ELSEVIER

Signal Processing: *Image Communication* 8 (1996) 551–562

SIGNAL PROCESSING: IMAGE COMMUNICATION

On the design of a hierarchical BTC-VQ compression system

Pasi Fränti *, Timo Kaukoranta [1], Olli Nevalainen

Department of Computer Science, University of Turku, Lemminkäisenkatu 14 A, FIN-20520 Turku, Finland

Received 16 February 1994

Abstract

In the present paper we study the use of vector quantization in the BTC-VQ image compression system. We propose an inverted order of proceeding in the BTC-VQ algorithm, so that the interaction of coding the bit-plane and the quantization data will be taken into consideration. The quality of the image depends radically on the codebook used in VQ. The use of frequencies in the selection of the initial codebook turns out to be superior to random selection.

Keywords: Image compression; Vector quantization; Quantization methods; Hierarchical decomposition; Codebook construction

1. Introduction

Block truncation coding (BTC) [2] is a simple way to compress *digital gray-scale images*. The basic idea in the method is to divide the pixel matrix into blocks and quantize the pixels to two different values, a and b. These along with a *bit-plane* (*bit-matrix*), indicating the choice between a and b, are transmitted as a compressed image.

A large number of variants of the basic BTC-algorithm have been proposed during the past 15 years [6]. Their performance can be compared using the *bit-rate* (bits per pixel), the *mean square error* (MSE) of the reconstructed image, and the *overall complexity* of the algorithm. Efficient variants of BTC typically achieve bit-rates of the order of magnitude 1.25 with MSE 32.5 for the test image Lena [5,13].

Several approaches have been proposed for reducing the volume of the bit-plane by the use of vector quantization (VQ) [3,13,17,19–21]. The basic idea of VQ is to select a small set of representative *vectors* (binary blocks) and to code all possible vectors by an index to this set.

In this paper we propose a new *hierarchical* BTC algorithm (HBTC-VQ). A hierarchy of blocks is considered in the algorithm and vector quantization is applied when the distortion caused by the VQ is expected to be small. We consider different design alternatives while constructing the codebook, performing best match search, and coding the quantization data. The new algorithm competes well in respect to bit-rate, MSE, and speed. With the bit-rate of 1.24, the algorithm gives MSE of 22.79 for Lena. For an extended summary of the present paper, see [4].

We start in Section 2 by shortly reviewing the previous works on BTC-VQ algorithms. The new algorithm is given in Section 3. Choices in the implementation; including the codebook generation, search method, re-

* Corresponding author. E-mail: franti@utu.fi.

[1] Timo Kaukoranta acknowledges support by the Academy of Finland under grant No. 18584.

PII S0923-5965(96)00008-2

vision of the VQ-organization (called inverted order VQ) and coding of the quantization data are discussed in Section 4. A summary of the test results is documented in Section 5, and finally conclusions are drawn in Section 6.

2. Background

The use of VQ for coding the bit-plane and the quantization levels was first introduced by Udpikar and Raina [17]. They apply vector quantization for coding both the bit-plane and the quantization data (a, b) separately. The *codebook* of the vector quantization is generated off-line on the basis of training vectors by the *generalized Lloyd algorithm* [7,12]. In the coding phase, the element to be coded is matched for each code vector in the codebook, and the one which minimizes the given distortion function is chosen.

Several improvements and modifications of the above scheme have been proposed in the literature. In the present paper we consider VQ only for coding the bit-plane, but compress the quantization levels (a, b) by other means.

Most of the previous works [3,17,19–21] rely on the generalized Lloyd algorithm as the codebook generation method, or select the codebook ad hoc [13]. The size of the codebook was 256 in [3,17,19,21] and 128 in [13], giving the bit-rate of the bit-plane 0.5 and 0.44, respectively. Several codebooks of different sizes were used by Weitzman and Mitchell [20]. The size depends on the contrast of a block (as measured by $b-a$): the greater the contrast the larger the codebook.

Nasiopoulos et al. [13] coded only blocks whose contrast is in a given range (not too large but not too small either) by VQ. The codebook was formed so that *edge blocks* are well represented. This also reduces the so-called staircase effect, which is otherwise impaired by VQ. A *classified* VQ algorithm was proposed by Efrati et al. [3] for avoiding the staircase effect. BTC-VQ is used in this method for the low contrast blocks, and a three-level quantizer is applied for the high contrast blocks. Thus, the algorithm uses two different codebooks, one for the low contrast and another for the high contrast blocks.

The representative for a bit-plane is usually chosen by a *full search*, i.e. by testing each candidate codevector and selecting the one which minimizes the given distortion function. A *localized search* was proposed by Weitzman and Mitchell [19]. They divide the codebook into several overlapping sub-codebooks. For each bit-plane, its *combination index* is calculated. It determines the sub-codebook in which the search is performed.

3. HBTC-VQ algorithm

It is a well-known fact that BTC performs poorly in the regions of high contrast. The problem can be attacked by using variable block sizes [10,14]. With large blocks, one can decrease their total number and therefore reduce the bit-rate. On the other hand, small blocks improve the image quality.

