

Cascaded VQ offers several advantages over single stage VQ. At each stage of cascaded VQ, the codebooks are much

smaller than in single stage VQ. Since the generation of a codebook is basically a search in the space of possible code vectors, the design of smaller codebooks involves a reduced search, therefore, the time and effort required are significantly decreased. In addition, smaller codebooks require less overhead for transmission and storage. In general, this simplicity of standard cascaded VQ comes at the expense of PSNR. The overall PSNR for the cascaded VQ scheme just described is in general smaller than the PSNR of the optimized single stage VQ for the same compression ratio. This is because the codebook that results is, in general, not the optimal one. With cascaded VQ, one optimizes locally at each stage, so the algorithm is greedy. Further improvement in PSNR may be obtained by the use of optimally designed cascaded VQ codebooks as presented in [12].

Selector: In our experiments, we use a modified version of cascaded VQ. The modification is the inclusion of a selector in between consecutive stages of the cascaded VQ. We explain the role of the selector with reference to Fig. 6. Start with the first stage of cascaded VQ and generate the difference vectors $(W_{d_1})_i = W_i - \hat{W}_i$ which act as the input to the second stage. The purpose of the selector is to drop those difference vectors that correspond to vectors which have been represented satisfactorily by the first stage. Only the remaining difference vectors are vector quantized by the second stage. This procedure is now repeated at each subsequent stage, until all of the vectors have been dropped or the permissible number of stages has been exhausted.

To decide if a vector has been satisfactorily coded, we have a criterion which is based upon the average absolute distortion. The average distortion d_{av} introduced by vector quantizing all of the vectors in the first stage is evaluated as $d_{\text{av}} = \Sigma |W_i - \hat{W}_i|/K$ where $K = N^2/n^2$ is as before the total number of input vectors. We then define a threshold λ_1 to be in between 30% to 75% of the average distortion d_{av}. This threshold is used to select the vectors that will be kept for the second stage. The distortion incurred in vector quantizing a vector at the first stage is compared to the threshold. If it falls below the threshold, the vector is considered satisfactorily represented by the first stage and its difference vector $(W_{d_1})_i$ dropped from further consideration. Otherwise, the difference vector $(W_{d_1})_i$ is selected and passed to the second stage. The selected difference vectors $(W_{d_1})_i, i = 1, \cdots, K_2$ from the first stage form the training set for the generation of the codebook at the second stage. The training set for the second stage will be much smaller than that of the first stage, i.e., $K_2 < K$. It consists of only those vectors which are not *adequately* represented by the first stage. A new codebook is generated at the second stage, the difference vectors encoded, the differences generated and passed on to the selector, and the process continued. The threshold λ_2 at the input of the second stage selector can be taken to be λ_1 or made adaptive by repeating the distortion computation.

The introduction of the selector serves two purposes. First, it prevents additional error for those vectors which are represented adequately by earlier stages. Second, it attempts to spread the quantization error uniformly over the entire error image. By keeping for subsequent stages of additional quantization only the vectors with a higher degree of distortion, the selector enables allocating more bits to these data vectors, with the overall goal of reducing their final level of distortion before transmission. The net result of the selector is to improve significantly the overall PSNR of cascaded VQ. The introduction of the selector enables us to reduce the complexity involved in single stage VQ without having to pay a high price of reduced PSNR. The inclusion of the selector requires the additional overhead of one bit per difference vector. This overhead is needed to code the status of the difference vector, i.e., whether it is dropped or not. On the other hand, the selector drops vectors which are represented adequately by the previous stage. The threshold value of the selector is chosen in such a way that the net overhead needed for coding the status information is compensated by this savings.

Barnes and Frost [12] study the optimal design of cascaded VQ (referred to as direct sum VQ). They show that for cascaded VQ, optimal encoding can be implemented by a sequential encoder. However, the complexity of such an encoder exceeds that of an exhaustive search of the direct sum codebooks. For computational simplicity, we use the suboptimal scheme discussed in this section.

VI. Quadtree Cascaded VQ Scheme

Based on the discussion of the two previous sections, the proposed VQ scheme, referred to as quadtree cascaded VQ (QCVQ), is illustrated in Fig. 7 and consists of the following major steps.

1) The error image which is the output of the noncausal prediction step, see Section II, is first preprocessed to remove the mean. Since most regions of the error image consist of highly homogeneous information, instead of removing means from each (4 × 4) block (constituting a vector), a variable block size mean removal technique is used. In particular, we use quadtrees to find a fast segmentation algorithm. The quadtree technique, as explained before, leads to variable block size removal. These means are scalar quantized and are transmitted to the receiver along with the block segmentation information. Although we have not implemented this in our experimental studies, we can also apply further compression to the block means, either lossy compression like DPCM, or, lossless compression like arithmetic or entropy coding, thus further reducing the associated overhead.
2) The mean removed error image is now vector quantized using cascaded VQ. Data vectors are formed by dividing the mean removed error image in 4 × 4 blocks. The generation of the codebook at each stage uses the LBG algorithm [6]. LBG is highly sensitive to the choice of the initial codebook. To generate the initial codebook, we used the method of Splitting [6]. The mean removed error image is quantized using the final codebook and transmitted to the receiver. The difference between the mean removed error image and the quantized mean removed error image, referred to as the difference error image, is passed to the next stage of VQ.

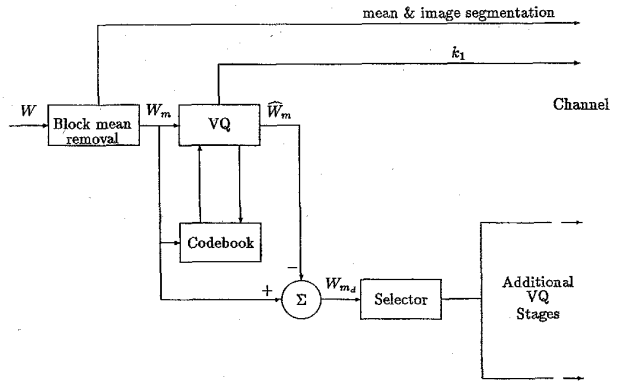


Fig. 7. Quadtree cascaded VQ.

