Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Digital Signal Processing 15 (2005) 255–275
Digital Signal Processing
www.elsevier.com/locate/dsp

A fractional bit encoding technique for the GMM-based block quantisation of images

Kuldip K. Paliwal, Stephen So *

School of Microelectronic Engineering, Griffith University, Brisbane QLD 4111, Australia

Available online 14 March 2005

Abstract

In this paper, a simple technique for encoding fractional bit components used in fixed-rate Gaussian mixture model-based (GMM) block quantisers is presented. While block transform image coding has not been very popular lately in the presence of current state-of-the-art wavelet-based coders, the GMM-based block quantiser, without the use of entropy coding, is still very competitive in the class of fixed-rate transform coders. It consists of a set of individual block quantisers operating at different bitrates, and the problem is that these bitrates are mostly fractional. Fixed-rate block quantisers based on an integer number of bits can be designed and through the use of heuristic algorithms, can approach the fractional target rate. However, the use of level-based scalar quantisers in the block quantiser allows better utilisation of the bit budget; a finer 'spread' of the bit budget across components; and better preservation of optimality. Our technique, which is based on a generalisation of positional value number systems, allows the use of level-based scalar quantisers in a fixed-rate coding framework. Experimental results comparing the use of the bits-based GMM-based block quantiser with the levels-based one in image coding show a finite improvement in the PSNR performance.

Keywords: Image coding; Transform coding; Block quantisation; Gaussian mixture models

* Corresponding author.
E-mail addresses: k.paliwal@griffith.edu.au (K.K. Paliwal), s.so@griffith.edu.au (S. So).

1051-2004/$ – see front matter
doi:10.1016/j.dsp.2004.11.002

1. Introduction

Block quantisation or transform coding,[1] has been used as a less complex alternative to full-search vector quantisation in the coding of images [3]. Proposed by Kramer and Mathews [13] and analysed by Huang and Schultheiss [12], it involves decorrelating the components within a block or vector of samples using a linear transformation before scalar quantising each component independently. When quantising for minimum distortion, the Karhunen–Loève transform (KLT) is the best transform for correlated Gaussian sources [9].

The probability density functions (PDF) of real life sources are rarely Gaussian. Since the non-uniform scalar quantisers in block quantisers are designed to quantise Gaussian sources efficiently, any PDF mismatch will invariably cause a degradation in performance. Rather than assuming the source PDF to be a standard function such as Gaussian, Laplacian, etc., one can design a quantiser that matches the source PDF as close as possible and quantises the source optimally. The K-means algorithm, otherwise known in the literature as the generalised Lloyd algorithm (GLA) and Linde–Buzo–Gray (LBG) algorithm,[2] allows one to design vector quantisers which match the PDF of training data and quantise it optimally (at least, locally optimal) via the Lloyd conditions [8,14].

There have been numerous studies in the coding literature on source PDF modelling for quantiser design. The basic idea is that if we can compactly model the PDF of the source accurately, then we can design quantisers that can code this source efficiently. These can be broadly classified as either *non-parametric* or *parametric modelling*. Ortega and Vetterli [15] estimated the source model in a non-parametric fashion using piecewise linear approximation. Similarly, multidimensional histograms were used by Gardner and Rao [7] to model the PDFs of line spectral frequencies (LSF) in order to evaluate the theoretical bounds of split vector quantisers. In relation to parametric modelling, Su and Mercereau [20] applied Gaussian mixture models (GMM) in the estimation of the PDF of DC transform coefficients. On the speech side, Hedelin and Skoglund [11] and Samuelsson and Hedelin [17] used GMMs for designing and evaluating high-rate vector and recursive quantisers of LSFs, respectively, while Subramaniam and Rao [18] applied GMMs to improve block quantisation.

In our previous work [16], where we applied the GMM-based block quantiser of [18] to the fixed-rate coding of images, we showed that this quantiser was able to achieve peak signal-to-noise ratio (PSNR) gains of 5–10 dB at high bitrates and 2–4 dB at medium to low bitrates over the single Gaussian block quantiser. The advantages of the GMM-based block quantiser, compared with vector quantisers, are [18]:

[1] While these two terms are generally synonymous, the latter term generally encompasses coders that use variable rate entropy coding after the quantisation step, like the JPEG image coding standard [22], while block quantisers specifically use non-uniform scalar quantisers of fixed rate with no entropy coding [12,23]. This paper is focused on block quantisation only.

[2] The only notable difference between the K-means algorithm and GLA/LBG algorithm is the initialisation used. Since the refinement steps, which are based on the Lloyd conditions for optimality, are essentially the same, we will use the terms 'K-means algorithm,' 'GLA,' and 'LBG algorithm,' interchangeably in this paper.

- Compact representation of the source PDF, stored at the encoder and decoder.
- Bitrate scalability with 'on-the-fly' bit allocation.
- Low search complexity and memory requirements which are independent of the bitrate of the system.

A block quantiser, which we will refer to as the *cluster block quantiser*, is designed for each Gaussian mixture component or cluster. For fixed-rate quantisation, the total number of bits available for quantising each vector is constant and these bits are non-uniformly distributed to each cluster block quantiser, depending on the covariance and probability of that cluster in the GMM. This non-uniform bit allocation among clusters (intercluster bit allocation) results in the cluster block quantisers being given a fractional number of bits. These will need to be truncated as scalar quantisers generally operate at integer bits. By using scalar quantisers based on levels rather than bits, we can achieve a finer level of granularity since the number of levels is not limited to powers of two, as is the case with integer numbers of bits. However, encoding the quantiser level indices in a fixed-rate quantiser framework is not as straightforward since the total number of levels is multiplicative as opposed to the total number of bits being additive, for each block.

In this paper, we present a simple method, based on a generalisation of positional-value number systems, for binary encoding the quantiser level indices of the cluster block quantisers in a fixed-rate GMM-based block quantiser. This technique is not only computationally simple, but also leads to some improvement in the PSNR. Also, algorithms for dealing with cases of negative and non-integer bit allocations are presented.

The organisation of the paper is as follows: Section 2 gives an introduction to GMM-based block quantisation as well as its bit allocation schemes. Following this, the using of quantiser levels in block quantisers and the problems of binary encoding them is defined in Section 3. Section 4 gives an introduction to positional value number systems which provides the preliminaries for Section 5, where our proposed method of using a generalised number system to encode fractional bits/integer levels is described. Section 6 lists some heuristic-based methods for compensating under-allocation and over-allocation problems in the context of quantiser levels. Section 7 describes the experiments we have performed to evaluate this quantisation scheme. Experimental results are presented in Section 8 as well as a discussion of the benefits of our scheme. In Section 9, we give our conclusions as well as issues which require further investigation.