Here we consider rectangular raster images consisting of $X * Y$ pixels each represented by 8 bits. The image is segmented into blocks of size $m_1 * m_1$ $(m_1 = 2^n)$. If standard deviation σ of a block is less than a predefined threshold σ_{th}, the block is coded by a BTC-VQ algorithm. Otherwise, it is divided into four subblocks and the same process is repeated until the threshold criterion is met, or the minimal block size $(m_2 * m_2)$ is reached. The hierarchy of the blocks is represented by a *quadtree structure*.

A compressed block appears as a triple (a, b, B), where B stands for the bit-plane giving the quantization of the pixel values. The image is reconstructed at the decoding phase by assigning the value a to the 0-value pixels and b to the 1-value pixels of the bit-plane, where a and b are given by

$$a = \frac{1}{m-q} \sum_{x_i < \bar{x}} x_i, \tag{1}$$

$$b = \frac{1}{q} \sum_{x_i \geqslant \bar{x}} x_i. \tag{2}$$

Here x_i stands for a pixel value, $\bar{x}$ the average of their values, m is the total number of the pixels, and q is the number of 1-bits in the block. This selection of a and b is known as the *absolute moment* BTC (AMBTC) [11]. It preserves the *first moment* $(\bar{x})$ and the *first absolute central moment* $(\frac{1}{m} \sum |x_i - \bar{x}|)$ of the coded block, and is optimal in the MSE-sense in respect to the quantization threshold.

```
Main()
    WHILE unprocessed rows of the image exists DO
        Read the next m1 lines of the image into the buffer.
        Split the lines into nx=⌈X/m1⌉ blocks of m1*m1 pixels.
        FOR i:=1 to nx DO
            EncodeBlockByBTC(block_i).
    END-WHILE.

EncodeBlockByBTC(block)
    Calculate mean (x̄) and standard deviation (σ) of the pixels in block.
    IF σ<σth or the block size is m2*m2 THEN
        IF block size > m2*m2 THEN encode hierarchy bit 1 (=leaf node).
        Construct the bit plane B.
        IF block size is 2*2 THEN
            Output the bit plane.
        ELSE
            EncodeBitPlaneByVQ(B).
        Calculate (a,b) for the reconstructed bit plane.
        Quantize (a,b) to 6+6 bits.
        Compress (a,b) by FELICS.
    ELSE
        Encode hierarchy bit 0 (=internal node).
        Divide the block into four subblocks.
        Process the subblocks recursively in the same manner.

EncodeBitPlaneByVQ(B)
    IF block size is 4*4 THEN
        v := SearchNearestMatch(B).
        Output the index v of the codevector.
        Replace the bit plane B by the codevector Cv.
    ELSE
        Divide the bit plane into four subplanes.
        Encode the subplanes recursively.

SearchNearestMatch(B)
    w := Combination index [19] of the bit plane B.
    For each candidate codevector Cj in subbook w DO:
        Calculate (a,b) corresponding to Cj.
        Calculate the MSE for this part of the block when compressed with Cj.
        Select the one with smaller MSE as the best match.
    Return the index j∈[1,256] of the best match.
```

Fig. 1. The HBTC-VQ algorithm.

Variable block sizes cause problems in the VQ method. One can apply VQ to blocks of variable size as proposed in [7, 22], but we have not found in the literature a BTC-VQ scheme for this case. A simple and straightforward solution to the problem is proposed next.

In HBTC-VQ we apply vector quantization for the bit-planes of $4*4$-blocks only. Whenever a block is

larger than that, the *bit-plane* of the block (not the block itself!) is divided into four subplanes of equal sizes. This process is continued until a $4*4$ subplane is reached and VQ can be applied. No extra bits are needed to code the subplane hierarchy. If the block size is $2*2$ we simply disallow the use of VQ. This is argued by the fact that $2*2$-blocks usually lie in the high contrast regions, which are unfavorable for VQ.

The HBTC-VQ algorithm includes the following methods:

- Quadtree block decomposition of the image [10,14].
- Absolute moment BTC [11].
- Vector quantization [17] with localized search [19] and MSE criterion.
- FELICS [9] for coding of (a, b).

In the implementation, (a, b) are first quantized (by rounding) to 6+6 bits and then encoded by FELICS (*Fast and Efficient Lossless Image Compression System*). The parameter selections for the hierarchy are $m_1 = 32$, $m_2 = 2$, and the threshold values for the levels $32*32$, $16*16$ and $8*8$ is $\sigma_{\text{th}} = 6$, as proposed in [6]. For the $4*4$-level the threshold value is left as an adjusting parameter. The size of the VQ codebook is fixed to 256, thus the compression effect will be 0.5 bpp for every block coded by VQ. The main structure of the algorithm is given in Fig. 1.

4. Implementation aspects

Though the algorithm of the last section seems to be straightforward and clear, there remain a number of questions to be solved. We next analyze different design alternatives to find a good compromise between them. All the results are for the non-hierarchical BTC with $4*4$ block size, unless otherwise noted.