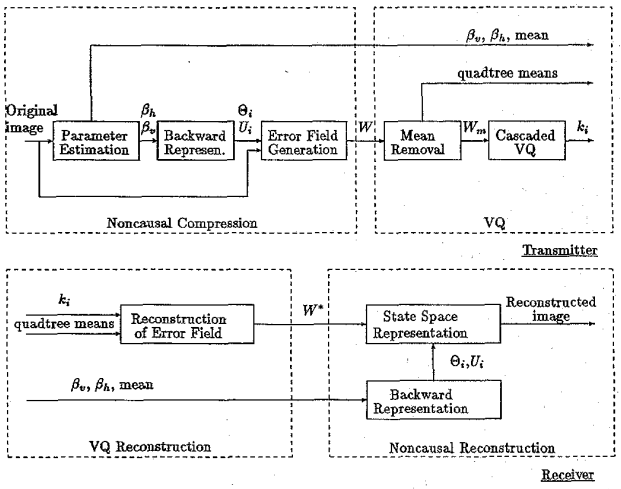


Fig. 8. NRQ/CVQ compression system.

3) The next stage is a replica of the first except for the up-front inclusion of a selector. The difference error image is fed to the selector. The selector retains only those vectors which have comparatively higher distortion and passes them onto the second stage. Vectors which have been represented adequately by the first stage are dropped. The difference vectors that survive the selector form the training set for the codebook at the second stage. Using the newly generated codebook, the selected difference vectors are vector quantized and, again, the difference between the original difference vectors and their quantized versions is evaluated. The VQ in stages is repeated, until all the vectors have been dropped or the permissible number of stages has been exhausted. The introduction of the selector offers several advantages: improved PSNR for the same compression rate; less storage for storing the codebook; reduced search time, hence, lesser complexity for generation of the codebook; and faster speed for quantization of the vectors.

A final remark on cascaded VQ is that it offers an opportunity for a variable rate coding system. At each stage, we measure the actual distortion incurred in encoding the input vectors. If this distortion exceeds the desired level, additional VQ stages are added. This ensures good picture quality for highly detailed error images, while offering a higher compression ratio for the simpler ones.

VII. NRQ/CVQ

A. Algorithm

The noncausal predictor, the quadtree mean removal, and the cascaded VQ blocks described in Sections II–VI are integrated into the compression scheme represented in Fig. 8. We refer to this scheme as noncausal codec with residual quadtree cascaded VQ (NRQ/CVQ). We discuss its main features with reference to Fig. 8.

1) *Noncausal Prediction:* The first block of the codec is the noncausal prediction algorithm of [2], which is discussed in Section II. It is used to remove the spatial redundancy from the image by subtracting the predicted pixel intensity values from the actual pixel intensities. The output of this block is the uncorrelated error image W. The field correlations (β_v, β_h) and the global mean, which are needed for reconstruction of the image at the receiver, represent transmission overhead.
2) *Quadtree mean removal:* The block following the noncausal prediction is the quadtree mean removal algorithm of Section IV. It is applied to the uncorrelated error image W. The residual error image W_m is passed to the cascaded VQ. The quadtree means are transmitted to the receiver. Although not implemented in our experiments described below, the quadtree means may be compressed by DPCM coding, scalar quantization, and entropy coding.
3) *Cascaded VQ:* The residual error image W_m is vector quantized by the cascaded VQ algorithm of Section V. We have found that two stages are usually a good compromise between simplicity of implementation and performance. In between the stages, we use a selector as described in Section V. By entropy coding, the quantization output can be compressed further. Our experimental results in Subsection VII-B do not include this lossless compression stage.
4) *VQ decoder:* At the decoder, a replica W^* of the uncorrelated error image is reconstructed in the first block by VQ decoding and by reintroducing the block means at the appropriate locations.
5) *Reconstruction:* The second block of the decoder uses the field interactions (β_v, β_h) to construct the backward unilateral equivalent representative model (Θ_i, U_i) to regenerate the image from the reconstructed error image W^*. Finally, the global mean is reintroduced.

We compare briefly our NRQ/CVQ with the approach in Vaisey and Gersho (V & G) [13]. Vaisey and Gersho use a quadtree mean removal algorithm to divide the *original* image into square blocks of various sizes. The decision to further subdivide a block is made on the basis of the absolute value of the *means*. A tolerance value λ_m is selected at each level.

Further subdivision of a given block stops if the absolute value of the means of the blocks resulting from the next subdivision are within the tolerance λ_m of the mean of the present block. The mean removed image is then vector quantized using a variable block-sized VQ.

In contrast, NRQ/CVQ vector-quantizes the *error* image after the noncausal prediction block, i.e., it quantizes the error image W formed by subtracting the noncausally predicted values of the pixels from their actual values. The dynamic range of this error image, and so the dynamic range of the block means, is much smaller than the corresponding dynamic range in V & G. This in itself significantly improves the performance of the coupled quadtree mean removal with cascaded VQ. Because of this reduced dynamic range, using a *mean* block decision in our quadtree as in V & G would cause very little subdivision to occur. We base instead the quadtree subdivision on the variances of the blocks which do not depend on the corresponding block means. This provides a very sensitive indicator of the block details, hence, of when to subdivide or not. Quadtree subdivision in NRQ/CVQ proceeds then in much the same way as in V & G, except that decisions to terminate or to go further into the tree are always based on the block variances.

The second major difference between NRQ/CVQ and V & G lies in the vector quantization of the error image after the block mean removal, referred to as the residual image. There are several issues here. Vaisey and Gersho use a variable block size VQ to vector quantize the residual image. Before quantization, the blocks are grouped according to their size. Within each of these groups, the vectors representing the pixels of the blocks are arranged in lexicographic order. For example, we may have groups of 4^2-dimensional vectors all the way up to 32^2-dimensional vectors (32 being the maximum block size allowed in V & G). Separate codebooks are then developed for each of these groups of different dimensions.

In contrast with V & G, in NRQ/CVQ, the VQ step is decoupled from the quadtree mean removal step. The residual error image is subdivided in blocks of equal size. Only a single codebook of constant dimension is developed at each stage. There is no need for the additional overhead of transmitting this information to the receiver.