2. GMM-based block quantisation

The GMM-based block quantiser of [18] models any arbitrary source of data vectors as a mixture of individual and overlapping Gaussian sources (or, clusters), otherwise known as a Gaussian mixture model (GMM). In an m-cluster GMM framework, each observation vector is assumed to be generated *by one of the Gaussian sources*, depending on the weight or probability of that cluster. Using the GMM framework for source modelling allows us to decompose any complex and arbitrary PDF into a series of Gaussian basis PDFs and for each Gaussian basis source, we can then design efficient quantisers. This is analogous to

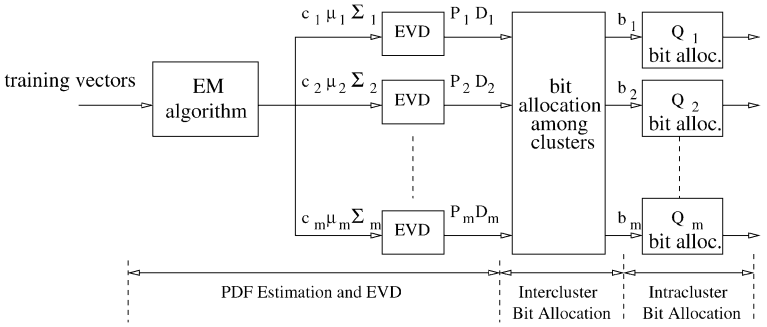

Fig. 1. PDF estimation and bit allocation from training data.

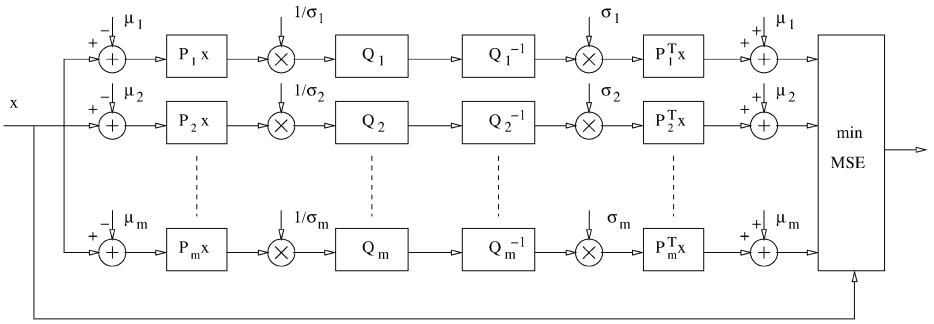

Fig. 2. Minimum distortion block quantisation (Q-cluster block quantiser).

the transform coding idea of breaking the signal down into basis components and designing individual quantisers for each one.

Block quantisers [12], which consist of the Karhunen–Loève transform in combination with Gaussian Lloyd–Max scalar quantisers, are known to be efficient for correlated Gaussian sources [9], hence one is designed for each cluster. Because an observation vector is assumed to be generated by one Gaussian source only, then the cluster block quantiser for that source will code it will minimal distortion (in theory, at least). However, the overlapping of Gaussian clusters coupled with the relatively low complexity of block quantisers allows us to use a soft-clustering approach to choose the best cluster block quantiser based on minimum distortion, rather than maximum likelihood [18].

This quantisation scheme differs from the adaptive transform coders [1–3,5,6] found in the image coding literature, where the vector space is partitioned into *disjoint* regions and transform coders are designed for each region. The aim of these adaptive transform coders is to partition the vector space into regions of similar statistics, in conjunction with the design of transform coders, in order to minimise quantiser distortion [3]. Whereas for the GMM-based block quantiser, the aim is to approximate the source PDF accurately using a GMM, decomposing it into multiple overlapping Gaussian clusters [18]. Block quantisers, which can code correlated Gaussian sources efficiently, are then designed for each cluster.

The GMM-based block quantiser can be broken down into three stages: PDF estimation, bit allocation, and minimum distortion block quantisation. These will be described in the

following sections. Figures 1 and 2 show the PDF estimation and bit allocation procedure and minimum distortion block quantisation, respectively. These are similar to those that appear in [18].

2.1. PDF estimation using Gaussian mixture models

The PDF model, G, as a mixture of multivariate Gaussians, $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$, can be given by

$$G(\boldsymbol{x}|\boldsymbol{M}) = \sum_{i=1}^{m} c_i \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \tag{1}$$

$$\boldsymbol{M} = \left[m, c_i, \ldots, c_m, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_m\right], \tag{2}$$

$$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} e^{-(1/2)(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}, \tag{3}$$

where $\boldsymbol{x}$ is a source vector, m is the number of mixture components (or clusters), and n is the dimensionality of the vector space. c_i, $\boldsymbol{\mu}_i$, and $\boldsymbol{\Sigma}_i$ are the weight, mean, and covariance matrix of the ith mixture, respectively. Note the words 'mixture component' and 'cluster' will be used interchangeably in this paper.

The parametric model parameters, $\boldsymbol{M}$, are initialised by applying the LBG algorithm [14] on the training vectors representing the source distribution and m clusters are produced, each represented by a mean or centroid, $\boldsymbol{\mu}$, a covariance matrix, $\boldsymbol{\Sigma}$, and a cluster weight, c. These form the initial parameters for the GMM estimation procedure. Using the expectation–maximisation (EM) algorithm [4], the maximum-likelihood estimate of the parametric model is computed iteratively and a final set of means, covariance matrices, and weights are produced.

An eigenvalue decomposition (EVD) is found which converts the m covariance matrices into diagonal matrices, $\{\boldsymbol{D}_i\}_{i=1}^{m}$, producing m eigenvalues, $\{\boldsymbol{\lambda}_i\}_{i=1}^{m}$, and eigenvectors. The eigenvectors form the rows of the orthogonal transformation matrix, $\boldsymbol{P}$, of the Karhunen–Loève transform.

Since the GMM parameters for the estimated PDF and the KLT transformation matrices are properties of the source and thus independent of the bitrate, this procedure only needs to be done once (training). The GMM parameters and KLT transformation matrices are stored at the encoder and decoder, hence there is no transmission overhead associated [18].

2.2. Bit allocation

There are two types of bit allocation that are required: *intercluster bit allocation* and *intracluster bit allocation*. Bitrate scalability is a feature of the GMM-based block quantiser [18]. The bit allocation, depending on the chosen bitrate, is done 'on-the-fly' by both the encoder and decoder, based on the common PDF model and KLT parameters. Therefore, the bit allocation is fixed and synchronised between the encoder and decoder. For the purpose of 'on-the-fly' bitrate scalability, the bit allocation procedure needs to be computationally efficient. For this reason, classical bit allocation algorithms, based on the

high-resolution performance of Gaussian scalar quantisers [12], are preferred over Riskin's algorithm [19], though the latter achieves performance that is closer to the theoretical rate-distortion curve than high-resolution based algorithms.

