Compression of Digital Images by Block Truncation Coding: A Survey

© *The Computer Journal*, 37 (4), 308-332, 1994

Pasi Fränti, Olli Nevalainen and Timo Kaukoranta
Department of Computer Science, University of Turku
Lemminkäisenkatu 14 A, FIN-20520 Turku, FINLAND
E-mail: franti@utu.fi

Abstract:

Block truncation coding is a lossy moment preserving quantization method for compressing digital gray-level images. Its advantages are simplicity, fault tolerance, the relatively high compression efficiency and good image quality of the decoded image. Several improvements of the basic method have been recently proposed in the literature. In this survey we will study the basic algorithm and its improvements by dividing it into three separate tasks: performing quantization, coding the quantization data, and coding the bit plane. Each phase of the algorithm will be analyzed separately. On the basis of the analysis, a combined BTC algorithm will be proposed, and comparisons to the standard JPEG algorithm will be made.

Index terms: image compression, quantization methods, lossy compression techniques.

1. INTRODUCTION

Block truncation coding (BTC) is a simple and fast *lossy* compression technique for digitized gray scale images originally introduced by Delp and Mitchell [10]. The key idea of BTC is to perform *moment preserving* (MP) quantization for blocks of pixels so that the quality of the image will remain acceptable and at the same time the demand for the storage space will decrease. Even if the compression gain of the algorithm is inferior to the standard JPEG compression algorithm [46], BTC has gained popularity due to its practical usefulness. Several improvements of the basic method have been recently proposed in the literature.

In this survey we study BTC and its improvements by dividing the algorithm into three separate tasks: performing the quantization of a block, coding the quantization data, and coding the bit plane. Each phase of the algorithm is analyzed separately by first introducing various improvements presented in the literature, including some new ideas of our own, and then comparing the performance of them against each other. On the basis of the analysis, a new combined BTC algorithm is proposed and compared to the JPEG algorithm.

We start in Section 2 by recalling factors characterizing the usefulness of compression techniques. Then the original BTC method are briefly described in Section 3. Variants of the basic algorithm is studied in Section 4 in the following way.

Several alternatives for the quantization method are presented in Section 4.1 [10, 14, 19, 23, 31, 47]. These methods try to either preserve certain moments [10, 23, 31], or minimize a fidelity criterion which is usually *mean square error* (MSE) or *mean absolute error* (MAE). They can also aim at the same goal by using a heuristic algorithm [14, 19] or by performing indirect quantization via a neural network [47].

The quantization data of a block can be expressed either by two statistical values (usually mean and standard deviation), or by the quantization level values themselves. For these two approaches, several coding methods are discussed in Section 4.2. One can significantly reduce the amount of data at the cost of image quality by using vector quantization [61] or discrete cosine transform [73]. Remarkable savings are also possible via entropy coding [16, 34] without any loss in the quality. The application of any other *lossless* image coding method is also possible, and thus we propose the use of FELICS coding [25] the implementation and testing of which is described in this paper.

In Section 4.3, we review methods for reducing the storage size of the bit plane. Most of these select a representative for the bit plane from a smaller subset, which is either a *root signal set* of some filtering technique [3, 65, 66], a codebook of a vector quantization method [14, 61, 73], or designed ad hoc [41]. A different approach is the use of interpolation [76], or entropy coding [16].

BTC can also be applied hierarchically using variable sized blocks [27, 54]. With a large block size one can decrease the total number of blocks and therefore reduce the bit rate. On the other hand, small blocks of the hierarchy improve the image quality. The technique is studied in Section 5.

Several extensions of BTC are discussed in Section 6 including 3-level quantization and generalization to *n*-level quantization [12, 19, 35, 53, 56]. Hybrid formulations of BTC are considered [10, 11] as well as some pre- and post processing techniques [37, 41]. BTC's application to color [31, 74] and video images [24, 54] are shortly discussed. Some block to block adaptive approaches [14, 22, 37, 41] are discussed in Section 7.

In Section 8, we evaluate the different methods experimentally, starting with the quantization methods in Section 8.1. We will observe, that because of unavoidable rounding errors, the moment preserving quantization does not preserve the first moment any better than the other tested quantization methods. It turns out that the moment preserving quantization is the worst in the MSE-sense, and that a simple and fast heuristic algorithm produces high quality decompressed images with only 5 % deficiency from the MSE-optimal quantization.

Some aspects of the coding of the quantization data are studied in Section 8.2. FELICS turns out to be a very good method for its efficiency and practical usefulness. Several bit plane coding methods are studied experimentally in Section 8.3. Among these, interpolation gives the best results in the MSE-sense. Hierarchical block decomposition is studied in Section 8.4. The choice for the minimum block size of hierarchy turns out to be the most important factor when small MSE-values are desired.

The need for block to block adaptive schemes is analyzed in Section 8.5. For typical images, more than 50 % of all MSE originates from the 10 % of the high variance blocks. Even so,

there was no evidence that the superiority between any two given bit plane coding methods was dependent on the variance of a block.

In Section 8.6 we link together the most promising methods to form an efficient BTC algorithm. With this we perform several test runs and make comparisons with the standard JPEG algorithm. The overall performance of BTC turns out to be weaker than that of JPEG, but because of the high speed (especially at the decoding phase) and simplicity, BTC is useful in practical implementations.

Notations:

a, b	Quantization levels.
$\hat{a}$, $\hat{b}$	Predictions of the quantization levels.
B	Bit plane of a block.
bpp	Bits per pixel.
$e_{i,j}$	Difference between the original and predicted pixel value.
k	Number of bits assigned to a single pixel.
m	Number of pixels in a block.
MAE	Mean absolute error of the reconstructed image.
MSE	Mean square error of the reconstructed image.
N	Number of pixels in an image.
q	Number of pixels in a block whose intensity is greater than or equal to x_{th}.
SNR	Signal to Noise Ratio.
v_{th}	Variance threshold.
x_i	Pixel value of the original image.
$\hat{x}_i$	Predicted pixel value of x_i.
x_{max}	Maximum pixel value.
x_{min}	Minimum pixel value.
x_{th}	Quantization threshold.
$\overline{x}$	Mean of x_i-values.
$\overline{x^r}$	Mean of ${x_i}^r$-values.
$\overline{x}_l$	Lower mean; average of $x_i < x_{th}$.
$\overline{x}_h$	Higher mean; average of $x_i \geq x_{th}$.
y_i	Pixel value of the reconstructed (decompressed) image.
α	Sample first absolute central moment.
σ	Sample standard deviation.
σ'	Standard deviation according to x_{th}.