4.1. Vector quantization of the bit-plane

Traditionally, VQ is used for coding the bit-plane independently from the quantization level values (a, b), which are calculated on the basis of the original bit-plane. Now, if the representative of the bit-plane differs from the original, the quantization levels (a, b) do not anymore correspond to the mean values of the two partitions as they are assumed to, and thus they are no more optimal in the MSE-sense. We have therefore reorganized the BTC algorithm so that the bit-plane is constructed and coded by VQ *before* the pair (a, b) is calculated. The quantization levels (a, b) are then calculated according to the reconstructed bit-plane C_v, so that a corresponds to the mean value of the pixels belonging to the 0-partition and b the pixels belonging to the 1-partition. In the compressed file, the pair (a, b) can still appear before the bit-plane, since the inversion is invisible at the decoding phase. The bit-plane is thus omitted if a and b are equal.

In this perspective, vector quantization serves as an *indirect quantization method* of the block, not as a sole coding phase of the bit-plane. The effect of this is studied in more detail in Section 4.4. Until then, we assume the original order of processing in VQ.

4.2. Codebook generation

The codebook is constructed off-line on the basis of training vectors by using a suitable algorithm. Then the same codebook is used for whatever images are to be coded. Adaptive vector quantization [8] was considered in [21] but the results of their investigations indicate that the non-adaptive, *universal codebook* VQ should be preferred because of the speed.

We use the *Generalized Lloyd algorithm* (GLA) [7,12] for generating the codebook. The method needs an initial codebook which is then improved by Lloyd's iteration algorithm. This gives a locally optimal codebook in respect to the training set and the initial codebook. We select the initial codebook heuristically as the m most frequent matrices of the training set. Another simple method would have been to select the matrices randomly from the training set [3,17]. However, the results of the frequency-based method are clearly superior, see Table 1. The codebooks in GLA are here iterated until no improvement is achieved. The number of iterations, corresponding to the codebook sizes of (128, 256, 512), were (3, 3, 2) for the case of random heuristic, and (2, 1, 2) for the case of frequency heuristic. The small number of iterations (originating from the binary source) emphasize the importance of the choice of the initial codebook. The training set consists of four images (different from the test images of Table 1) with in total 23 944 different matrices.

Table 1
Performance comparison (in MSE) of the heuristics for constructing the initial codebook in GLA. Full search is applied over the codebooks using MSE as a distortion function

Heuristics	Random			Frequency		
Codebook size	128	256	512	128	256	512
Airplane	84.67	70.49	64.07	50.02	45.78	43.46
Bridge	230.23	190.27	162.61	158.13	143.44	131.75
Galaxy	1.60	1.38	1.29	1.42	1.34	1.20
Lena	78.17	64.62	61.59	53.42	49.99	47.46
New Orleans	32.27	27.47	25.28	22.70	20.77	19.36
X-ray	47.33	22.94	22.02	19.63	18.71	18.09

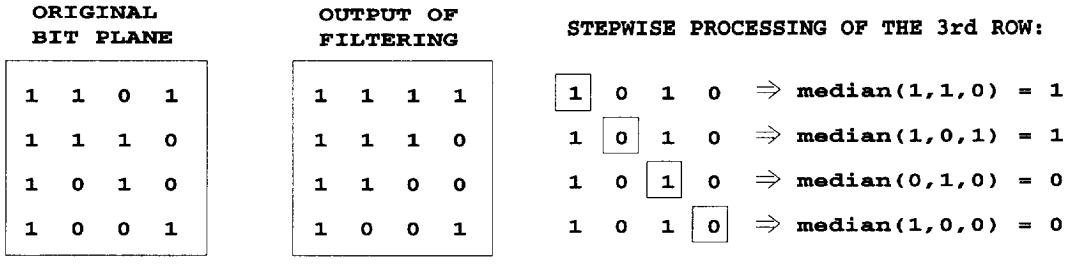


Fig. 2. Illustration of horizontal 1D-median filtering with a window of three pixels.

Another approach to bit-plane compression was proposed by Arce and Gallagher [1]. They apply filtering techniques, which are in a sense related to vector quantization in this context. *Median filtering* [15] is a signal processing technique where the input signal is processed by replacing each input value with the *median* of the original values within a given window. The signal can be filtered over and over again until it reaches its *roots*, i.e. it is not affected by the filtering anymore. The *root signal set* includes all the signals that are roots. The root signal set can be considered as a codebook of a vector quantization method, where the number of roots determines the size of the codebook.

In the comparison we included three filtering techniques. These are the *1D-median filtering* (1D-MF) [1], *separable median filtering* (SMF) [18], and *cross median filtering* (CMF) [18]. The codebooks of these methods were selected as subsets of their root signal sets, so that the size of the particular codebooks are close to exact powers of two; that is (2030, 4096, 8112) for (SMF, 1D-MF, CMF). For details see [1,18].

In the 1D-MF, the window was set to three pixels including the current pixel and its two neighboring pixels in the same row. For the border pixels, the outer ones are assumed to be of the same intensity as the border pixel, see Fig. 2. SMF is a two-stage procedure where filtering with a three pixel window is first applied in horizontal direction and then in vertical direction. In CMF a symmetrical two-dimensional window is used and only one stage is needed for one filtering pass.