2.2.1. Intercluster bit allocation

With intercluster bit allocation, the range of quantiser levels is partitioned to each of the m cluster block quantisers. Following the derivation given in Ref. [18], for a fixed-rate quantiser, the total number of quantiser levels is fixed,

$$2^{b_{\text{tot}}} = \sum_{i=1}^{m} 2^{b_i}, \tag{4}$$

where b_{tot} is the total number of bits in the bit budget, b_i is the number of bits assigned to cluster i, and m is the number of clusters. Since the cluster weights can be thought of as probabilities of occurrence of each cluster [18], the average distortion is approximated by[3]

$$D_{\text{tot}} = \sum_{i=1}^{m} c_i D_i(b_i). \tag{5}$$

Using the high resolution approximation for distortion of a single Lloyd–Max scalar quantiser, the total distortion of a block quantiser is [18]

$$D_i(b_i) = K n \Lambda_i 2^{-2b_i/n}, \tag{6}$$

$$\Lambda_i = \left[\prod_{j=1}^{n} \lambda_{i,j} \right]^{1/n} \quad \text{for } i = 1, 2, \ldots, m, \tag{7}$$

where n is the dimension of the vectors, m is the number of clusters, $\lambda_{i,j}$ is the jth eigenvalue of cluster i, and K is a constant which is equal to $\pi\sqrt{3}/2$ for Gaussian sources [10].

Using Lagrangian minimisation, the distortion can be minimised under the fixed rate constraint of (4), and the following bit allocation formula is derived [18]:

$$2^{b_i} = 2^{b_{\text{tot}}} \frac{(c_i \Lambda_i)^{n/(n+2)}}{\sum_{i=1}^{m} (c_i \Lambda_i)^{n/(n+2)}} \quad \text{for } i = 1, 2, \ldots, m, \tag{8}$$

where c_i is the weight of cluster i.

2.2.2. Intracluster bit allocation

After the bit budget is partitioned to each cluster, the bits need to be allocated to each of the n components within each cluster block quantiser using existing bit allocation techniques for transform coding [8,18]. Following the derivation presented in [12], the total number of bits is fixed:

$$b_i = \sum_{j=1}^{n} b_{i,j} \quad \text{for } i = 0, 1, \ldots, m-1, \tag{9}$$

[3] Note that this is based on the assumption is that there is negligible overlap between clusters.

where $b_{i,j}$ is the number of bits assigned to component j of cluster i. Again, using the high resolution approximation for the distortion of a Lloyd–Max scalar quantiser, the average distortion of cluster i is given by [12]

$$D_i = \frac{1}{n}\sum_{j=1}^{n} \lambda_{i,j} K 2^{-2b_{i,j}} \quad \text{for } i = 1, 2, \ldots, m. \tag{10}$$

Using Lagrangian minimisation, the average distortion is minimised under the fixed rate constraint of (9) and the following is obtained:

$$b_{i,j} = \frac{b_i}{n} + \frac{1}{2}\log_2 \frac{\lambda_{i,j}}{\left[\prod_{j=1}^{n} \lambda_{i,j}\right]^{1/n}} \quad \text{for } i = 1, 2, \ldots, m. \tag{11}$$

2.3. Minimum distortion block quantisation

Figure 2 shows a diagram of minimum distortion block quantisation. At first glance, it can be seen to consist of m independent Gaussian block quantisers, Q_i, each with their own orthogonal matrix, $\boldsymbol{P}_i$, and bit allocations, $\{b_{i,j}\}_{j=1}^{n}$. The following coding process is also described in Ref. [18].

To quantise a vector, $\boldsymbol{x}$, using a particular cluster i, the cluster mean, $\boldsymbol{\mu}_i$, is first subtracted and its components decorrelated using the orthogonal matrix, $\boldsymbol{P}_i$, for that cluster. The variance of each component is then normalised by the standard deviation to produce a decorrelated, mean-subtracted, and variance-normalised vector, $\boldsymbol{z}_i$,

$$\boldsymbol{z}_i = \frac{\boldsymbol{P}_i(\boldsymbol{x} - \boldsymbol{\mu}_i)}{\boldsymbol{\sigma}_i}. \tag{12}$$

These are then quantised using a set of n Gaussian Lloyd–Max scalar quantisers as described in Ref. [12] with their respective bit allocations producing indices, $\boldsymbol{q}_i$.[4] These are decoded to give the approximated normalised vector, $\hat{\boldsymbol{z}}_i$, which is multiplied by the standard deviation and correlated again by multiplying with the transpose, $\boldsymbol{P}_i^T$, of the orthogonal matrix. The cluster mean is then added back to give the reconstructed vector, $\hat{\boldsymbol{x}}_i$,

$$\hat{\boldsymbol{x}}_i = \boldsymbol{P}_i^T \boldsymbol{\sigma}_i \hat{\boldsymbol{z}}_i + \boldsymbol{\mu}_i. \tag{13}$$

The distortion between this reconstructed vector and original is then calculated, $d(\boldsymbol{x} - \hat{\boldsymbol{x}}_i)$.

The above procedure is performed for all clusters in the system and the cluster, k, which gives the *minimum distortion* is chosen:

$$k = \underset{i}{\operatorname{argmin}}\, d(\boldsymbol{x} - \hat{\boldsymbol{x}}_i). \tag{14}$$

We have chosen to use the mean-squared-error as the distortion measure. It is noted that the search complexity is a function of the number of clusters, m, rather than the bitrate [18]. Thus it is a computational advantage over full search vector quantisers where the search complexity is an exponential function of the bitrate [10].

[4] The non-uniform scalar quantisers may be replaced with uniform scalar quantisers with appropriate companding and expanding, as is done in Ref. [18].

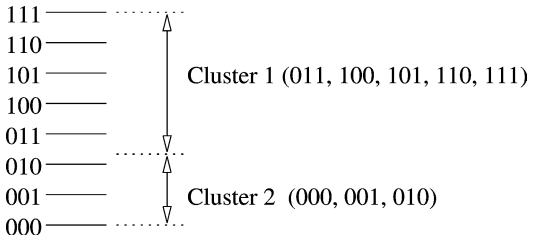

Fig. 3. Example of quantiser level encoding and cluster number partitioning.

2.4. Quantiser index encoding

Each quantised vector has associated with it, a number identifying which cluster was used for coding. As proposed in Ref. [18], this side information can be made inherent in the encoding. For an m cluster system operating at b bits per vector, $\log_2 m$ bits are required to uniquely identify each cluster. Therefore, on average, $b - \log_2 m$ bits are available for quantising each vector which is equivalent to $2^b/m$ quantiser levels. Hence, our range of quantiser levels has been partitioned into m segments.