2. BACKGROUND

Let us consider a digital gray-level still image of the size N pixels where each pixel is expressed by k bits. The aim of the image compression is to transform the image to a space efficient (compressed) form so that the information content is preserved as far as possible when decompressing the encoded image.

The quality of a compression method can be characterized by considering a set of features which describe the usefulness of the method. These features include the *bit rate*, which gives the average number of bits per stored pixel of the image. Bit rate is the principal parameter of a compression technique because it measures the efficiency of the technique.

Another feature is the ability to *preserve the information content*. A compression technique is *lossless* if the decoded image is exactly the same as the original one. Otherwise the technique is *lossy*. Most of the image compression techniques are of the latter type. The *quality* of the reconstructed image can be measured by the *mean square error* (MSE), *mean absolute error* (MAE), *signal to noise ratio* (SNR), or it can be analyzed by a human photo analyst. Let x_i stand for the i'th pixel value in the original image and y_i the corresponding pixel in the reconstructed image (i=1,2,...,N). Here the row-major order of pixels of the image matrix is supposed. The three quantitative measures are defined as

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N} \left[y_i - x_i \right]^2 \tag{1}$$

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N} \left| y_i - x_i \right| \tag{2}$$

$$\mathrm{SNR} = -10 \cdot \log_{10}\left[\mathrm{MSE} / 255^2 \right] \tag{3}$$

These parameters are widely used but unfortunately they do not always coincide with the evaluations of an human expert [36]. For a survey of different quality measures, see Eskicioglu and Fisher [15].

The third feature of importance is the *processing speed* of the compression and decompression algorithms. In on-line applications the response times are often critical factors. In the extreme case a space efficient compression algorithm is useless if its processing time causes an intolerable delay in the image processing application. In some cases one can tolerate longer compression time if the compression operation can be done as a background task. However, fast decompression is desirable.

The fourth feature of importance is *robustness* against data *transmission errors*. The compressed file is normally an object of a data transmission operation. The transmission is in the simplest form between internal memory and secondary storage but it can as well be between two remote sites via transmission lines. Visual comparison of the original and a disturbed images gives a direct evaluation of the robustness. However, the data transmission systems commonly contain fault tolerant internal data formats so that this property is not always obligatory.

Among other interesting features of the compression techniques we may mention the *ease of implementation* and the *demand for a working storage space*. Nowadays these factors are often of secondary importance. From the practical point of view the last but often not least feature is *complexity of the algorithm itself*. Reliability of the software often to a high degree depends on the complexity of the algorithm.

3. ORIGINAL BTC ALGORITHM

The N pixel image is divided into smaller blocks of the size m pixels and each block is processed separately. A compact representation is searched locally for each block. The decompression process transforms the compressed blocks back into pixel values so that the decoded image resembles the original one as much as possible. The mean value ($\overline{x}$) and standard deviation (σ) are calculated and encoded for each block.

$$\overline{x} = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{4}$$

$$\overline{x^2} = \frac{1}{m}\sum_{i=1}^{m} x_i^{\,2} \tag{5}$$

$$\sigma = \sqrt{\overline{x^2} - \overline{x}^{\,2}} \tag{6}$$

Then a two-level quantization is performed. Pixels with values less than the *quantization threshold* ($x_i<x_{th}$) are quantized to value a and pixels with values greater than or equal to the threshold ($x_i \geq x_{th}$) are quantized to value b. Here the threshold x_{th} is supposed to be $\overline{x}$ and the values a and b are chosen so that the *first and second sample moments* ($\overline{x}, \overline{x^2}$) are preserved in the compression. This is the case for the selection

$$\begin{aligned} a &= \overline{x} - \sigma \cdot \sqrt{\frac{q}{m-q}} \\ b &= \overline{x} + \sigma \cdot \sqrt{\frac{m-q}{q}} \end{aligned} \tag{7}$$

where q stands for the number of pixels $x_i \geq x_{th}$.

A compressed block appears as a triple ($\overline{x}$,σ,B), where $\overline{x}$ and σ give mean and standard deviation of the pixel values in the block and B stands for the *bit plane*, giving the quantization of the pixel values, see Fig. 1.

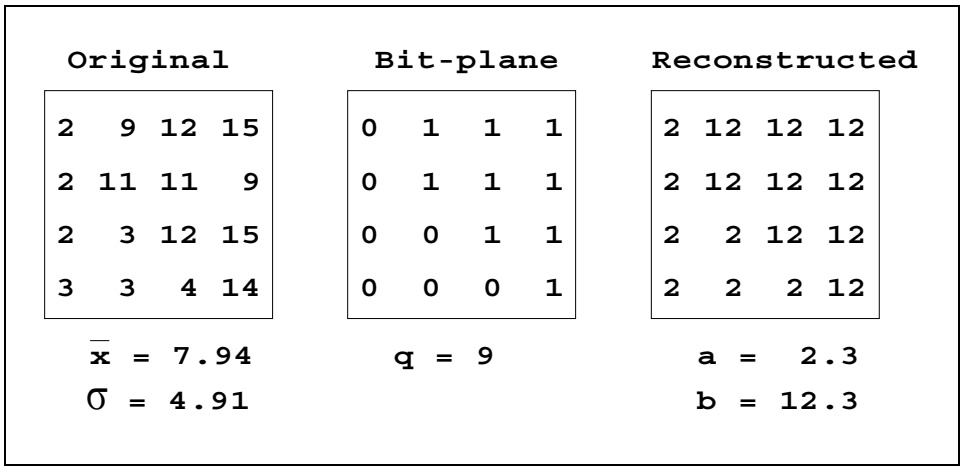


FIGURE 1. BTC by ($\bar{x}$,σ,*B*).

A straightforward coding of $\bar{x}$ and σ by *k* bits yields a bit rate of

$$\frac{k+k+m}{m} = 1 + \frac{2k}{m} \text{ bits per pixel (bpp)} \tag{8}$$

The method is fast, requires very little extra memory, is easy to implement [28], and has low computational demands. It preserves the quality of the reconstructed image and retains the edges. It also recovers excellently from transmission errors. One can still improve it in several different ways.

4. VARIANTS OF THE BASIC METHOD

Several modifications and improvements of the basic BTC have been proposed in the literature. They aim mostly at a lower bit rate, but other weaknesses of the method are also considered. The greatest deficiency of BTC is a relatively high bit rate as compared to other coding schemes, like DCT [1], vector quantization [18, 42], or the so-called second generation image compression techniques [29, 30]. Other significant drawbacks are *ragged edges* (*staircase effect*) in the reproduced image, and the *blocky appearance* in some cases.