A second issue concerns the complexity associated with the design of large dimension codebooks as they may arise for example in V & G. In V & G, the codebook dimensions are as large as the largest block size resulting from the quadtree subdivision, which is typically 32 × 32. For our cascaded VQ, we subdivide the residual error image in 4 × 4 blocks, regardless of the dimension of the largest block size resulting from the quadtree step (which in an extreme case could be the whole error image). In addition, by having several cascaded stages, we further reduce the size of the largest codebook. For example, for a total budget of six bits per 4 × 4 quantization blocks, a single stage VQ leads to a codebook of dimension 64, while the more typical two stage cascaded VQ, with say two bits assigned to the first stage and four bits to the second stage, leads to a four vector codebook for the first stage and a sixteen vector codebook for the second stage. These are much simpler to implement than the codebook of dimension 64.

A third issue is still concerned with the dimension of the largest block in the quadtree. Because we have decoupled Quadtree mean removal from the cascaded VQ, the blocks in the Quadtree mean removal stage that we use can be as large as it takes. This is not the case in V & G where they constrain the maximum quadtree block size to a relatively small number (32 × 32) resulting in considerable more subdivisions in their quadtree algorithm than may actually be required.

B. Experimental Study

In this subsection, we compare the bit rate/quality performance of NRQ/CVQ with that of alternative coding/decoding algorithms. We carry out qualitative and quantitative studies. The qualitative analysis is based on the subjective evaluation of the image reconstructed by different compression schemes. This provides a feel for the subjective quality achieved by each scheme. The second comparison study is based on a quantitative measure, namely the PSNR (or equivalently MSE) as defined in (2). The use of PSNR does not always correlate well with the perceived image quality. Nevertheless, it is commonly used by other authors in evaluating compression algorithms. For this reason, we also include PSNR comparisons.

We compare NRQ/CVQ with the following compression techniques: baseline JPEG; quadtree cascaded VQ (QCVQ); causal prediction (DPCM) with quadtree cascaded VQ (DPCM/QCVQ); and noncausal prediction with conventional VQ.

JPEG: The JPEG algorithm is the baseline algorithm standardized by the Joint Photographic Experts Group, see [14]. It is the current technology benchmark. For reproducibility of the results, we use the Stanford implementation of Hung [15], available through anonymous ftp from havefun.stanford.edu. By adjusting the value of the Q-factor, different compression ratios may be obtained. In our studies with JPEG, we include the final stage of lossless entropy coding, which, we stress, is not present in any of the other compression schemes. If entropy coding is included in our scheme, we should expect a further reduction in the reported bit per pixel (bpp) rate of at least 10 to 20%.

Quadtree Cascaded VQ (QCVQ): The test images are compressed using quadtree mean removal cascaded VQ. Starting, for example, from a 256 × 256 image, the quadtree mean removal algorithm, as described in Section IV, is initiated on each 128 × 128 subblock. The lowest block size, at which the subdivision is finally stopped, is 4 × 4. The residual image generated after subtracting the quadtree means is then vector quantized using the modified cascaded VQ scheme explained in Section V. The VQ is trained on sets derived from the residual image and VQ is performed on each 4 × 4 block. QCVQ is like our codec NRQ/CVQ except that there is no noncausal prediction step. In other words, the quadtree mean removal and cascaded VQ are applied directly to the original image, not to the error image, the output by the noncausal predictor.

Causal Prediction (DPCM) with QCVQ: DPCM is a widely used causal predictive coding scheme. We include our tests with this algorithm in order to evaluate to what

TABLE I
PARAMETERS FOR CAUSAL AND NONCAUSAL MODELS

Image	Causal Model			Noncausal Model		Mean
	β_h^c	β_v^c	β_d^c	β_h	β_v	
Lenna	0.863320	0.666224	-0.556740	0.114133	0.383855	94.2464

degree the *noncausal* image models deliver better results in comparison with the causal ones. In DPCM, the prediction is subtracted from the actual pixel intensity to form an error image, referred to as DPCM differential image. We use as DPCM model a third order Markov Mesh. The differential error image is then vector quantized using a QCVQ stage. In other words, DPCM with QCVQ is NRQ/CVQ with the noncausal predictor step substituted by the DPCM predictor.

Noncausal Prediction with Conventional VQ: This scheme is the noncausal predictive step in NRQ/CVQ followed with a conventional single stage VQ. It does not include residual quadtree mean removal.

Noncausal Prediction with Residual Quadtree Cascaded VQ (NRQ/CVQ): This is the algorithm described in the paper. It couples noncausal predictor with QCVQ.

The parameters for the causal and noncausal prediction model for each of the images are shown in Table I. These are obtained using the techniques described in [3].

Subjective Evaluation: We first compare the five schemes by showing the images reconstructed by using each of the five techniques listed above. It should be noted that limitations in the printing process can hide details that are apparent using a higher resolution display medium. Figs. 9 to 11 illustrate the visual comparison among the five compression schemes, discussed earlier. We display the results with Lenna contained in the USC database.

Figs. 9 and 10 show the reconstructed Lenna at 0.375 bpp which corresponds to a compression ratio of 21.33. Fig. 9 is the only comparison we show of NRQ/CVQ with noncausal prediction with conventional VQ and with quadtree mean removal cascaded VQ (QCVQ). It clearly shows that the combination of both noncausal prediction with quadtree mean removal cascaded VQ as done with NRQ/CVQ, Fig. 9(d), is better than noncausal prediction with conventional VQ, Fig. 9(b), or QCVQ alone, Fig. 9(c). Accordingly, our subsequent studies are only concerned with comparing NRQ/CVQ with DPCM/QCVQ and JPEG.

Fig. 10 illustrates the results of the JPEG, DPCM/QCVQ, and NRQ/CVQ at 0.375 bpp, again a compression ratio of 21.33. A similar comparison with Lenna at 0.188 bpp (a compression ratio of 42.55) is performed in Fig. 11. In Fig. 11, the test image is a square of 256 $\times$ 256 pixels, instead of the 128 $\times$ 128 pixels as used earlier.