In effect, this partitioning of the range of quantiser levels allows the cluster number to be found by determining which partition the block code belongs to. An example of this encoding scheme is shown in Fig. 3 where a total of 3 bits are available to encode each block and the system uses 2 clusters. Cluster 1 has been assigned 5 levels (2.322 bits) while cluster 2 has the remaining 3 levels (1.583 bits). If cluster 1 was deemed the most suitable for encoding the current block, its quantiser levels would be contained within the range of $011\ldots101$. Therefore, the decoder can easily determine which cluster the block belongs to by working out which partition the code falls into. Hence this removes the need for extra side information to be transmitted.

The example also shows that the number of levels assigned to the block quantiser belonging to each cluster are not powers of two and hence the bits assigned to the quantisers are fractional.

3. Allocation of levels for a single block quantiser

Traditionally, the design of individual scalar quantisers for each respective component of a block is done by allocating quantiser bits to components based on their relative variances. This is the method presented in Ref. [12] which leads to (11). The constraint used in the minimisation derivation is that the sum of the bits to encode a block is constant, and so fixed-rate coding is assumed. That is

$$b_{\text{tot}} = \sum_{i=1}^{n} b_i, \tag{15}$$

where n is the size of the blocks, b_i is the number of bits allocated to the ith component, and b_{tot} is the total number of bits in the bit budget. By noting that the relation

Table 1
Example of integer bit allocation table

3	2	1	1

Table 2
Example of binary coding a block

101	11	0	0	x

Table 3
Example of fractional bit allocation table

9	5	2	2

between quantiser levels, l, and quantiser bits, b, is $b = \log_2 l$, then (15) can be written as

$$\log_2 l_{\text{tot}} = \sum_{i=1}^{n} \log_2 l_i = \log_2 \prod_{i=1}^{n} l_i, \qquad l_{\text{tot}} = \prod_{i=1}^{n} l_i. \tag{16}$$

Equation (16) effectively says that in terms of quantiser levels, the *product* of all the individual levels, l_i, must equal the total, $l_{\text{tot}} = 2^{b_{\text{tot}}}$. While this conversion from bits and levels is fairly easy, the encoding of quantiser levels in a block is not so straightforward. To demonstrate the problem, it is best to use a simple example.

Consider a fixed-rate cluster block quantiser, in the context of GMM-based block quantisation, that uses blocks of size 4 and the number of bits allocated to coding each block is 7.492 bits. This means that the target bitrate is 1.873 bits per sample. Also, assume the integer bit allocation calculated for the source is shown in Table 1. One can see that in terms of integer bits, the allocated bitrate is only 1.75 bits per sample, rather than the target 1.873 bits per sample. Therefore, under-allocation has occurred. However, the encoding of a block is fairly straightforward. If the quantiser indices for each component are shown in Table 2 (where the 'x' shows an unused bit that is padded on to ensure constant bitrate). Then this block would be encoded as 10111000_2 or 184_{10}.

However, consider an allocation table based on integer levels in Table 3. The total levels allocated are $9 \times 5 \times 2 \times 2 = 180$ while the target number of levels is $2^{7.492} \approx 180$. The effective bitrate achieved when using quantiser levels is 1.8729 bits per sample which is very close to the target bitrate of 1.873. Therefore when using quantiser levels, more of the bit budget is utilised and this should translate to better performance. Another way of achieving a fractional bitrate is using variable-rate entropy coding. This is what is essentially done in JPEG, where scalar quantisers, based on quantiser levels, are applied on discrete cosine transform coefficients [22]. However, in contrast with the levels-based block quantiser considered in this paper, JPEG uses uniform scalar quantisation coupled with runlength and Huffman encoding which is known to significantly outperform fixed-rate block quantisation [23]. However, we are only considering a fixed-rate quantiser which does not have the added complexity of variable-rate schemes.

However, the method of encoding these levels is not as straightforward as encoding bits. The complication arises from the fact that fractional bits are used and it is not well-defined as to how one can allocate a fractional part of the bit budget to each component, remembering that the total number of levels consists of a *product* of the individual component levels rather than a *sum*. For example, in the first component of Table 3, where 9 levels are allocated, this corresponds to approximately 3.17 bits while in the second component, 3 levels corresponds to approximately 2.322 bits, etc. That is, how is one to combine these fractional bits so that the block can be coded into an integer of 180 levels or 7.492 bits? In order to develop an understanding into how one may approach the solution to this problem, one needs to investigate positional value number systems and its generalisations.

4. Positional value number systems

Most common number systems used are *positional value systems* where the value of a digit is determined by its position [21]. For example, consider the number 3512_{10}. The value is determined by multiplying each digit with its positional value or weight, w. That is, $3512_{10} = 3 \times 10^3 + 5 \times 10^2 + 1 \times 10^1 + 2 \times 10^0$. Each positional weight is related to the number of states or levels that can occur in the previous position. For example, in the decimal system:

- The least significant position (position 1) can take on 10 different levels or numerals (0, 1, 2, 3, 4, 5, 6, 7, 8, 9).
- The next least significant position (position 2) can take on 10 different levels as well. Each level in position 2 represents 10 'lots' of levels of position 1, thus the positional weight is 10.
- Each level in position 3 represents 10 'lots' of levels of position 2 which each have the weight of 10, thus the positional weight is $10 \times 10 = 100$; and so forth.

This system of determining the positional weights is summarised in Table 4. As shown in this table, the positional weights are determined by

$$w_n = \prod_{i=0}^{n-1} l_i, \quad \text{where } l_0 = 1. \tag{17}$$

For the decimal case, the number of levels, l, is equal to 10 for all positions. With this system defined, it allows other number systems to be created where each position may have different levels. For example, consider the number system, S, whose positional weighting system is shown in Table 5. To find the decimal equivalent of a number 2013_S:

$$2013_S = 2 \times 80 + 0 \times 10 + 1 \times 5 + 3 \times 1 = 168_{10}.$$

There are two ways of converting a decimal number into the S number system. Method 1 involves repeated division with each positional weight. One obtains the digits by truncating the quotient and continuing the division on the remainder as shown in Table 6. Method 2 involves repeated division with the number of levels. One obtains the digits via the remainders, as shown in Table 6.

It can be seen that the mapping from any number system of defined levels to the decimal system is a one-to-one mapping and is reversible.

Table 4
System of positional weights

Position	n	...	4	3	2	1
Number of levels	l_n	...	l_4	l_3	l_2	l_1
w	$\prod_{i=0}^{n-1} l_i$	...	$l_3 l_2 l_1$	$l_2 l_1$	l_1	1

Table 5
Positional weights for the S number system

Position	4	3	2	1
Number of levels	7	8	2	5
w	80	10	5	1

5. Fractional bit encoding

Using the concept of positional value number systems where any arbitrary system of defined levels can be mapped to a decimal number, it is possible to apply the technique to quantising blocks using a fractional number of bits.