Our intention is to make a summary of the BTC variants, their similarities and differences, and finally draw conclusions from the ideas presented. For this study BTC is divided into three separate steps, and each of these is analyzed separately. The overall structure of the algorithm is as follows:

BTC algorithm for coding a single block

1) Perform quantization
   - Select the threshold.
   - Select the quantization levels (*a*,*b*).
2) Code the quantization data.

3) Code the bit plane.

In many variants, the selection of the threshold and the pair (a,b) are interdependent, while some variants concern only one or the other. The stages 1, 2 and 3 are relatively independent.

Most of the variants do not presuppose a particular block size. Their implementations, however, are often designed for 4*4-blocks only. For clarity, we will therefore fix this block size. However, different block sizes will be studied in the computer simulations of Section 8. (See also the hierarchical decomposition of the image, in Section 5.) We also presume that the pixel intensities are represented by k=8 bits unless otherwise noted.

4.1. PERFORMING THE QUANTIZATION

In principle, selecting the threshold pixel value and selecting the quantization levels can be seen as two separate stages. On the other hand, most BTC algorithms are designed to preserve certain sample moments, or to minimize some fidelity function. Therefore x_{th}, a and b are in close association with each other.

Next we will present several variants, which can be classified roughly into three categories. The first category consists of the methods that will preserve certain moments. The methods of the second category minimize the MSE-value, and the methods in the third one minimize the MAE-value. The remainder of the variants are intermediate between these three types, or aim at the same goal with some heuristic algorithm.

A) Moment preserving quantization

A great number of the modifications found in the literature are based on the original, two moments preserving quantization [10]. The following formulas must hold for this:

$$\begin{aligned} m\overline{x} &= (m-q)\cdot a + q\cdot b \\ m\overline{x^2} &= (m-q)\cdot a^2 + q\cdot b^2 \end{aligned} \qquad (9)$$

This will be attained by using $\overline{x}$ as a threshold and selecting a and b as in (7). This selection is not optimal in the MSE-sense. However, the basic idea behind the method has been a different fidelity criterion, i.e. preserving the sample moments of the input [12]. A motivation for the method is that the human eye does not observe small changes of intensity between individual pixels, but is sensitive to the average value and contrast in larger regions [36].

B) Third moment preserving quantization

One can modify BTC so that it preserves in addition to $\overline{x}$ and $\overline{x^2}$ the third moment $\overline{x^3}$ [10], too. This can be achieved by selecting x_{th} so that the parameter q attains the value

$$q = \frac{m}{2} \cdot \left(1 + A \cdot \sqrt{\frac{1}{A^2 + 4}}\right) \tag{10}$$

where

$$A = \frac{3\bar{x}\overline{x^2} - \overline{x^3} - 2\,\bar{x}^{\,3}}{\sigma^3} \tag{11}$$

In comparison to the basic BTC the coding phase is more involved. On the other hand, decoding is still fast. The point of the variant is that it improves some subtle features (near edges) of the image. In fact, the third moment corresponds to the skewness of the distribution [9]. Negative values indicate negative skewness and positive values positive skewness. As in method A, the quantization levels (*a*,*b*) must be selected according to (7).

C) Generalized moment preserving quantization

The BTC method can be generalized to preserve the *r*, 2*r* and 3*r*'th moments, see Halverson *et al.* [23]. We bypass here the exact formulation of the method. Higher moments cause overflow problems and rarely good quality images. If one wants to preserve *r* and 2*r*'th moments only, the threshold is

$$x_{th} = \sqrt[r]{\overline{x^r}} \tag{12}$$

Compared to the original first and second moments preserving quantization, the method slightly improves the MSE-value [23].

D) Absolute Moment BTC

Absolute moment block truncation coding (AMBTC) by Lema and Mitchell [31] preserves $\bar{x}$ and the sample *first absolute central moment*

$$\bar{\alpha} = \frac{1}{m} \cdot \sum_{i=1}^{m} \left| x_i - \bar{x} \right| \tag{13}$$

Quantization levels are calculated from

$$a = \bar{x} - \frac{\bar{\alpha}}{2} \cdot \frac{m}{m - q} \tag{14}$$

$$b = \bar{x} + \frac{\bar{\alpha}}{2} \cdot \frac{m}{q} \tag{15}$$

One can show that $a=\bar{x}_l$ (lower mean) and $b=\bar{x}_h$ (higher mean), where

$$\bar{x}_l = \frac{1}{m-q} \cdot \sum_{x_i < \bar{x}} x_i \tag{16}$$

$$\bar{x}_h = \frac{1}{q} \cdot \sum_{x_i \geq \bar{x}} x_i \tag{17}$$

Furthermore, this selection minimizes the MSE-value among the BTC-variants that use $\bar{x}$ as a quantization threshold [60]. It can, however, easily be shown that it is optimal for other selections of x_{th}, too. The coding and decoding processes are very fast for AMBTC because square root and multiplication operations are omitted.

E) MSE-optimal quantization

The MSE-optimal choice for (*a*,*b*) is according to AMBTC, and it is independent of the quantization threshold. Therefore the MSE-optimal quantization can be obtained by selecting the quantization threshold so that it minimizes:

$$MSE = \sum_{i=1}^{m-q-1} x_i - a^{\,2} + \sum_{i=m-q}^{m} x_i - b^{\,2} \tag{18}$$

(We suppose that x_i-values are here in an ascending order.) An obvious way to find the right threshold, is an exhaustive search among the *m* candidate pixel values of a block [10]. The MSE-optimal quantization is also known as *minimum MSE quantization* (MMSE).

F) MAE-optimal quantization

The MAE fidelity criterion (2) can be minimized by selecting

$$a = median\ x_1, x_2, \ldots, x_{m-q-1} \tag{19}$$

$$b = median\ x_{m-q}, \ldots, x_m \tag{20}$$

Now the optimal threshold is again found by an exhaustive search [10]. The MAE-optimal quantization is also known as *minimum MAE quantization* (MMAE).

G) Heuristic criterion

Goeddel and Bass [19] proposed a heuristic selection criterion for the threshold:

$$x_{th} = \frac{x_{\min} + x_{\max}}{2} \tag{21}$$

where x_{min} and x_{max} stand for the *minimal* and *maximal pixel values* of a block. The criterion gives improved MSE-values in comparison to method A. The moments will not be preserved by the technique. However, if we use formulas (16-17) to select (*a*,*b*) instead of (7), this

criterion can be considered as a practical approximation for the MSE-optimal quantization (method E).