Table 2 gives a comparison of several methods for generating the codebook. Here we apply full search over the codebooks using MSE as a distortion function. ACC (*Adaptive Compression Coding*) refers to the codebook proposed in [13]. It has been primarily designed for edge blocks only, but it is rather good for all blocks in the MSE-sense. It is observed that the size is the most important factor in the selection of the codebook: the larger the codebook, the smaller is the distortion in MSE. The choice of a codebook building method turns out to be of minor importance. In most cases GLA is slightly better when compared to the filtering methods (SMF, 1D-MF, CMF), but the difference is small.

Table 2
Performance comparison (in MSE) of methods for generating the codebook. Full search is applied over the codebooks using MSE as a distortion function

Method Codebook size	GLA 128	GLA 256	GLA 512	GLA 2048	GLA 4096	GLA 8192
Airplane	50.02	45.78	43.46	40.96	40.30	39.81
Bridge	158.13	143.44	131.75	114.40	107.85	103.16
Galaxy	1.42	1.34	1.20	1.05	0.99	0.94
Lena	53.42	49.99	47.46	42.63	41.22	40.29
New Orleans	22.70	20.77	19.36	17.27	16.72	16.30
X-ray	19.63	18.71	18.09	16.90	16.41	16.03

Method Codebook size	ACC 128	SMF 2030	1D-MF 4096	CMF 8112	AMBTC —	Optimal 65536
Airplane	53.58	41.03	42.65	39.91	41.32	34.51
Bridge	162.34	118.17	116.71	105.26	99.37	87.67
Galaxy	1.43	1.14	1.08	1.03	0.86	0.82
Lena	57.03	42.39	44.66	40.37	40.51	35.54
New Orleans	24.08	16.89	17.61	16.22	16.44	14.21
X-ray	22.27	18.39	18.11	17.59	15.57	14.69

GLA = Generalized Lloyd algorithm [7,12].
ACC = Codebook proposed in [13].
SMF = Separable median filtering codebook [18].
CMF = Cross median filtering codebook [18].
AMBTC = Absolute moment BTC [11,16].
Optimal = Optimal quantization.

An interesting observation is that if the codebook is large enough, BTC-VQ may outperform BTC without any vector quantization (cf. the results for Airplane). This rather surprising effect originates from the non-optimal quantization of AMBTC (in the MSE-sense). Therefore, it is possible to find a better quantization matrix from a large codebook than to quantize the block according to the mean value of the block. The optimal quantization, on the other hand, can be found by testing all the pixel values in the block as a candidate threshold, and by choosing the one that gives the lowest MSE-value.

4.3. Search method

We consider three alternative search methods:
- Full search with MSE distance.
- Localized search with MSE distance.
- *Look-up table* (LUT) implementation using Hamming distance (or filtering).

The full search is always optimal (in respect of the distortion function) but it is somewhat impractical for large codebooks. In HBTC-VQ we thus use the localized search as proposed in [19]. Here the codebook is divided into several (overlapping) sub-codebooks. For each bit-plane, its so-called *combination index* is calculated. This index determines the codebook in which the search is performed.

The performance of a search method depends not only on the organization of the codebook, but also on the distortion function. In HBTC-VQ we use MSE, which is the most obvious measure. However, it cannot be calculated independently from the quantization data (a, b). Most of the previous works have thus used Hamming distance as a distortion measure [13,17,21]. If there are two or more codevectors with the same Hamming distance from the original bit-vector, the one with the same number of 1-bits is chosen [21].

The advantage of using Hamming distance is that it can be implemented by *look-up table* (LUT). In fact, this is possible whenever the distortion function can be calculated independently from (a, b). The

Table 3
Performance comparison (in MSE) of look-up table methods

Codebook Search/dist.	SMF (2030)		1D-MF (4096)		CMF (8112)	
	Hamming	Filtering	Hamming	Filtering	Hamming	Filtering
Airplane	45.28	45.85	47.87	47.29	43.40	42.85
Bridge	139.45	154.98	137.22	138.95	118.49	121.15
Galaxy	1.31	1.51	1.25	1.32	1.15	1.30
Lena	46.79	49.30	50.81	51.65	43.59	44.35
New Orleans	18.63	19.50	19.94	19.96	17.62	17.52
X-ray	20.39	22.24	20.61	21.47	19.61	21.35

criterion is also met in the filtering schemes. Filtering can be considered as a search technique, where the result (root) corresponds to the reconstructed bit-plane from the codebook (i.e. root signal set). A drawback of the filtering techniques is that they only produce codevectors in a root signal set, whereas the Hamming distance can be calculated between any two codevectors, no matter what the codebook is.

We compared the use of Hamming distance and filtering for different codebooks, see Table 3. The codebooks were selected corresponding to the filtering methods, and for each codebook, both search methods were applied. The results favor Hamming distance for two reasons: it gives slightly better MSE-values, and does not restrict the choice of the codebook as the filtering methods do.

Finally, the three different search methods are compared in Table 4. MSE is used as a distortion measure both in the full and localized search, and Hamming distance is used in the *LUT-search*. The results show that full search is superior to the LUT-method. However, if the encoding time is critical, LUT-search is a proper choice. Localized search offers a suitable compromise between the speed and MSE, and we thus prefer it as the primary search method. Note that the decoding phase is independent from the search method in the coding, and thus it is always fast.