Before continuing on with the comparison, we indicate how we compute the compression ratio, in particular how we get 0.188 bpp. In NRQ/CVQ, we use a two stage cascaded VQ to compress the error image. The codebooks used at the two stages are of size 2 and 4 code vectors, respectively. Hence three bits are needed to encode the vector corresponding to each 4 $\times$ 4 subblock of the mean removed error image. If all vectors are kept in both stages, i.e., no selector is used, this implies a bit rate of 3/16 or 0.1875 bpp. However, this raw bit rate is modified by the actions of the quadtree mean removal and of the selector. The number of blocks resulting from Quadtree subdivision is 521. A three-bit scalar quantizer is used to encode the mean of each block. The amount of overhead for the transmission of the means is then

$$\frac{521 \times 3}{256 \times 256} = 0.022 \text{ bpp}. \tag{26}$$

This overhead can be reduced further by using some coding scheme like DPCM to code the bit stream representing the block means, therefore, it is not included in evaluating the overall bit rate. Note that in our comparison studies we also exclude a similar overhead in causal DPCM with QCVQ or the overhead associated with JPEG. The overhead required for the selector is evaluated as follows. The selector transmits one additional bit to code the status of each difference vector, i.e., to code whether the difference vector is dropped or retained. Dividing the 256 $\times$ 256 error image in 4 $\times$ 4 subblocks for VQ results in 4096 vectors. Hence, 4096 bits are needed as overhead after the first stage. However, by setting the selector threshold value to 75% of the average distortion, only half of the difference vectors are retained for the final stage. This means savings of two bits per each vector discarded or a total of 2 $\times$ 2048 bits, which equals the status overhead. Based on this analysis, the bit rate for Lenna works out to be 0.1875 bpp.

Figs. 10 and 11 illustrate the improvement in the visual quality with NRQ/CVQ when compared with JPEG and DPCM/QCVQ. This is not surprising since clearly JPEG and DPCM/QCVQ are not designed to operate at these compression ratios. JPEG exhibits blocking or tiling throughout the image, edges are distorted, and fine details appear to be lost. In Fig. 11(b) for example, sharp edges of the cap, hair, etc., are not maintained, fine details such as the eyelashes and the lines of the hat are blurred. Important features such as the nose and the lips are difficult to distinguish from their respective surroundings.

DPCM/QCVQ (see Fig. 11(c)) performs a better job than JPEG on the edges and finer details. Still, it exhibits other forms of distortions, in particular, vertical streaking is observed in the image. As apparent in Fig. 11(c), this streaking is more apparent in bright intensity areas, such as the cap and the face, resulting in a poor perceived quality of the image.

There is little evidence of any such distortion in Fig. 11(d), the image compressed with NRQ/CVQ. Edge integrity is maintained and the monotone areas are reproduced with all the smooth variations remaining intact. Only the fine details such as the eyelashes and the lines on the hat are affected, these too with comparatively little degradation.

Implementation Issues: We discuss briefly the computational effort and codebook design associated with NRQ/CVQ. We will then propose alternative implementations that significantly improve the cost effectiveness of the NRQ/CVQ implementation and provide test results on this practical implementation.

Computational effort: The direct implementation of the noncausal prediction step as described in Section II involves a maximum likelihood estimation procedure and an iterative Riccati equation that are computationally intensive steps. In

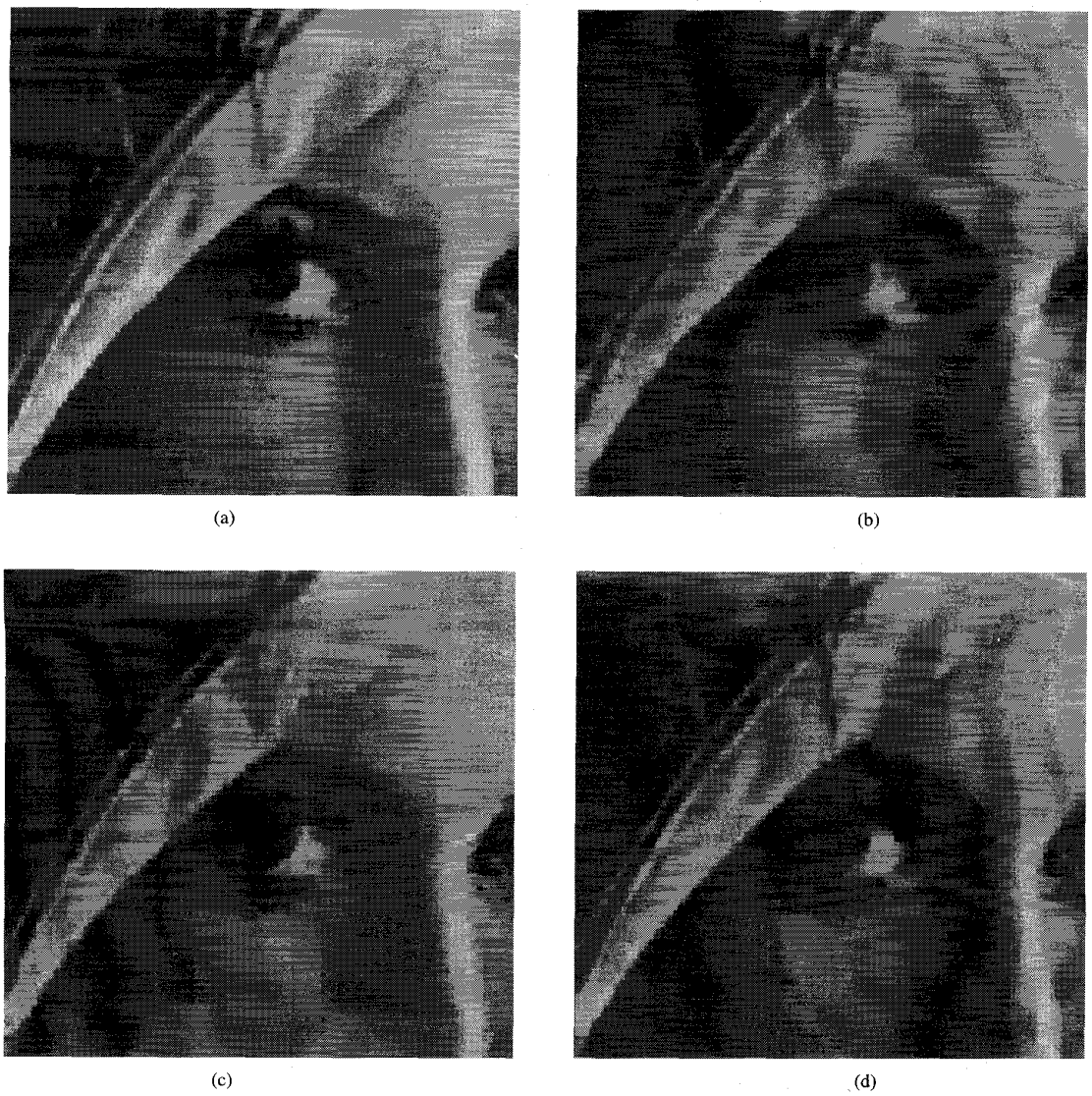


Fig. 9. (a) Original 128 × 128 Lenna face, (b)–(d) comparison of reconstructed Lenna face after compression to 0.3750 bpp (b) using noncausal prediction with conventional VQ (c) using Quadtree mean removal cascaded VQ (d) and using standard NRQ/CVQ.

alternative, a more practical implementation uses the steady state solution to the Riccati equation, see equations (19) and (20), and the approximate parameter estimates given by equations (21) and (22). We refer to this implementation as the practical NRQ/CVQ.