H) Lloyd quantization

Another nearly MSE-optimal quantization is obtained by an iterative algorithm proposed by Efrati *et al.* [14] and Lu *et al.* [34]:

1. Let $x_{th} = \overline{x}$.
2. Compute *a* and *b* according to (16) and (17).
3. If $x_{th} = (a+b)/2$ then exit.
4. Let $x_{th} = (a+b)/2$. Go to step 2.

The algorithm produces the quantization levels and threshold of the *Lloyd quantization* [33]. The average number of iterations needed (phase 2), is only 1.71 per block with our set of test images (see Section 8).

I) Hopfield neural network

A totally different approach to the quantization is proposed by Qiu *et al.* [47, 48]. No direct quantization threshold is given, but instead a Hopfield neural network outputs for each input pixel, whether it should be quantized to *a* or *b*. This method gives circa 7 % better MSE-values compared to the method D [47]. It has also been shown that the result is not optimal, but gives virtually identical results to the Lloyd quantization [38].

4.2. CODING THE QUANTIZATION DATA

The second task in the structured BTC-algorithm is to encode the given quantization data of a block. Before this can be done, an important choice must be made: which data should be coded. There are two alternative approaches for sending the quantization data to the decoder.

In the first approach the quantization data is expressed by two statistical values, representing the *mid-value* and *variation quantity* of a block (usually mean and standard deviation). It is known that the dispersion of the variation quantity is smaller than the dispersion of individual pixel intensities, and small values occur more frequently than large ones. This gives a compact representation for the block without any redundancy.

In the original moment preserving BTC (methods A and B) the quantization data are represented by the pair $(\overline{x},\sigma)$. This is not the case for the other quantization methods. For example AMBTC (method D) uses the mean and the absolute first central moment (13). If the heuristic threshold selection of [19] (method G) is applied, then one has to code the threshold and deviation corresponding to it:

$$\sigma' = \sqrt{\frac{1}{m}\sum_{i=1}^{m} x_i - x_{th}^{\ 2}} \qquad (22)$$

A drawback to this approach is that the quantization levels are calculated at the decoding phase from the quantized values of $(\bar{x},\sigma)$ containing rounding errors. Thus extra degradation is caused by the coding phase.

The other approach is to calculate the quantization levels (*a*,*b*) already at the encoding phase and transmit them. In this way one can minimize both the quantization error and the computation needed at the decoding phase. They are also independent of the quantization method selected. Certain methods (e.g. methods C, E, F, H, and I) of Section 4.1 even exclude the possibility of expressing the quantization by statistical quantities, therefore (*a*,*b*) is the only choice. The pair (*a*,*b*) contains redundancy, which can be easily removed by a suitable prediction and coding technique.

Several methods for coding the quantization data will be presented next.

A) Fixed number of bits

The most straightforward approach is to code the quantization data by 8+8 bits as such. No extra computations are then needed and direct access to the coded file is retained. It has been observed [10], that the decrease in the quality of the image will be small if $(\bar{x},\sigma)$ is quantized to 6+4 bits. Here one can simply allocate the values evenly within the ranges $\bar{x} \in [0,255]$, and $\sigma \in [0,127]$ (or even truncate the standard deviation to the range $\sigma \in [0,63]$). In the same way, less than 8+8 bits can be allocated for the pair (*a*,*b*). Our experiments indicate that the MSE-value of the test image Lena increases only circa 3 % when using 6+6 bits in the quantization, see Section 8.

B) Joint Quantization

Healy and Mitchell [24] improved the 6+4 bits coding system by applying *joint quantization* to $(\bar{x},\sigma)$ with 10 bits. Here the accuracy of $\bar{x}$ depends inversely on the value of σ, see Fig. 2. For example if σ=4 there are 10-3=7 bits left to code $\bar{x}$. The technique still gives the same bit rate, but it has been argued that the quantization error is more visible to the human eye in the low variance regions, and a more accurate representation is needed in these. In contradistinction to this effect, our experiments show that the greatest contribution to the MSE-values comes from the high variance blocks. This indicates that there is a contradiction between MSE and subjective quality.

The concept *joint quantization* was originally introduced by Mitchell and Delp [37]. They used the method for 16- and 32-level images by using 6 and 7 bits for the pair $(\bar{x},\sigma)$, respectively.

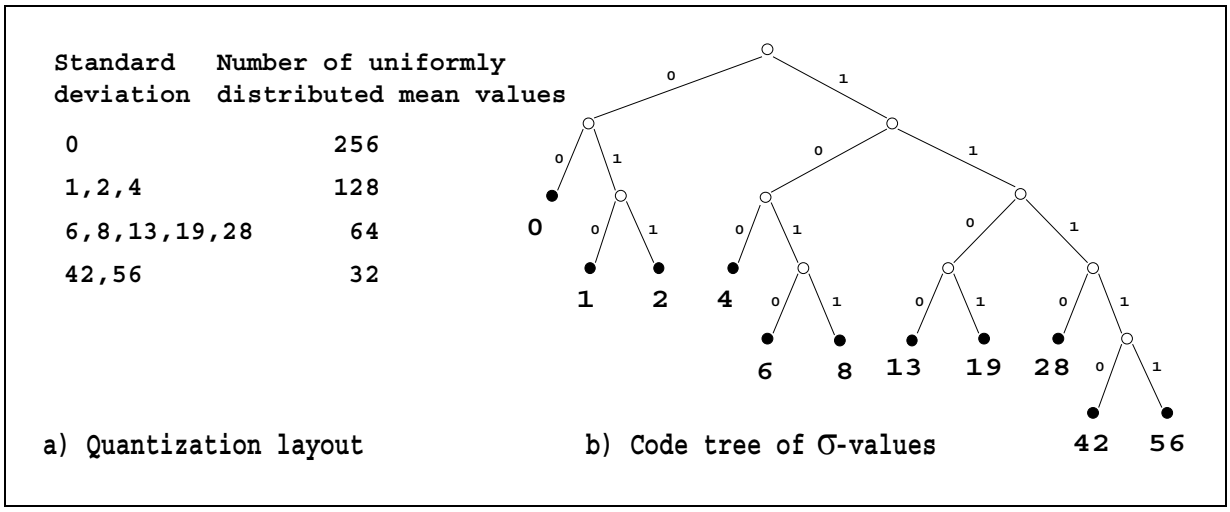


FIGURE 2. Joint quantization of $\bar{x}$ and σ.