Table 4
Performance comparison (in MSE) of the search methods. The codebook has been determined by GLA and contains 256 codevectors

Search method	Full	Localized	LUT
Airplane	45.78	45.96	50.33
Bridge	143.44	145.33	164.11
Galaxy	1.34	1.35	1.49
Lena	49.99	50.31	55.16
New Orleans	20.77	20.89	23.07
X-ray	18.71	18.82	20.20

4.4. Inverted order VQ

In the HBTC-VQ, the values of (a, b) must be known at the moment of vector quantization, since MSE is used as a distance measure. Unfortunately, they are not yet known because of the *inverted order* of proceeding. Therefore, it is necessary to determine temporary (a, b) for each candidate vector (given by the search method). Value a is calculated as the mean value of the pixels assigned by the 0-bits, and b as the mean value of the pixels assigned by the 1-bits. (Note that $\bar{x}$ is not used in this process.) In this way the VQ phase can be optimized in the MSE-sense. A side effect is that for some candidate vector B, the value of a can become greater than b, see Fig. 3. In the HBTC-VQ algorithm this is prevented by disallowing the use of such candidate vectors at a time.

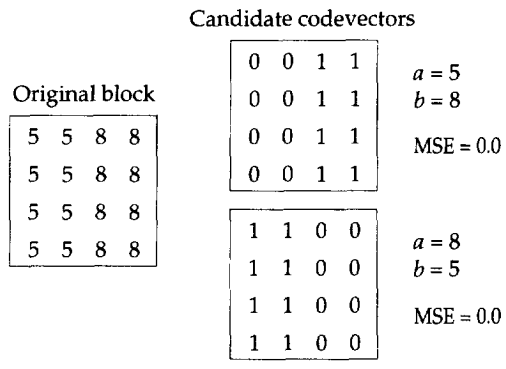


Fig. 3. Example of candidate codevectors.

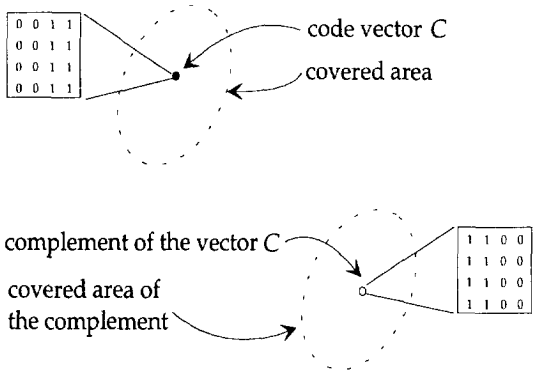


Fig. 4. Example of the dipolarity.

Table 5
Performance comparison (in MSE) of the inverted order VQ. The codebook has been determined by GLA and contains 256 entries. Full search is applied over the codebooks using MSE as a distortion function

Order	Original VQ	Inverted order VQ	
Dipolarity	Non-dipolar	Non-dipolar	Dipolar
Airplane	45.78	39.99	37.70
Bridge	143.44	124.34	113.82
Galaxy	1.34	1.23	1.13
Lena	49.99	44.11	41.52
New Orleans	20.77	18.00	16.75
X-ray	18.71	17.32	16.75

A *dipolar HBTC-VQ* is accomplished if no restriction is made about (a, b). The benefit is that more vectors in the codebook can be utilized at a time. For example, the second candidate vector of Fig. 3 is applicable as well as the first one. In fact, these codevectors are complements of each other, and therefore one of them can be eliminated from the codebook (or replaced by another). A drawback of the method is the decrease in the intra- and interblock correlation, which makes the coding of (a, b) less efficient, see Section 4.5.

For fully exploiting the dipolar algorithm, the codebook generation method (GLA) should be revised so that the distance measure (Hamming distance), and the centroid computation take care of the complements. Define the new distance measure by

$$d^* = \min[d(B, C), d(B, \bar{C})], \tag{3}$$

where $\bar{C}$ is the complement of the codevector C, and d is the original distance function. Each vector in the training set is now mapped to the closest original codevector, or to the closest complement. By this change, the covered area of the codevectors can be doubled in the best case, see Fig. 4.

The centroid of a partition is the *average* of the training vectors within the partition, with one exception: all vectors B that were connected to the complement $\bar{C}$ of the codevector C, must be inverted to their complement $\bar{B}$ before including in the calculation. The codevectors are thus *dipolar*.

Performance comparison of the inverted order VQ against the original order VQ is given in Table 5. The same codebook (GLA/256) has been applied for each case except for the dipolar VQ, which uses a dipolar codebook of the same size. The results for the inverted order methods are significantly better than those for the original VQ. The gain due to dipolarity is also significant. The effect of the dipolar vector quantization on the overall compression system is examined in more detail in Sections 4.5 and in 5.

4.5. Coding the quantization data

Two *subsample images* are formed, one from the a-values and another from the b-values of the blocks. In [5] the subsample images were coded by *arithmetic coding* with a suitable prediction and context models. The arithmetic coding, however, makes the compression system rather complex. In HBTC-VQ we apply FELICS coding [9] because of its simple implementation and speed.