Local Codebook: The results displayed in Figs. 9 to 11 using some form of VQ use vector quantizers trained with local codebooks, i.e., codebooks designed using a single error image derived from the test image.

Local codebooks add to the computational effort and to the overhead associated with image codecs. While these factors are a major deterrent to using standard VQ, cascaded VQ presents significant improvement on both grounds. For example, at 0.188 bpp, with a two stage cascaded VQ, we need to design two codebooks, the first one with two codewords and the second one with four codewords, which reduces significantly the associated computational effort. In terms of overhead, six codewords need to be transmitted to the receiver. For a 256 × 256 image divided in 4 × 4 subblocks this computes to

$$\frac{16 \times 8 \times 6}{256 \times 256} = 0.012 \text{ bpp}$$

where we use eight bits to quantize each entry in the codevector. Adding to this the overhead of the block means, see (26), amounts to less than 18% of the coding budget. This overhead compares favorably with the overhead associated with JPEG and is on the order of the additional compression gain provided by a final stage of lossless compression.

We now present results with the practical version of NRQ/CVQ and using a universal codebook. Fig. 12 compares the 256 × 256 Lenna image at 0.188 bpp, again a compression ratio of 42.55, using JPEG, and NRQ/CVQ. We provide results for two different implementations of NRQ/CVQ: the

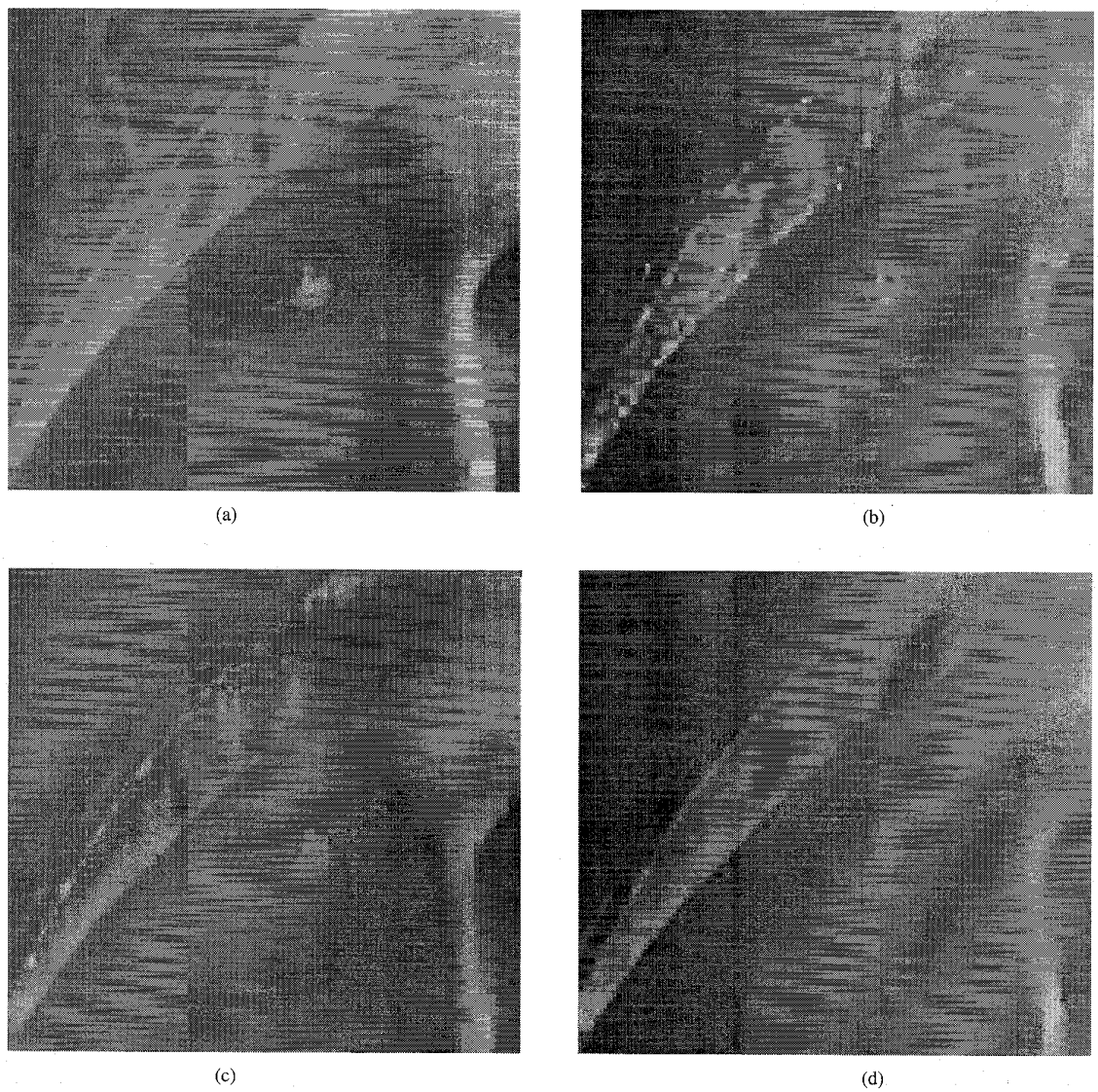
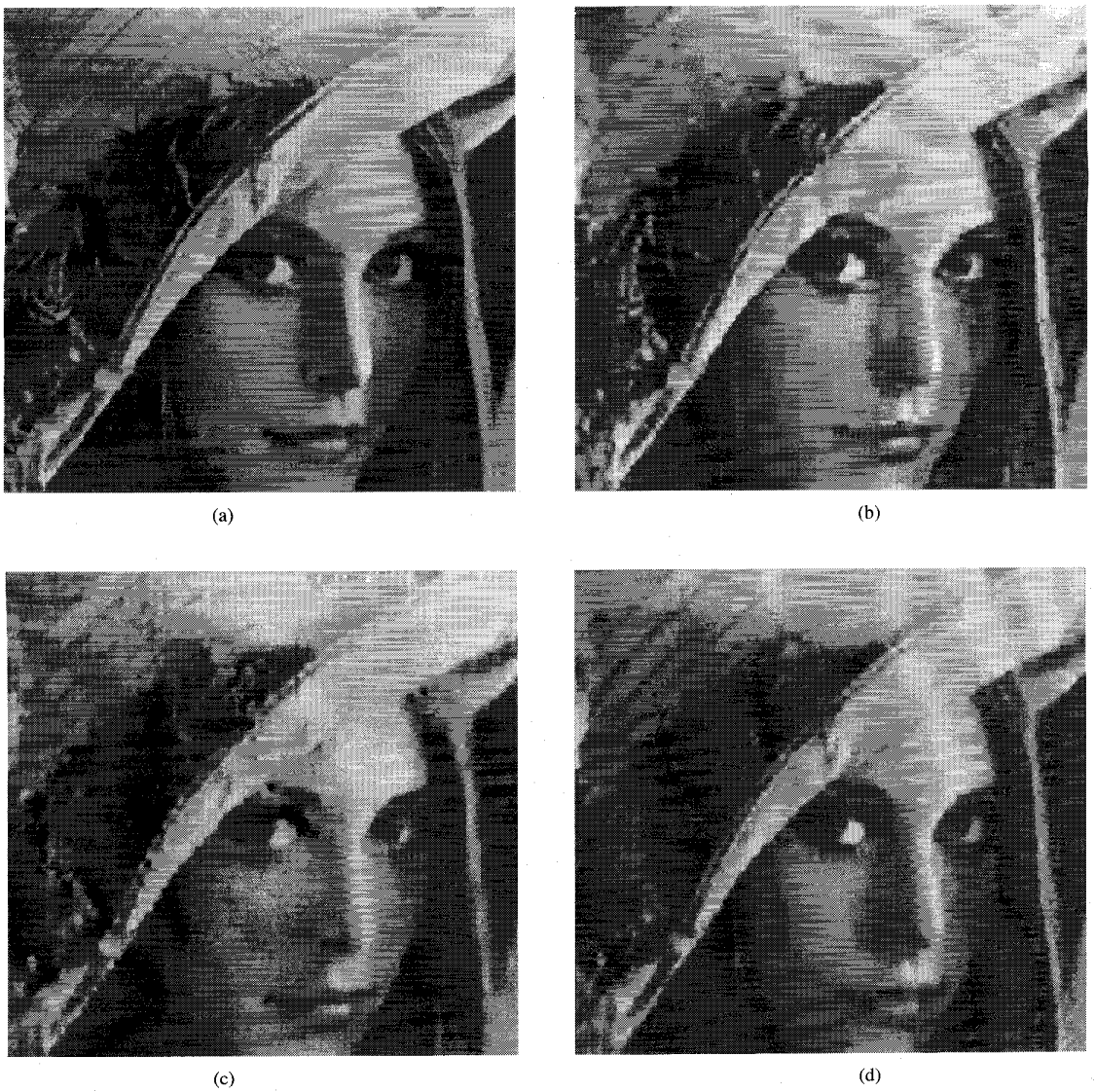