C) Vector Quantization

The use of *vector quantization* [18, 42] for coding the bit plane was first proposed by Udpikar and Raina [61]. They used a variant of AMBTC [60] that stores a compressed block in the form ($\bar{x}$,$\bar{x}_l$,*B*). The decoder replaces each 0-bit of *B* by $\bar{x}_l$ and each 1-bit by $\bar{x}_h$, where

$$\overline{x_h} = \overline{x_l} + \frac{m}{q} \cdot \overline{x} - \overline{x_l} \tag{23}$$

By applying *vector quantization* (VQ) to the pair ($\bar{x}$,$\bar{x}_l$), the bit rate can be reduces to 8 bits for ($\bar{x}$,$\bar{x}_l$) with the cost of increased MSE-value. The method, however, does not consider interblock correlations, i.e. the dependencies between neighboring blocks.

Weitzman and H.B. Mitchell [69] apply *predictive vector quantization* for reducing the interblock redundancy. Instead of coding (*a*,*b*), they apply VQ to the prediction errors (ea_i,eb_i):

$$\begin{aligned} ea_i &= a_i - \left(2a_{west} + 2a_{north} + a_{northwest}\right)/5 \\ eb_i &= b_i - \left(2b_{west} + 2b_{north} + b_{northwest}\right)/5 \end{aligned} \tag{24}$$

Here a_{west}, $a_{northwest}$, and a_{north} refer to the *a*-values of the neighboring blocks that have already been coded.

Another approach was proposed by Alcaim and Oliveira [2]. They encode $\bar{x}$ and σ separately, making two subsample images, one from the $\bar{x}$-values and another from the σ-values of the blocks. These subsample images are decomposed into 2*2 blocks which are then coded by VQ. In this way one can reduce both the interblock and intrablock redundancy.

D) Discrete Cosine Transform

Wu and Coll [73] propose the compression of the pair (*a*,*b*) by the use of *discrete cosine transform* (DCT). Two *subsample images* are formed, one from the *a*-values and another from the *b*-values of the blocks. These subsample images contain one pixel per block of the original image. They are then coded by the *adaptive* DCT presented in [4, 5]. Details of the subsample images must be preserved as accurately as possible, since they have an influence on a large number of pixels in the reconstructed image.

An improved variant of the method is also given in [73]. Here the second subsample image is not formed from the *b*-values, but from the (*b*-*a*)-differences instead. If one wants to retain the same MSE-level, the differences must be calculated at the coding phase by *b*-*a'*, where *a'* is the reconstructed *a*-value after inverse DCT.

The use of a computationally demanding algorithm like DCT is justified by the fact that the subsample images are only 1/16 of size of the original image, for the block size 4*4. When the compression ratio of the DCT is predefined to 3:1, the total bit rate of *a* and *b* is 0.33 bpp at the cost of higher MSE.

E) Lossless coding

The two subsample images can be compressed by several different image coding algorithms. As stated earlier, it is important that the information of the subsample images is preserved as far as possible. Therefore the use of a *lossless* image compression algorithm is apparently a good choice. Two variants of this type, and another related method, will be presented next. For more information on lossless image compression, see [25, 26, 49, 50, 57, 58].

1. Predictive entropy coding:

Fränti and Nevalainen [16] propose the use of a *predictive coding* scheme with a suitable *context model*. For each (*a*,*b*)-pair to be coded, a prediction is made using the data already coded. Instead of coding (*a*,*b*), the prediction errors ea_i and eb_i are calculated:

$$\begin{aligned} ea_i &= a_i - \frac{a_{west} + a_{north}}{2} \\ eb_i &= b_i - a_i \end{aligned} \qquad (25)$$

Denote the entropy of an individual prediction error ea_i by $H_0(ea_i)$ corresponding to *memoryless source*:

$$H_0(ea_i) = -\log p\ ea_i \qquad (26)$$

Here $p(ea_i)$ stands for the probability of the occurrence of a particular prediction error ea_i. (The entropy of $H_0(eb_i)$ is found similarly.) Because a *one-pass compression method* is preferred, the probabilities are determined *adaptively* from the coded image. By applying *arithmetic coding* [72] one can express ea_i and eb_i by the number of bits given by the entropy.

The entropy coding scheme can be improved by using ea_{i-1} and eb_{i-1} as contexts when coding the prediction errors ea_i and eb_i. Here ea_{i-1} and eb_{i-1} refer to the prediction errors of the previously coded block. The benefit of the context model remains relatively small since the predictive coding already takes advantage of the neighboring blocks.

A drawback of the method is that arithmetic coding is time consuming, and the overall compression system is rather complex compared to the original BTC algorithm.

2. FELICS coding:

We consider next a new approach for coding the two subsample images. We apply FELICS coding (Fast and Efficient Lossless Image Compression System) by Howard and Vitter [25]. The method is highly applicable to BTC because of its practical usefulness, and it is also one of the most efficient lossless methods known.

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 it. Denote the values of the neighboring pixels by L and H so that L is the one which is smaller. Howard and Vitter [25] have found that the probability of individual pixels obeys the distribution given in Fig. 3. Our experiments show that the same distribution holds very well for the pixels of the subsample images, too.

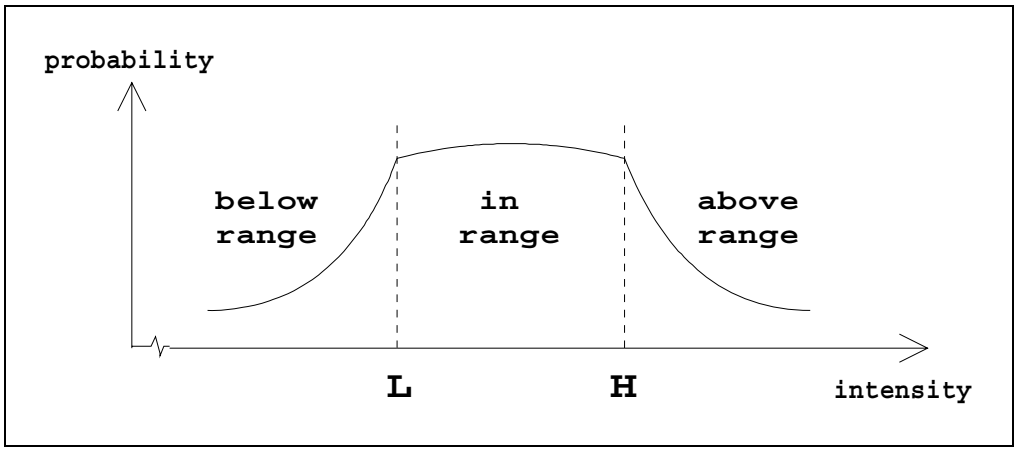


FIGURE 3. Probability distribution of intensity values.