FELICS coding uses the information of two adjacent pixels when coding the current one. These are the one to the left of the current pixel, and the one above

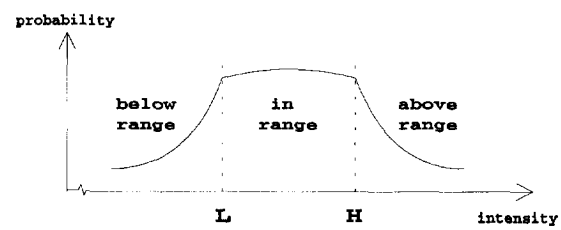


Fig. 5. Probability distribution of intensity values.

Table 6
Bit-rate comparison of different coding methods. The results show the number of bits per (a, b). Note that the pair (a, b) is quantized to 6+6 bits before compression, and thus the volume of the source data is 12 bits

VQ Coding method	Non-dipolar		Dipolar	
	EC	FELICS	EC	FELICS
Airplane	6.43	6.50	7.52	7.15
Bridge	8.48	8.67	10.26	9.97
Galaxy	3.18	4.34	3.65	4.50
Lena	6.75	7.01	8.29	7.89
New Orleans	6.50	6.37	7.82	7.25
X-ray	5.71	6.32	7.02	6.82

it. Denote the values of the neighboring pixels by L and H so that L is the smaller of the two. Howard and Vitter [9] have found that the probability of *individual pixels* obeys a distribution of the type given in Fig. 5. Our experiments indicate that the same distribution holds very well for the pixels of the *subsample images*, too.

The coding scheme is as follows. A code bit indicates whether the actual value falls into the in-range. If so, an *adjusted binary coding* is applied. Here the hypothesis is that the in-range values are uniformly distributed. Otherwise, the above/below-range decision requires another code bit, and the value is then coded by *Rice coding* with adaptive k-parameter selection.

Table 6 gives a performance comparison of the entropy coding scheme (EC) and FELICS, both in the case of dipolar and non-dipolar VQ. In FELICS, the intrablock correlation between a and b is not properly taken into consideration. In spite of this, the results are still surprisingly close to EC, and sometimes even better. Dipolar VQ decreases both the intra- and interblock correlation, and the coding of (a, b) becomes less efficient. This affects FELICS in lesser extent because it does not rely on the intrablock correlations. The usefulness of the dipolarity depends overall on whether the gain in bit-plane compression outweights the loss in the coding of (a, b), see Section 5.

5. Test results

Two versions (HBTC-VQ and dipolar HBTC-VQ) of the new algorithm are experimented here, with the parameter selections of Table 7. The performance of BTC (Block Truncation Coding [2]), ACC (Adaptive Compression Coding [13]), HBTCE (Hierarchical Block Truncation Coding with Entropy coding [5]), and the HBTC-VQ (the method of this paper) is profiled in Table 8 and Fig. 6. Different bit-rates can be achieved by changing the hierarchy threshold parameter σ_{th} at the $4*4$-level. The results of Table 8 are for Lena and Airplane. For our other test images the results were similar.

The results of HBTC-VQ compare favorable with the other BTC variants, cf. for the bit-rate 1.25 the MSE-values of (ACC, HBTCE, HBTC-VQ) were (32.5, 32.3, 22.8). The original BTC is ineffective because of the lack of hierarchy, and because it omits the compression

Table 7
Parameters of the HBTC-VQ algorithms

	HBTC-VQ	Dipolar HBTC-VQ
VQ codebook	GLA/256	Dipolar GLA/256
Search method	Localized, with MSE	Localized, with MSE
Order of process	Inverted	Inverted
Allow $a > b$?	NO	YES
Coding of (a, b)	FELICS	FELICS
Block sizes	$32 \rightarrow 2$	$32 \rightarrow 2$

Table 8
Performance comparison of several BTC-variants

Method	LENA		AIRPLANE	
	Bit-rate	MSE	Bit-rate	MSE
BTC	2.00	44.76	2.00	44.46
ACC	1.36	29.90	1.21	33.06
ACC	1.25	32.50	1.14	34.81
ACC	0.76	82.80	0.81	49.56
HBTCE	1.74	14.40	1.49	11.99
HBTCE	1.50	19.28	1.26	17.18
HBTCE	1.25	32.32	1.01	39.18
HBTC-VQ	1.75	16.01	1.51	13.41
HBTC-VQ	1.52	17.96	1.24	15.46
HBTC-VQ	1.24	22.79	1.00	22.62
HBTC-VQ	1.00	33.82	0.75	45.81

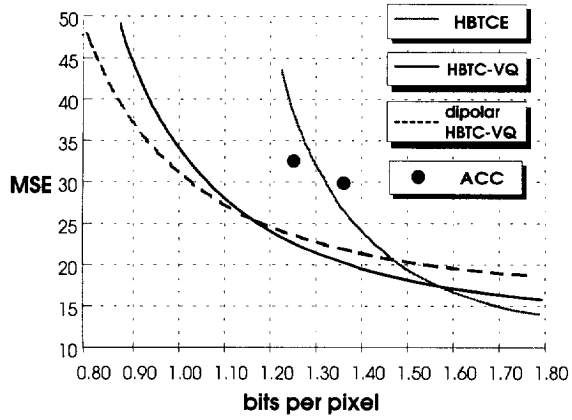


Fig. 6. MSE as a function of bit-rate for Lena.