Fig. 10. (a) Original 128 × 128 Lenna face, (b)–(d) comparison of reconstructed Lenna face after compression to 0.3750 bpp (b) using JPEG (c) using DPCM/QCVQ (d) and using standard NRQ/CVQ.

standard version, referred to as standard NRQ/CVQ, which uses the parameters (β_v, β_h) given in Table I evaluated using the likelihood function [3], the regressors (U_i, Θ_i) computed using (11)–(12), and residual codebooks based on the test image only (local codebook); and a practical version, referred to as practical NRQ/CVQ, which uses $\beta_v = 0.246982$ and $\beta_h = 0.243056$ evaluated using the approximated formulas given by (22) and (21), the steady state regressors $(U_\infty, \Theta_\infty)$ computed with the analytical solution given in (19) and (20), and residual codebooks derived from a training sequence. The images used in the training sequence are image1 and image4 of the USC database, and face.256 and hat.256 of the MIT database. Hence, the training sequence does not include the Lenna image which is image18 of the USC database.

The results of both versions of NRQ/CVQ are superior to that of JPEG. JPEG introduces a blocking or tiling effect which results in loss of edges and of most of the details. If we compare the two NRQ/CVQ's, the perceived visual quality of the standard NRQ/CVQ is slightly better than that of the practical NRQ/CVQ. On the other hand, practical NRQ/CVQ offers reduced computational complexity and faster speed, making it practically more attractive. The point is that there is a trade-off between visual quality versus the computational speed. The enhancement in speed offered by practical NRQ/CVQ is worth the price paid in terms of the small loss in visual quality.

Quantitative Evaluation: The PSNR obtained by applying NRQ/CVQ, JPEG, and DPCM/QCVQ to the test images are shown in Table II. The PSNR values confirm the subjective evaluation conclusion that NRQ/CVQ outperforms DPCM/QCVQ and JPEG. In the case of Lenna at 0.188 bpp (a compression ratio of 42.5), there is a gain in PSNR of about 2.8 dB over JPEG and a gain of about 2 dB over DPCM/QCVQ.



(a) (b) (c) (d)

Fig. 11. (a) Original 256 × 256 Lenna face, (b)–(d) comparison of reconstructed Lenna face after compression to 0.1880 bpp (b) using JPEG (c) using DPCM/QCVQ (d) and using standard NRQ/CVQ.