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* [52] with adaptive k-parameter selection. For details of FELICS coding, see [25].

We can improve the coding of b if we remember that a is already known at the moment of coding b, and that $b \geq a$. Thus, if a falls into the same interval as b, the actual range of b can be reduced so that only the potential values are considered. For example if a belongs to the in-range of b, the interval is reduced by changing the lower bound L to a. The improvement, however, remains marginal.

The results of FELICS are quite similar compared to the results of the predictive entropy coding scheme, but it is fast and much less involved, see Section 8.

3. LGD-algorithm:

Lu et al. [34] propose a coding method, which also involves predictive coding and entropy coding, as in [16]. The difference between the methods is that in [34] the predictive coding is performed for the original image, while in [16] it is applied to the quantization levels.

The coding scheme is as follows: for each pixel in a block to be coded, differences between the value of a current original pixel and the value of the previous reconstructed pixel in the same horizontal line is constructed. Quantization levels a and b are then calculated for the differences by the Lloyd quantization of Section 4.1. For this it is known that $x_{th}=(a+b)/2$, so the pair $(x_{th},x_{th}-a)$ can be coded instead of (a,b). It is expected that these values are small integers close to zero, so a simple prefix code is proposed in [34]. Here the values close to zero are represented by fewer bits, see Fig. 4. Notice, that one extra bit is needed for the sign of x_{th}, because the values are differences, and they can have a negative value, too. It seems evident that *Rice-Colomb coding* [7, 52] could be applied instead of this code table.

value	code	value	code	value	code
0	000	6	11000	12	11110
1	001	7	11001	13	1111100000000
2	010	8	11010	14	1111100000001
3	011	9	11011	15	1111100000010
4	100	10	11100	:	:
5	101	11	11101	:	:

FIGURE 4. Code table for the differences by [34].

4.3. BIT PLANE REDUCTION

In the basic BTC method, both quantization data and the bit plane of a block require 16 bits each. Several methods for reducing the bits needed for the quantization data was presented in Section 4.2. One can reduce circa 30 % of the bits needed without any (further) loss. A larger reduction is still possible at the cost of decreased image quality. In this section, we will consider several attempts to reduce the size of the bit plane.

A) Skipping the bit plane

If the contrast of x_i-values is very small one can omit the storing of B and thus implicitly code all pixels by $\bar{x}$, see [37, 41]. The criterion for using this one-level quantization varies. Mitchell and Delp [37] omit the bit plane if $\sigma<1$. The pair $(\bar{x},\sigma)$ is coded before the bit plane and therefore the decoder always knows whether the bit plane follows or not. Nasiopoulos *et al.* [41] use *range* as a *threshold criterion* instead of the variance:

$$range = x_{\max} - x_{\min} \tag{27}$$

A block is represented by its average if the range is less than a given threshold. An extra code bit indicates whether the bit plane is skipped or not, and therefore only $\bar{x}$ needs to be coded.

The one-level quantization method of uniform blocks causes increased blocky appearance in the reconstructed image. This is because the blocks which were represented by their averages have intensity values which are necessarily not close to each other and their boundaries can therefore be recognized. This negative effect can be avoided or at least reduced by *averaging filtering*, see Section 6.

B) Prediction technique

A simple *prediction technique* proposed by Mitchell and Delp [37] reduces the volume of the bit plane by encoding only half of the bits, see Fig. 5. The remainder of the bits are "concluded" on the basis of the coded ones by a set of logical expressions. This method naturally causes more distortion to the image, and in our experiments the MSE-value was multiplied by a factor of 2 to 4, see Section 8. The result of the prediction technique is weakest in the blocks with a large contrast. Therefore an *adaptive bit plane coding method* by [37] uses the prediction technique only when the variance is small enough, say $1 \leq \sigma \leq 4$.

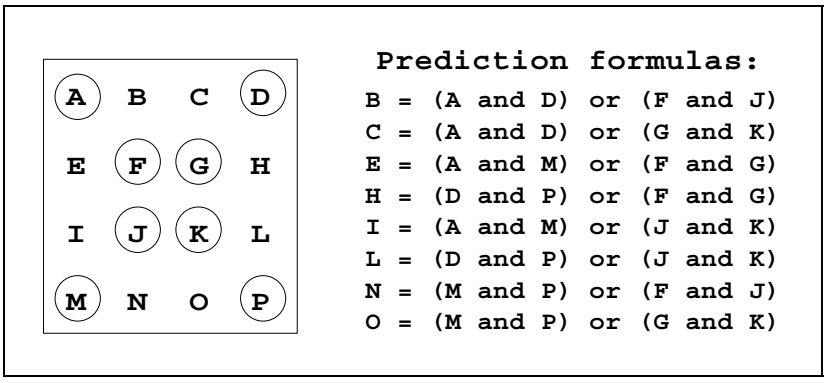


FIGURE 5. Description of the *prediction logic* [37]. Encircled bits are stored.

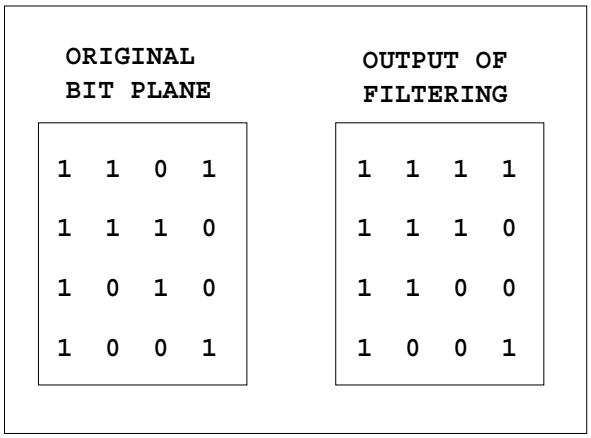
C) Filtering technique

Median filtering [59] 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. It has been shown that the block signals have a very high probability of belonging to the root signal set [3].

Arce and Gallagher [3] proposed a 1D-median filtering for the BTC bit plane. The window was set to three pixels including the current pixel and two of its neighboring pixels in the same row. For the border pixels, the outer pixels are presumed to be of the same intensity as the border pixel. Each row of a block is median filtered to its *roots*. In the case of 4*4 blocks, roots can be reached by a single filtering pass. Compression is achieved because the number

of possible patterns has decreased from 16 down to 10, see Fig. 6 and 7. All the combinations that include isolated bits have been eliminated.