Consider a fixed-rate GMM-based block quantiser which operates on vectors of dimension n with the total bit budget, b_{tot}. After the application of (9), b_i bits are assigned to the ith cluster block quantiser, where b_i may be fractional. After the application of (11), the bit allocation is obtained for each component of this cluster block quantiser, $\boldsymbol{b}_i = [b_{i,1}, b_{i,2}, b_{i,3}, \ldots, b_{i,n}]$, which are fractions and these are converted to integer quantiser levels, $\boldsymbol{l}_i = [l_{i,1}, l_{i,2}, l_{i,3}, \ldots, l_{i,n}]$. The encoding number system for this cluster block quantiser is the same as Table 4 consisting of a series of positional weights, $\boldsymbol{w}_i = [w_{i,1}, w_{i,2}, \ldots, w_{i,n}]$, which are calculated via (17). These positional weights remain fixed since the bit allocation is based on the static GMM and KLT parameters. Each transformed block, $\boldsymbol{y} = [y_1, y_2, \ldots, y_n]$, is quantised to produce levels indices, $\boldsymbol{q} = [q_1, q_2, \ldots, q_n]$. These level indices of the block are then encoded into a b_{tot}-bit integer, z, via the following formula:

$$z = \boldsymbol{w}\boldsymbol{q}^T \tag{18}$$

$$= \sum_{i=1}^{n} w_i q_i . \tag{19}$$

In order to decode this code, z, into the respective levels for the block, the procedure shown in either Tables 6 or 7 is followed.

It is useful to show that the range of integers that result from this fractional bit encoding is within that of a b_{tot}-bit integer. The maximum level for component i is $l_i - 1$, thus using (17) and (19), the maximum possible integer is given by

$$z = \sum_{j=1}^{n} \left[(l_j - 1) \prod_{i=0}^{j-1} l_i \right] = \sum_{j=1}^{n} \left(l_j \prod_{i=0}^{j-1} l_i - \prod_{i=0}^{j-1} l_i \right)$$

$$= l_n \prod_{i=0}^{n-1} l_i - \prod_{i=0}^{n-1} l_i + l_{n-1} \prod_{i=0}^{n-2} l_i - \prod_{i=0}^{n-2} l_i + l_{n-2} \prod_{i=0}^{n-3} l_i - \cdots - 1. \tag{20}$$

The second term and third term cancel each other and the fourth cancels with the fifth, etc., in (20). Thus only the first term and last remain.

Table 6
Example of converting decimal to the S number system via method 1

80	168	
10	rem 8	**2**
5	rem 8	**0**
1	rem 3	**1**
	rem 0	**3**

Table 7
Example of converting decimal to the S number system via method 2

5	168	
2	rem 33	**3**
8	rem 16	**1**
7	rem 2	**0**
	rem 0	**2**

$$z = l_n \prod_{i=0}^{n-1} l_i - 1 \tag{21}$$

$$= l_n \prod_{i=1}^{n-1} l_i - 1 \quad \text{since } l_0 = 1 \tag{22}$$

$$= \prod_{i=1}^{n} l_i - 1 \tag{23}$$

$$= l_{\text{tot}} - 1 \tag{24}$$

$$= 2^{b_{\text{tot}}} - 1. \tag{25}$$

Therefore it has been shown that the range of the block code, z, is from 0 to $2^{b_{\text{tot}}} - 1$ which is also that of a b_{tot}-bit integer.

6. Heuristic algorithms for compensating quantiser levels

6.1. Under-allocation

'Under-allocation' can occur because of the truncation of the number of levels to make them integers. The remaining levels can then be assigned heuristically based on 'fixed slope' or Pareto optimality [10] and is similar to Riskin's algorithm [19]. The idea is to approximate which component results in the most distortion drop when given an extra level while at the same time ensuring the product of the levels is equal to or below the target.

The high resolution performance of a Lloyd–Max scalar quantiser provides a way of approximating which component would lead to the largest drop in distortion if an extra level was assigned. The formula can be modified to be in terms of levels by substituting $b = \log_2 l$,

$$D(b) = \lambda K 2^{-2b}, \tag{26}$$

$$D(l) = \frac{\lambda K}{l^2}. \tag{27}$$

The distortion drop that would result when adding an extra level is therefore approximated by

$$\Delta D = D(l) - D(l+1) \tag{28}$$

$$= \frac{\lambda K}{l^2} - \frac{\lambda K}{(l+1)^2} \tag{29}$$

$$= \lambda K \left[\frac{2l+1}{l^2(l+1)^2} \right]. \tag{30}$$

Therefore the variables which determine the distortion drop are the variance of the component, λ, as well as the number of levels, l, already assigned to that component. It is apparent that when l is large, the denominator of (30) increases faster than the numerator.

Or in other words, as the operating rate of the scalar quantiser becomes higher, the performance improvement resulting from an extra level 'thins out.' Hence this agrees with the convex exponential behaviour of scalar quantisers predicted by high resolution analysis. The high resolution approximation may not be accurate at low rates so (30) serves as a guide only.

Once the estimated distortion drops of each component are determined, then the next step is to choose the component whereby an increase in levels will result in the total levels not exceeding the target. This can be done by calculating the change in the total product of levels if one component is increased by one,

$$l_{\text{tot}}^{(0)} = \prod_{i=1}^{n} l_i. \tag{31}$$

If a level is added to component j:

$$l_{\text{tot}}^{(1)} = (l_j + 1) \prod_{i=1,\ i\neq j}^{n} l_i \tag{32}$$

$$= \prod_{i=1}^{n} l_i + \prod_{i=1,\ i\neq j}^{n} l_i. \tag{33}$$

Hence the increase in the number of levels is

$$\Delta l = l_{\text{tot}}^{(1)} - l_{\text{tot}}^{(0)} \tag{34}$$

$$= \prod_{i=1,\ i\neq j}^{n} l_i. \tag{35}$$

If the increase in the total number of levels resulting from giving a level to a certain component is more than what is required, then the component with the second most distortion drop is tested and so forth. Once a component has been chosen, its expected distortion drop is updated.

6.2. Over-allocation

'Over-allocation' occurs when the target bitrate is low and some of the allocated bits become negative. These are set to zero which makes the total number of bits exceed the desired total. The components which, when having a quantiser level taken away, induce the least distortion increase are chosen. This procedure is repeated until the total product of the levels falls below the allocated number. Therefore, the over-allocation situation becomes one of under-allocation and the process outlined in Section 6.1 can be followed.