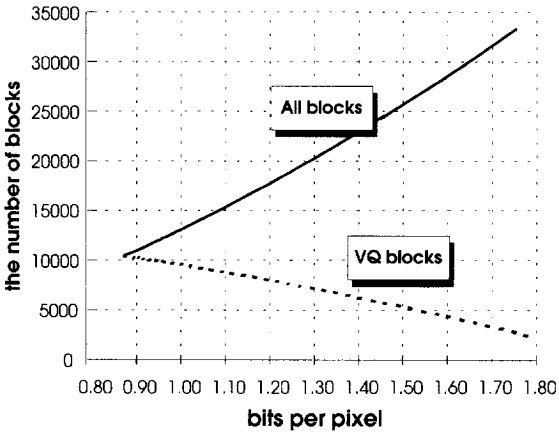


Fig. 7. The number of blocks in Lena for different bit-rates.

of (a, b) and the bit-plane. The hierarchical decomposition is advantageous since it allows adaptation to the variations in the image, with only a small penalty in the bit-rate. Moreover, for a high quality of the compressed image the use of $2*2$-blocks is essential in the high contrast regions.

HBTCE is a hierarchical BTC variant, in which the pair (a, b) and the bit-plane are compressed by entropy coding. A drawback of the method is that very low bit-rates cannot be achieved by it. This originates from the entropy coding applied for the bit-plane compression.

In dipolar HBTC-VQ benefit is gained from an *effectively* greater codebook, which is applied frequently at the lower bit-rates, see Fig. 7. On the other hand, the efficiency of FELICS decreases in the dipolar algorithm due to the increased variation of a and b. This effect is greater at the higher levels of the bit-rate, because the total number of blocks is greater, and thus there are more (a, b) pairs to be coded. The number of different blocks of size ($2*2$, $4*4$, $8*8$, $16*16$, $32*32$) in Lena is (3608, 8546, 794, 139, 24) at the bit-rate of 1.0 bpp in the non-dipolar case.

6. Concluding remarks

A new hierarchical BTC-VQ algorithm was proposed. Variable block sizes complicate the use of VQ, but a straightforward solution was proposed. The order of VQ was inverted so that the quantization levels are calculated corresponding to the reconstructed bit-plane. The most frequent bit-planes were selected to the initial codebook in order to improve the quality of the codebook generated by the GLA algorithm.

The full search of VQ was replaced by the localized search which is a compromise between speed and quality: the average number of candidate vectors was ca $\frac{1}{4}$ in comparison to the full search. Quantization data (a, b) was coded fast and efficiently by FELICS.

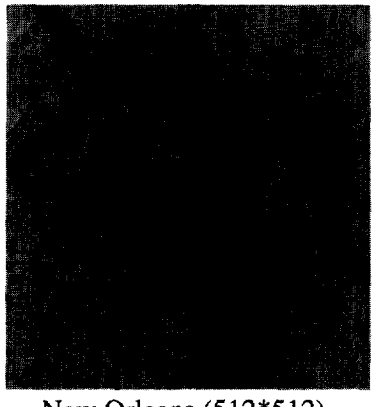
Note that the search of VQ was optimized only in the case of $4*4$ blocks because of simplicity. For a bit-plane of a $4*4$ block there are only 2^{16} possible combinations. On the other hand, in each $8*8$ block there are 4 bit-planes of the size $4*4$, and thus the number of all possible combinations is 2^{64}, which is clearly impractical.

The greatest deficiency of the proposed algorithm is that the method cannot adapt from image to image, like adaptive VQ. This emphasizes the importance of a proper selection of the training set and the initial heuristic. The problem of the universal codebook VQ approach should be studied further.

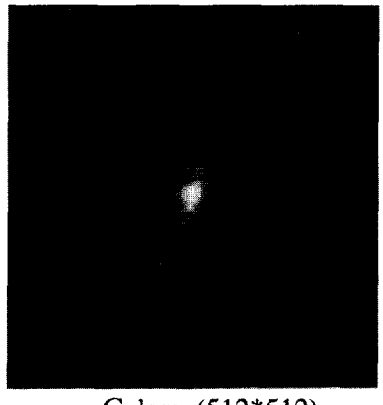
Appendix A. Test images

Airplane (512*512)

Bridge (256*256)

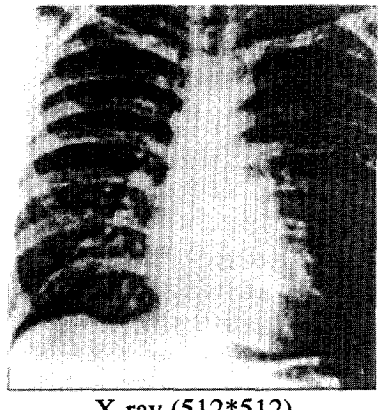


New Orleans (512*512)