To realize the benefit of the filtering, only the 8 most common combinations (marked by '*') of Fig. 7 are allowed to occur. The remaining patterns can now be coded by 3 bits each, giving a total 0.75 bits per pixel for the bit plane. The scheme is extremely simple and can be implemented very efficiently by look-up table.



Root#	Pattern	Experimental frequency
* 1:	0 0 0 0	0.20
* 2:	0 0 0 1	0.08
* 3:	0 0 1 1	0.11
4:	0 1 1 0	0.02
* 5:	0 1 1 1	0.07
* 6:	1 0 0 0	0.09
7:	1 0 0 1	0.02
* 8:	1 1 0 0	0.12
* 9:	1 1 1 0	0.09
*10:	1 1 1 1	0.20

FIGURE 6. Illustration of 1D median filtering. **FIGURE 7.** Roots of the filtering.

The method has been improved further so that the vertical correlations are also considered. Here the first row of each block is coded by 3 bits as proposed above, but for the other rows only 2 bits are used. This is done by reducing the number of transitions to 4 discarding the least likely patterns and taking into consideration the pattern of the previous row. This scheme reduces the bit rate of the bit plane to (3+2+2+2)/16 = 0.56 bits per pixel. For details, see [3].

Wang and Neuvo [66] proposed the use of 2D-filtering for exploiting the vertical correlations. They give two related filtering schemes, *separable median filtering* (SMF) and *cross median filtering* (CMF). The first one is a two-stage procedure where the length three window filtering is first applied in a horizontal direction and then in a vertical direction. In CMF a symmetrical two-dimensional window is used and only one stage is needed for one filtering pass. The number of roots for SMF is 2126 and for CMF 8208. The bit plane can therefore be coded respectively by 11, or 13 bits giving 0.69, or 0.81 bits per pixel.

Wang *et al*. [64, 65] also considered a third filtering scheme by applying 1-dimensional *morphological filters*. The main difference from the other filtering schemes is the way the root signal set is formed. Two sets of roots, namely roots of *opening* and roots of *closing*, are adaptively used in the method. For more information about morphological filters, see [43, 63].

A summary of test results for the different filtering methods is given in Table 1. The results are for the test image Lena with the moment preserving BTC [10], and are collected from [63].

TABLE 1. Summary of the filtering scheme results for Lena.

Method:	Reference:	No. of bits:	MSE:	MSE-inc.:
Original BTC	[10]	16	44.76	00.0 %
Cross-median	[66]	13	51.11	14.2 %
Morphologial	[65]	12	54.56	21.9 %
1D-median	[3]	12	55.21	23.3 %
Separable median	[66]	11	55.46	23.9 %
2D-median	[3]	9	Not given	Not given

D) Vector Quantization

Several approaches have been proposed for reducing the volume of the bit plane by the use of vector quantization [14, 61, 68, 73] or a similar ad hoc technique [41]. 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 [18, 48]. The *codebook* is generated off-line on the basis of training vectors by using a suitable algorithm, like the *generalized Lloyd algorithm* [18, 32]. Then the same codebook is used for whatever images are to be coded. Adaptive vector quantization [20] was considered by Wu and Coll [73], but they prefer the non-adaptive, *universal codebook* VQ, because of the speed.

The representative for a bit plane can be chosen by a total search, i.e. by testing each candidate codevector and selecting the one which minimizes the given distortion function. In this way, the result of the search is always optimal (corresponding to the distortion function) but it is impractical for large codebooks. The decoding, however, is always quick.

A localized search was proposed by Weitzman and H.B. Mitchell [68]. 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. Even if the search method is not extensive anymore, the correct code vector (corresponding to the extensive search) will still be found with a very high probability (> 99%) according to [68].

The performance of a search method depends not only on the organization of the codebook, but also on the selected distortion function. MSE is the most obvious measure, however, it cannot be calculated independently from the quantization data (*a*,*b*) [17]. Most of the existing methods thus use *Hamming distance*, i.e. the number of positions in which the elements of the two binary vectors differ, as a distortion measure [41, 61, 68, 73]. 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 [73]. The advantage of Hamming distance is that it can be implemented by *look-up table* (LUT).

The size of the codebook was 256 in [14, 61, 73] and 128 in [41], so the bit rate of the bit plane was respectively 0.5 and 0.44. Weitzman and H.B. Mitchell [70] proposed the use of several codebooks of different sizes. The size of a codebook depends on the contrast of a block (measured by *b*-*a*): the greater the contrast the larger the codebook. Nasiopoulos *et al.* [41] coded only blocks whose contrast is between a given range (not too large but not too low 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 is proposed by Efrati *et al.* [14] for avoiding the staircase effect. BTC algorithm with VQ is used for the low contrast blocks, and a three-level quantizer is applied for the high contrast blocks, see Section 6. Thus, the algorithm uses two different codebooks, one for the low contrast and another for the high contrast blocks.

E) Interpolation

All the methods discussed above were based on the idea that the bit plane is only partially coded, thus omitting a part of it, or by selecting a representative for the bit plane (or for its rows) from a smaller subset, which can either be called a root signal set or a codebook. Therefore the reconstructed image contains pixels where an *a*-value has been changed to a *b*-value or vice versa. The cost of these errors may be high in the MSE-sense, if the contrast in the block is large. To avoid this problem, a totally different approach is taken in the following method.

Zeng's *interpolative* BTC [76, 77] is similar to the prediction technique (method B) in the sense that half of the bit plane is omitted. At the decoding phase the partial image is first reconstructed and then the missing pixel values are interpolated on the basis of the existing ones. Zeng has proposed two variants of the method: one which codes 50 % (IBTC-1) and another which codes 25 % of the bits (IBTC-2), see Fig. 8.