TABLE II
PSNR'S FOR *Lenna*

Compression	NRQ/CVQ	JPEG	DPCM/QCVQ
0.5 bpp	31.73 dB	30.0126 dB	27.1319 dB
0.375 bpp	30.05 dB	28.7050 dB	26.6628 dB
0.188 bpp	26.9146 dB	24.1423 dB	24.9337 dB

The PSNR comparisons also seem to indicate the trend that the performance of NRQ/CVQ improves over JPEG as the compression ratio is increased. For example, the difference in the PSNR's of the two schemes at 0.5 bpp is 1.7174 dB increasing to about 2.78 dB at 0.188 bpp. This seems to be in opposition to the trend perceived when comparing NRQ/CVQ with DPCM/QCVQ. For Lenna, the differences are 4.6 dB at 0.5 bpp and 2 dB at 0.188 bpp. The reason for this behavior is the reduction in the size of the codebook used in VQ. The number of code vectors used decreases with higher compression ratios, causing an increase in the resulting MMSE due to the VQ of the error image. Still, the PSNR of NRQ/CVQ is higher than the PSNR of DPCM/QCVQ. The improved performance of NRQ/CVQ, which uses a noncausal prediction model, over DPCM/QCVQ, which uses a causal model, illustrates the superiority of noncausal prediction over causal prediction. Similarly, the improvement over JPEG shows that NRQ/CVQ can deliver for still images better quality than this international standard. Similar comparisons were also performed over a variety of test images. The results were similar to the ones indicated above.

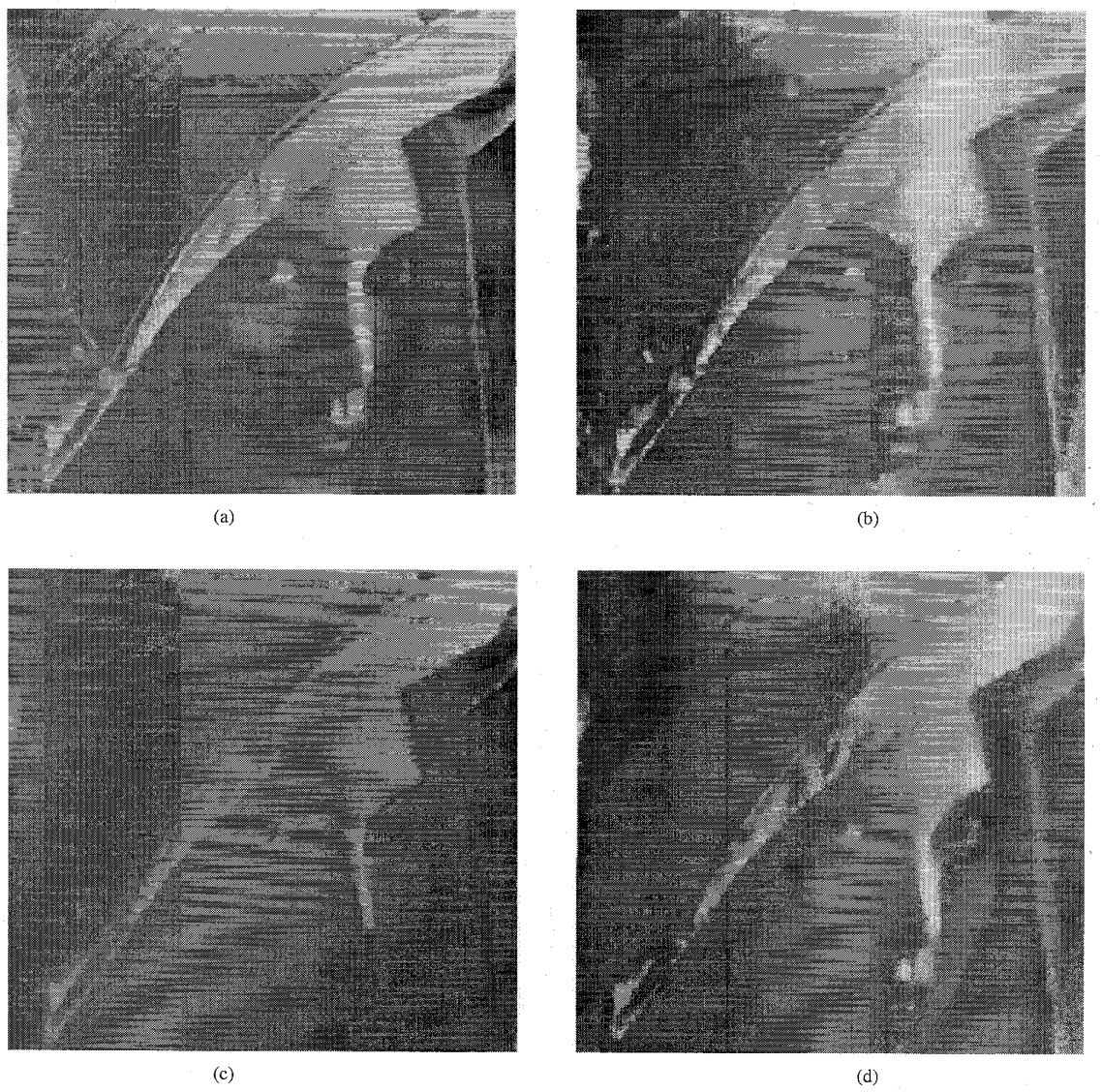


Fig. 12. Original 256 × 256 Lenna face, (b)–(d) comparison of reconstructed Lenna face after compression to 0.1880 bpp (b) using JPEG (c) using practical NRQ/CVQ (d) and using standard NRQ/CVQ.

VIII. Conclusion

We have described a new procedure for lossy compression of still images. We refer to it as the noncausal codec with residual quadtree mean removal and cascaded VQ (NRQ/CVQ). We used NRQ/CVQ to compress images up to $8/0.188 = 42.5:1$ compression ratio. NRQ/CVQ uses a noncausal prediction model to generate the residual image which is considerably less correlated than the original image. The residual image is then compressed using residual quadtree cascaded VQ. It is important to emphasize that quadtree mean removal is after, not before, noncausal prediction. In other words, we remove the block means in the error image resulting from the noncausal prediction step. This has proven in our studies to be much more efficient in both quality delivered and smaller number and larger blocks than block mean removal directly applied to the original image. The performance of NRQ/CVQ was compared to JPEG, and causal prediction with quadtree mean removal cascaded VQ as in DPCM/QCVQ. Our experimental results show that NRQ/CVQ outperform all alternative technologies, exhibiting a higher PSNR and a better perceived image quality. NRQ/CVQ is computationally more complex than JPEG and causal prediction. We discussed several short cuts to the implementation of NRQ/CVQ and the trade-offs in performance.

We summarize the main features of NRQ/CVQ.