In order to approximate the increase in distortion as a result of removing a quantiser level from a component, a similar derivation to the under-allocation case can be performed:

$$D(b) = \lambda K 2^{-2b}, \tag{36}$$

$$D(l) = \frac{\lambda K}{l^2}. \tag{37}$$

The distortion increase that would result when subtracting a level is therefore approximated by

$$\Delta D = D(l-1) - D(l) \tag{38}$$

$$= \frac{\lambda K}{(l-1)^2} - \frac{\lambda K}{l^2} \tag{39}$$

$$= \lambda K \left[\frac{1-2l}{l^2(l-1)^2} \right]. \tag{40}$$

7. Experimental setup

A fixed-rate GMM-based block quantiser was developed for image coding where blocks of 8×8 were used. A single Gaussian block quantiser with a static KLT, trained over all the training images, was also implemented to serve as a baseline for comparison with the GMM-based scheme. In total, 73,728 vectors from 18×8-bit greyscale images of size 512×512 were used in the training of the GMM and KLT and (8) was used to assign fractional bits to each cluster block quantiser. Six additional images ('boat,' 'kids,' 'crowd,' 'hat,' 'mill,' 'vegas'), which were not part of the training set, along with four of the training images ('Lena,' 'Einstein,' 'jet,' 'goldhill') were used for evaluation. Peak signal-to-noise-ratio (PSNR) was used to judge objective image quality.

For each cluster block quantiser, two different bit allocation algorithms were used for comparison. In the first algorithm, given the target bitrate for the cluster block quantiser, fractional bits from (11) were truncated to form integers. Then a bit-wise heuristic algorithm was applied to compensate for the remaining bits. In the second algorithm, fractional bits from (11) were converted to levels and then truncated. Level-wise heuristics, described in Section 6, were then applied to compensate for the truncation. Positional weights were determined based on the number of levels of each component and these were used to encode each block at a fractional bitrate.

8. Results and discussion

8.1. Comparison between single Gaussian block quantiser and GMM-based block quantiser

The improvements to image quality resulting from more accurate modelling of the PDF can be seen in Fig. 4 where the training image 'goldhill' was compressed using single Gaussian block quantisation as well as using the improved GMM-based block quantiser at the fixed bitrate of 0.5 bits/pixel. In Fig. 4b, it can be seen that the sky as well as the ground looks very grainy. Also, there is a large degree of blocked distortion observable in region of the houses, especially on the edges of the white house in the centre of the picture. By using a 2 cluster GMM to model the PDF, one can see in Fig. 4c that the sky and ground area appear much smoother. There is some apparent distortion observable in the fields immediately behind the houses. In Fig. 4d, where a 16 cluster GMM is used, the

Fig. 4. 'Goldhill' image (part of training set): (a) original 8-bit image; (b) Gaussian KLT-based block quantiser at 0.5 bits/pixel (28.07 dB); (c) GMM-based block quantiser using 2 clusters at 0.5 bits/pixel (29.58 dB); (d) GMM-based block quantiser using 16 clusters at 0.5 bits/pixel (30.74 dB).

sky and ground are considerably smoother and less noisy and distortions in the fields has been considerably reduced. Similarly, Fig. 5 shows the image 'boat,' which was not part of the training set. In Fig. 5b, where a single Gaussian block quantiser was used, the smooth areas such as the sky and the black bottom of the boat are very grainy. There is also block distortion on the white sides of the boat. It can be observed that in Figs. 5c and 5d, as more clusters are used, the graininess of the smooth regions has been reduced as well as the block distortions.

8.2. Comparison between integer and fractional bit-based cluster block quantisers in the GMM-based block quantiser

Tables 8 and 9 show the quantiser bits and level allocations for cluster 1, respectively, for a GMM-based block quantiser operating at 0.15 bits/pixel. It can be seen that the use of

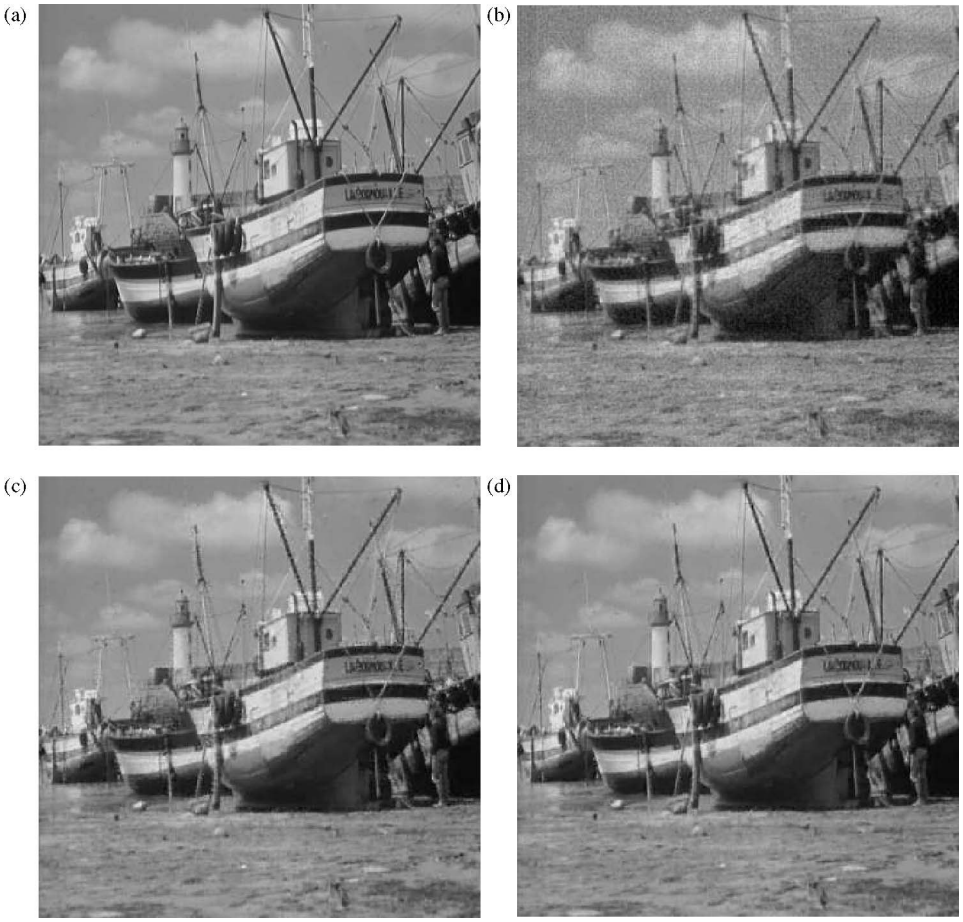

Fig. 5. 'Boat' image (not part of training set): (a) original 8-bit image; (b) Gaussian KLT-based block quantiser at 0.5 bits/pixel (26.02 dB); (c) GMM-based block quantiser using 4 clusters at 0.5 bits/pixel (28.12 dB); (d) GMM-based block quantiser using 16 clusters at 0.5 bits/pixel (29.24 dB).

quantiser levels allows comparably finer granularity in the adjustment of components than through the use of quantiser bits. For example, at such a low bitrate, only the first three coefficients are 'significant.' These are 3, 1, and 1, respectively, when using an integer number of bits or equivalently a total of 32 quantiser levels. Looking at Table 9 where fractional bits/integer levels are used, the first three coefficients are allocated 10, 3, and 2, respectively, which is equivalent to a total of 60 quantiser levels. Hence one can conclude that with greater freedom in the 'spreading' of fractions rather than integers (number of levels limited to powers of two), there is a better utilisation of the bit budget.