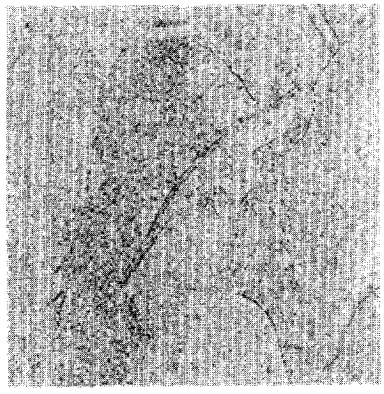
Galaxy (512*512)

Lena (512*512)



X-ray (512*512)

Reconstructed Lena
HBTC-VQ with 1.24 bpp



Error picture of Lena
HBCT-VQ with 1.24 bpp

References

[1] G. Arce and N.C. Gallagher, Jr., "BTC image coding using median filter roots", *IEEE Trans. Commun.* Vol. 31, No. 6, June 1983, pp. 784–793.

[2] E.J. Delp and O.R. Mitchell, "Image coding using block truncation coding", *IEEE Trans. Commun.*, Vol. 27, No. 9, September 1979, pp. 1335–1342.

[3] N. Efrati, H. Liciztin and H.B. Mitchell, "Classified block truncation coding-vector quantization: An edge sensitive image compression algorithm", *Signal Processing: Image Communication*, Vol. 3, Nos. 2–3, June1991, pp. 275–283.

[4] P. Fränti, T. Kaukoranta and O. Nevalainen, "A new approach to BTC-VQ image compression system", *Proc. Picture Coding Symp.*, Sacramento, 1994.

[5] P. Fränti and O. Nevalainen, "Block truncation coding with entropy coding", *IEEE Trans. Commun.*, Vol. 43, Nos. 1–2, 1995, pp. 1677–1685.

[6] P. Fränti, O. Nevalainen and T. Kaukoranta, "Compression of digital images by block truncation coding: A survey", *The Computer J.*, Vol. 37, No. 4, 1994, pp. 308–332.

[7] A. Gersho and R.M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, Dordrecht, 1992.

[8] M. Goldberg, P.R. Bouncher and S. Shlien, "Image compression using adaptive vector quantization", *IEEE Trans. Commun.*, Vol. 34, No. 2, February 1986, pp. 180–187.

[9] P.G. Howard and J.S. Vitter, "Fast and efficient lossless image compression", *IEEE Proc. Data Compression Conf.*, Snowbird, Utah, 1993, pp. 351–360.

[10] M. Kamel, C. Sun and L. Guan, "Image compression by variable block truncation coding with optimal threshold, *IEEE Trans. Signal Process.*, Vol. 39, No. 1, January 1991, pp. 208–212.

[11] M.D. Lema and O.R. Mitchell, "Absolute moment block truncation coding and its application to color images", *IEEE Trans. Commun.*, Vol. 32, No. 10, October 1984, pp. 1148–1157.

[12] Y. Linde, A. Buzo and R.M. Gray, "An algorithm for vector qauntizer design", *IEEE Trans. Commun.*, Vol. 28, No. 1, January 1980, pp. 84–95.

[13] P. Nasiopoulos, R. Ward and D. Morse, "Adaptive compression coding", *IEEE Trans. Commun.*, Vol. 39, No. 8, August 1991, pp. 1245–1254.

[14] J. Roy and M. Nasrabadi, "Hierarchical block truncation coding", *Opt. Engrg.*, Vol. 30, No. 5, May 1991, pp. 551–556.

[15] J.W. Tukey, *Exploratory Data Analysis*, Addison-Wesley, Reading, MA, 1971.

[16] V. Udpikar and J. Raina, "Modified algorithm for the block truncation coding of monochrome images", *Electronics Lett.*, Vol. 21, No. 20, September 1985, pp. 900–902.

[17] V. Udpikar and J. Raina, "BTC image coding using vector quantization", *IEEE Trans. Commun.*, Vol. 35, No. 10, October 1987, pp. 352–356.

[18] Q. Wang and Y. Neuvo, "Deterministic properties of separable and cross median filters with an application to block truncation coding", *Multidimensional Systems Signal Process.*, Vol. 4, No. 1, January 1993, pp. 23–38.

[19] A. Weitzman and H.B. Mitchell, "Fast multi-stage VQ search for the BTC–VQ algorithm", *Proc. 17th Convention of Electrical and Electronics Engineers in Israel*, Tel Aviv, Israel, 1991, pp. 182–185.

[20] A. Weitzman and H.B. Mitchell, "An adaptive BTC–VQ image compression algorithm operating at a fixed bit-rate", *Signal Processing: Image Communication*, Vol. 5, No. 4, October 1993, pp. 287–294.

[21] Y. Wu and D.C. Coll, "BTC-VQ-DCT hybrid coding of digital images", *IEEE Trans. Commun.*, Vol. 39, No. 9, September 1991, pp. 1283–1287.

[22] P. Yu and A.N. Venetsanopoulos, "Hierarchical multirate vector quantization for image coding", *Signal Processing: Image Communication*, Vol. 4, No. 6, November 1992, pp. 497–505.