- *Noncausal Prediction:* NRQ/CVQ models the real images as a noncausal Gauss Markov random field and utilizes the recursive prediction algorithms developed in [2], [3]. The noncausal prediction model outperforms the causal prediction model. As a result, the residual image generated by subtracting the predicted pixel intensities from the original values is considerably less correlated leading to higher compression.

- *Residual quadtree mean removal:* NRQ/CVQ applies quadtrees to segment the noncausal prediction error image into square blocks of variable size. This segmentation leads to an efficient mean subtraction (MS) procedure resulting in considerable savings in the transmission of the local means.
- *Quadtree coding:* NRQ/CVQ uses a simple pointerless representation of the quadtrees which at the receiver enables the reintroduction of the local means at the appropriate locations.
- *Quadtree cascaded vector quantizer*: NRQ/CVQ quantizes the quadtree mean removed residual image through a modified cascaded vector quantizer. The standard cascaded VQ is modified to incorporate a selector at each stage. The selector serves a dual purpose. First, it prevents additional errors from being introduced in those vectors which are already represented optimally at a previous stage. Second, it attempts to spread the quantization error uniformly. Quadtree cascaded VQ outperforms the conventional single stage VQ, both in terms of speed and PSNR.

REFERENCES

[1] R. M. Gray, S. J. Park, and B. D. Andrews, "Tiling shapes for image vector quantization," in *Proc. Comcon 3, Third Int. Conf. Commun. Contr. Syst.*, Victoria, BC, Oct. 16–18, 1991, pp. 818–829.

[2] N. Balram and J. M. F. Moura, "Recursive structure of noncausal Gauss Markov random fields," *IEEE Trans. Inform. Theory*, vol. 38, no. 2, pp. 334–354, Mar. 1993.

[3] ——, "Noncausal Gauss Markov random fields: Parameter structure and estimation," *IEEE Trans. Inform. Theory*, vol. 39, no. 4, pp. 1333–1355, July 1993.

[4] W. Zshunke, "DPCM picture coding with adaptive prediction," *IEEE Trans. Commun.*, vol. COM-25, pp. 1295–1302, Nov. 1977.

[5] J. W. Woods, "Two-dimensional discrete Markovian fields," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 232–240, 1972.

[6] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, pp. 84–95, Jan. 1980.

[7] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 127–135, Mar. 1982.

[8] R. L. Baker and R. M. Gray, "Differential vector quantization of achromatic imagery," in *Proc. Int. Picture Coding Symp.*, Mar. 1983.

[9] H. Samet, "Quadtree and related hierarchical data structures," in *Tech. Rep. 2, ACM Comput. Surveys*, June 1984, pp. 16:188–260.

[10] P. M. Farrelle, *Recursive Block Coding for Image Data Compression*. New York, NY: Springer Verlag, 1990.

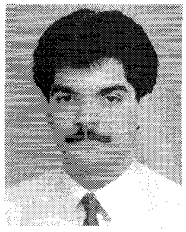
[11] B. H. Juang and A. H. Gray, "Multiple stage vector quantization for speech coding," in *Int. Conf. Acoustics, Speech, Signal Processing*, vol. 1, Paris, Apr. 1982, pp. 597–600.

[12] C. F. Barnes and R. L. Frost, "Vector quantizers with direct sum codebooks," *IEEE Trans. Inform. theory*, vol. 39, pp. 565–580, Mar. 1993.

[13] D. J. Vaisey and A. Gersho, "Variable block-size image coding," in *Int. Conf. Acoustics, Speech, Signal Processing*, 1987, pp. 1051–1054.

[14] G. K. Wallace, "The JPEG still picture compression standard," *Commun. ACM*, vol. 34, pp. 31–44, Apr. 1991.

[15] A. C. Hung, *PVRG-JPEG CODEC 1.1*. Portable Video Research Group, Stanford University, 3rd ed., June 1993.



Amir Asif was born in Lahore, Pakistan, in November 1966. He received the B.S. degree in electrical engineering from University of Engineering and Technology, Lahore, Pakistan, with the highest honor in 1990 and the M.S. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 1993. He is currently a Ph.D. student in electrical and computer engineering at Carnegie Mellon University, Pittsburgh, PA. His research interests include multidimensional statistical signal processing, image coding, and video processing.

José M. F. Moura (S'71–M'75–SM'90–F'94) received the engenheiro electrotécnico degree in 1969 from Instituto Superior Tecnico (IST), Lisbon, Portugal, and the M.Sc., E.E., and the D.Sc. in electrical engineering and computer science from the Massachusetts Institute of Technology (M.I.T.), Cambridge, in 1973 and 1975, respectively.

He is presently a Professor of Electrical and Computer Engineering at Carnegie Mellon University (CMU), Pittsburgh, PA, which he joined in 1986. Prior to this, he was on the faculty of IST where he was an Assistant Professor (1975), Professor Agregado (1978), and Professor Catedratico (1979). He has had visiting appointments at several institutions, including M.I.T. (Genrad Associate Professor of Electrical Engineering and Computer Science, 1984–1986, also associated with LIDS) and the University of Southern California (research scholar, Department of Aerospace Engineering, Summers 1978–1981). His research interests lie in statistical signal processing (one and two dimensional), image and video processing, array processing, underwater acoustics, and multiresolution techniques. He has organized and codirected two international scientific meetings on signal processing theory and applications. He has over 160 technical contributions, including invited ones, published in international journals and conference proceedings, and is coeditor of the books *Nonlinear Stochastic Problems* (Reidal, 1983) and *Accoustic Signal Processing for Ocean Exploration* (Kluwer, 1993).

Dr. Moura is currently the Editor in Chief for the IEEE TRANSACTIONS IN SIGNAL PROCESSING. He is a member of the IEEE Press Board since 1991, he was a technical Associate Editor for the IEEE SIGNAL PROCESSING LETTERS from 1993 to 1995 and a member of the Underwater Acoustics Technical Committee of the Signal Processing Society. He was an Associate Editor for the SIGNAL PROCESSING TRANSACTIONS (Sonar and Radar) from 1988 to 1992 and a member of the technical committee of the IEEE International Symposium on Information Theory (ISIT'1993). He was elected Fellow of the IEEE in November 1993 and corresponding member of the Academy of Sciences of Portugal (Section of Sciences) in July 1992. He is affiliated with several IEEE societies, Sigma Xi, AMS, IMS, and SIAM.