Tables 10 and 11 show the PSNR results for training images of both cases of underallocation and overallocation, respectively. It can be seen that a small increase in PSNR has been achieved through better utilisation of the bit budget when using fractional bits rather than with integer bits. The benefits of using fractional bits are more evident at low bitrates.

Table 8
Bit allocation table for cluster 1 at 0.15 bits/pixel

3	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Table 9
Level allocation table for cluster 1 at 0.15 bits/pixel

10	3	2	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

Table 10
PSNR performance of the fixed-rate (2 bits/pixel), 4 cluster GMM-based block quantiser using integer and fractional bit-based cluster block quantisers (underallocation) on images that were part of the training set

Image	PSNR (dB)	
	Integer	Fractional
Lena	38.32	38.38
Einstein	41.06	41.12
Jet	36.37	36.50
Goldhill	37.25	37.32

Table 11
PSNR performance of the fixed-rate (0.15 bits/pixel), 4 cluster GMM-based block quantiser using integer and fractional bit-based cluster block quantisers (overallocation) on images that were part of the training set

Image	PSNR (dB)	
	Integer	Fractional
Lena	24.81	25.50
Einstein	26.33	27.39
Jet	22.60	23.67
Goldhill	24.83	25.46

Table 12
PSNR performance of the fixed-rate (2 bits/pixel), 4 cluster GMM-based block quantiser using integer and fractional bit-based cluster block quantisers (underallocation) on images that were not part of the training set

Image	PSNR (dB)	
	Integer	Fractional
Boat	34.76	34.81
Kids	29.55	29.57
Crowd	26.96	27.02
Hat	37.38	37.44
Mill	30.48	30.52
Vegas	38.49	38.58

Table 13
PSNR performance of the fixed-rate (0.15 bits/pixel), 4 cluster GMM-based block quantiser using integer and fractional bit-based cluster block quantisers (overallocation) on images that were not part of the training set

Image	PSNR (dB)	
	Integer	Fractional
Boat	23.32	23.95
Kids	21.52	21.92
Crowd	17.09	17.16
Hat	24.08	24.60
Mill	20.61	20.71
Vegas	24.06	24.35

At 0.15 bits/pixel, the results for the images ‘Einstein’ and ‘jet’ show a 1 dB improvement in PSNR while other images gain about 0.6 dB which is more significant than those observed at 2 bits/pixel. Tables 12 and 13 show the PSNR results for images that were not part of the training set. As before, better utilisation of the bit budget has led to a slightly higher PSNR.

These performance improvements may be explained by Tables 14 and 15 which shows the effective bitrate compared with the target bitrate of each cluster block quantiser. In

Table 14
Effective bitrates for each cluster at 2 bits/pixel

Cluster	Target		Integer		Fractional	
	Bitrate	Total bits	Bitrate	Total bits	Bitrate	Total bits
1	1.943308	124.371726	1.9375	124	1.943224	124.366336
2	1.966590	125.861752	1.953125	125	1.966462	125.853568
3	1.991375	127.44803	1.984375	127	1.991121	127.431744
4	1.895964	121.341682	1.890625	121	1.895655	121.32192

Table 15
Effective bitrates for each cluster at 0.15 bits/pixel

Cluster	Target		Integer		Fractional	
	Bitrate	Total bits	Bitrate	Total bits	Bitrate	Total bits
1	0.093308	5.971726	0.078125	5	0.092295	5.90688
2	0.116590	7.461752	0.109375	7	0.116554	7.459456
3	0.141375	9.048003	0.140625	9	0.141375	9
4	0.045964	2.941682	0.03125	2	0.043865	2.80736

Table 15, the total fractional bits for cluster 1 is roughly 5.97 bits. For the integer bit-based cluster block quantiser, this total is truncated to 5 bits (32 levels), hence 0.97 bits are not utilised. While for the fractional bit-based cluster block quantiser, the total equivalent bits is approximately 5.9 bits (60 levels). The increase in the performance may also be due to the fact that the optimality of the constrained minimisation formula of (11) is better preserved in fractional bit encoding. For the integer bit case, fractional parts calculated from the formula are discarded and a heuristic algorithm is then used to compensate the truncated bits. The heuristic algorithm is itself dependent on various assumptions and approximations and may only produce a suboptimal solution. On the other hand, most of the fractions from (11) are preserved when converting from bits to the levels since the truncation occurs after the conversion. For example, consider a bitrate of 5.9 bits calculated from (11). In the integer bit case, this bitrate would truncated to 5 bits or 32 levels. In the fractional bit case, converting to levels gives $2^{5.9} = 59.714$ and after truncation results in 59 levels. By changing the point at which truncation is performed, an extra 27 quantiser levels are preserved. Therefore, it is expected that fractional bit-based coding allows us to get closer to the values specified by the Lagrangian-minimised formula and have less dependence on the use of heuristics and high resolution approximations.

9. Conclusion and further work

In this paper, the problem of fractional bitrates occurring in fixed-rate GMM-based block quantisers was defined. This type of quantiser consists of a set of individual block quantisers operating at different bitrates, each adapted to a Gaussian cluster. Side information in the form of the cluster number is inherent in the encoding via partitioning of the quantiser level range and this leads to a fractional number of bits being assigned to

each cluster block quantiser. Traditionally, scalar quantisers are designed to operate on an integer number of bits while block quantisers use a set of scalar quantisers to achieve an overall fractional bitrate. The bitrate can be achieved exactly only if the number of bits assigned to the block quantiser is an integer. In an m cluster GMM-based block quantiser, where there are m block quantisers operating, the number of bits assigned to each are most likely fractional.

The use of quantiser levels rather than bits lends itself to finer granularity which leads to better utilisation of the bit budget. Secondly, the finer granularity also allows a better 'spread' of the bit budget across all components. In transform-based image coding, this allows more of the bitrate to be allocated to the high frequency components which leads to better visual quality. And lastly, the fractional bits allocated are closer to those calculated from constrained minimisation since truncation is performed on fractional levels rather than on fractional bits, as well as less dependence on heuristic algorithms. Experimental results from a fixed-rate GMM-based block quantiser on images show that this concept translates to slightly better performance without an increase in the bitrate and complexity. The benefits of this method are most evident at low bitrates such as 0.15 bits/pixel, where gains of up to 1 dB were observed.

The encoding and transmission of quantiser levels from each component of the cluster block quantisers in the fixed-rate GMM-based block quantiser is not as straightforward as the encoding of quantiser bits though. Through a generalisation of positional value number systems, a simple method of encoding components with different numbers of levels into a single integer was proposed in this paper. It solves the problem of encoding and transmitting fractional bits in a fixed-rate coding framework.

The condition that the product of all levels must equal the total presents a complication in how heuristic algorithms are developed in comparison with those used when operating on bits. A simple and effective 'fixed-slope' algorithm was presented where the criteria is to find the component that influences the distortion most favourably while at the time satisfying the total level product constraint. However, it would appear that this algorithm works only soundly as high resolution approximations are inadequate for the bitrates under consideration and further work on a more effective bit allocation is expected to produce better gains.

References

[1] C. Archer, T.K. Leen, Optimal dimension reduction and transform coding with mixture principal components, in: Proceedings of International Joint Conference on Neural Networks, July 1999.

[2] C. Archer, T.K. Leen, From mixtures of nixtures to adaptive transform coding, in: T. Leen, T. Dietterich, T. Tresp (Eds.), Advances in Neural Information Processing Systems, vol. 13, MIT Press, Cambridge, MA, 2001, pp. 925–931.

[3] C. Archer, T.K. Leen, A generalised-Lloyd type algorithm for adaptive transform coding, IEEE Trans. Signal Process. 52 (1) (2004) 255–264.

[4] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. Royal Stat. Soc. 39 (1977) 1–38.

[5] R. Dony, S. Haykin, Optimally adaptive transform coding, IEEE Trans. Imag. Process. 4 (10) (1995) 1358–1370.

[6] M. Effros, P. Chou, R.M. Gray, Weighted universal image compression, IEEE Trans. Imag. Process. 8 (10) (1999) 1317–1328.
[7] W.R. Gardner, B.D. Rao, Theoretical analysis of the high-rate vector quantization of LPC parameters, IEEE Trans. Speech Audio Process. 3 (1995) 367–381.
[8] A. Gersho, R.M. Gray, Vector Quantization and Signal Compression, Kluwer Academic, Dordrecht/ Norwell, MA, 1992.
[9] V.K. Goyal, Theoretical foundations of transform coding, IEEE Signal Process. Mag. 18 (5) (2001).
[10] R.M. Gray, D.L. Neuhoff, Quantization, IEEE Trans. Inform. Theory 44 (6) (1998) 2325–2383.
[11] P. Hedelin, J. Skoglund, Vector quantization based on Gaussian mixture models, IEEE Trans. Speech Audio Process. 8 (4) (2000) 385–401.
[12] J.J.Y. Huang, P.M. Schultheiss, Block quantization of correlated Gaussian random variables, IEEE Trans. Commun. Syst. CS-11 (1963) 289–296.
[13] K.P. Kramer, M.V. Mathews, A linear coding for transmitting a set of correlated signals, IRE Trans. Inform. Theory IT-17 (1971) 751–752.
[14] Y. Linde, A. Buzo, R.M. Gray, An algorithm for vector quantizer design, IEEE Trans. Commun. 28 (1) (1980) 84–95.
[15] A. Ortega, M. Vetterli, Adaptive scalar quantization without side information, IEEE Trans. Imag. Process. 6 (1997) 665–676.
[16] S. So, K.K. Paliwal, Efficient block coding of images using Gaussian mixture models, in: Proc. Fourth Australasian Workshop on Signal Processing and Applications 2002, September 2002, pp. 71–74.
[17] J. Samuelsson, P. Hedelin, Recursive coding of spectrum parameters, IEEE Trans. Speech Audio Process. 9 (5) (2001) 492–503.
[18] A.D. Subramaniam, B.D. Rao, PDF optimized parametric vector quantization of speech line spectral frequencies, IEEE Trans. Speech Audio Process. 11 (2) (2003) 130–142.
[19] E.A. Riskin, Optimal bit allocation via the generalized BFOS algorithm, IEEE Trans. Inform. Theory 37 (2) (1991) 400–402.
[20] J.K. Su, R.M. Mersereau, Coding using Gaussian mixture and generalized Gaussian models, in: Proc. IEEE Int. Conf. Image Processing, Lausanne, Switzerland, 1996, pp. 217–220.
[21] R.J. Tocci, Digital Systems: Principles and Applications, sixth ed., Prentice Hall International, Englewood Cliffs, NJ, 1995, pp. 6–9.
[22] G.K. Wallace, The JPEG still picture compression standard, Commun. ACM 34 (4) (1991) 30–44.
[23] P.A. Wintz, Transform picture coding, Proc. IEEE 60 (7) (1972) 809–820.

Kuldip K. Paliwal was born in Aligarh, India, in 1952. He received the BS degree from Agra University, Agra, India, in 1969, the MS degree from Aligarh Muslim University, Aligarh, India, in 1971, and the PhD degree from Bombay University, Bombay, India, in 1978.

He has been carrying out research in the area of speech processing since 1972. He has worked at a number of organizations including Tata Institute of Fundamental Research, Bombay, India, Norwegian Institute of Technology, Trondheim, Norway, University of Keele, UK, AT&T Bell Laboratories, Murray Hill, NJ, USA, AT&T Shannon Laboratories, Florham Park, NJ, USA, and Advanced Telecommunication Research Laboratories, Kyoto, Japan. Since July 1993, he has been a professor at Griffith University, Brisbane, Australia, in the School of Microelectronic Engineering. His current research interests include speech recognition, speech coding, speaker recognition, speech enhancement, face recognition, image coding, pattern recognition, and artificial neural networks. He has published more than 250 papers in these research areas.

Dr. Paliwal is a Fellow of Acoustical Society of India. He has served the IEEE Signal Processing Society's Neural Networks Technical Committee as a founding member from 1991 to 1995 and the Speech Processing Technical Committee from 1999 to 2003. He was an associate editor of the IEEE Transactions on Speech and Audio Processing during the periods 1994–1997 and 2003–2004. He also served as associate editor of the IEEE Signal Processing Letters from 1997 to 2000. He was the general co-chair of the Tenth IEEE Workshop on Neural Networks for Signal Processing

(NNSP 2000). He has co-edited two books: *Speech Coding and Synthesis* (published by Elsevier), and *Speech and Speaker Recognition: Advanced Topics* (published by Kluwer). He has received IEEE Signal Processing Society's best (senior) paper award in 1995 for his paper on LPC quantization. He is currently serving the Speech Communication journal (published by Elsevier) as its editor-in-chief.

Stephen So was born in Hong Kong, in 1977. He received the B.Eng (Hons) degree in microelectronic engineering from Griffith University, Brisbane, in 1999. He is currently pursuing the PhD degree in the School of Microelectronic Engineering at Griffith University, Brisbane. His interests are in the areas of image coding, speech coding, audio coding, and distributed speech recognition.