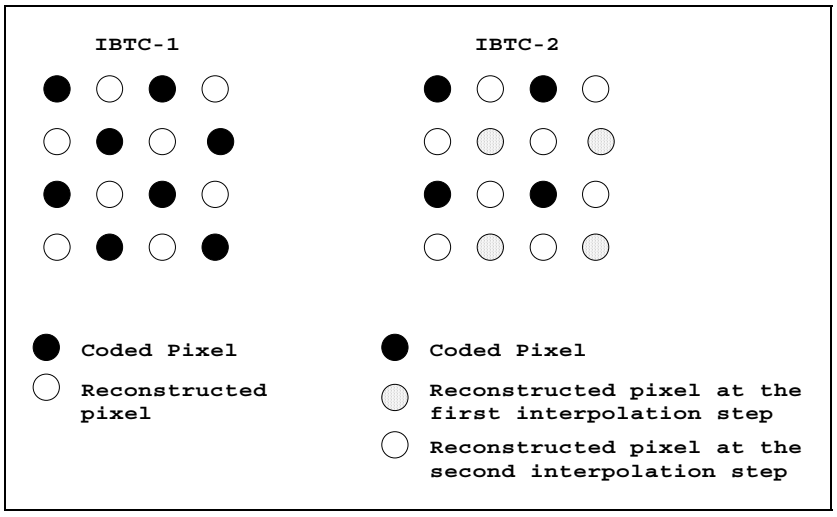


FIGURE 8. Interpolation patterns of IBTC-1 and IBTC-2.

The interpolation is done by stack filters designed to be optimal in the MAE-sense [71, 76, 78] or by using median filters [77]. The latter is recommended because of the ease of implementation. In the IBTC-1 four adjacent coded pixels are used to calculate the intensity of the current one:

$$y_{i,j} = \text{median}\ \left\{ y_{i,j-1}, y_{i,j+1}, y_{i-1,j}, y_{i+1,j}, \frac{1}{4} \cdot \left(y_{i,j-1} + y_{i,j+1} + y_{i-1,j} + y_{i+1,j} \right) \right\} \tag{28}$$

In the IBTC-2 a two-phase interpolation is used. At the first phase the following median filter is used, where each pixel is interpolated on the basis of the coded pixels.

$$y_{i,j} = \text{median}\ \left\{ y_{i-1,j-1}, y_{i-1,j+1}, y_{i+1,j-1}, y_{i+1,j+1}, \frac{1}{4} \cdot \left(y_{i-1,j-1} + y_{i-1,j+1} + y_{i+1,j-1} + y_{i+1,j+1} \right) \right\} \qquad (29)$$

At the second phase, the median filter of (28) is used. Note that the interpolation is a separate phase of the decoding process and therefore the block boundaries do not disturb the reconstruction of the missing pixels.

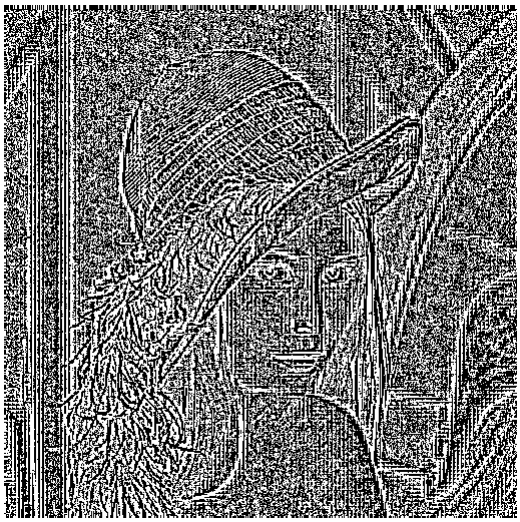
We also implemented a third interpolative scheme. Three pixels out of four were coded in it. The missing ones are interpolated by (28). This gives three different interpolation levels with corresponding bit rates 0.75, 0.50 and 0.25 for the bit plane.

F) Entropy coding

Since the bit plane can be considered as a *binary image*, one might expect that a binary image coding method could be used. On the other hand, the BTC-process ensures that the number of 0 and 1 bits are almost equal and the bits are relatively evenly distributed all over the bit plane, see Fig 9. Therefore simple coding methods, like *run length coding* [44], are unlike to give any compression.

Entropy based bit plane compression has been described in [16]. The image (the bit plane) is proceeded in row major order from left to right. A high degree Markov model is applied for each pixel. The value of a pixel is predicted by the 7-bit context template shown in Fig. 10, and then coded by arithmetic coding according to its probability. QM-coder [45, 46] was used as the arithmetic coding component.

The size and shape of the template have major effect on the coding efficiency when compressing *binary images*, and better results can be achieved with larger templates [39]. However, this does not necessarily apply to the block truncation coding. The problem here is the blockwise quantization according to the mean $\bar{x}$ of each individual block. This destroys the correlation of neighboring bits belonging to two different blocks.



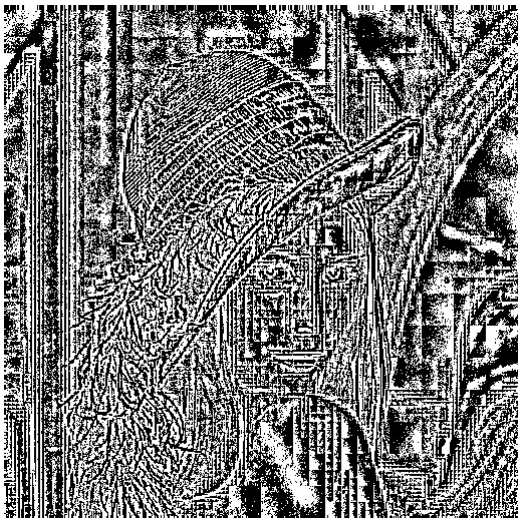


FIGURE 9. a) Bit plane of Lena by BTC b) Bit plane of Lena by HBTC

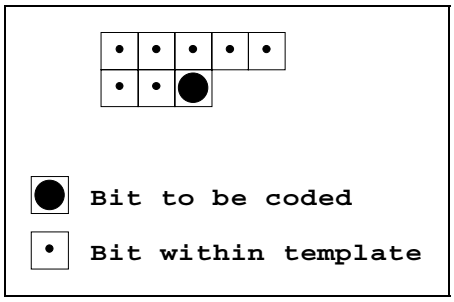


FIGURE 10. 7-bit template.

The benefit of the entropy coding remains relatively small: for Lena, only ca. 0.1 bits per pixel (=10%) can be saved. The result are slightly better when hierarchical decomposition is applied, but not as good as expected.

5. HIERARCHICAL DECOMPOSITION OF THE IMAGE

A natural extension of the single-level BTC is to use a hierarchy of blocks [27, 54]. In h*ierarchical block truncation coding* (HBTC) the processing begins with a large block size (m_1*m_1). If *variance* σ^2 of a block is less than a predefined threshold v_{th} the block is coded by a BTC-variant. Otherwise it is divided into four subblocks and the same process is repeated until the variance threshold criterion is met, or the minimal block size (m_2*m_2) is reached, see Fig. 11. A *quadtree structure* [55] is maintained to render the decoding possible.

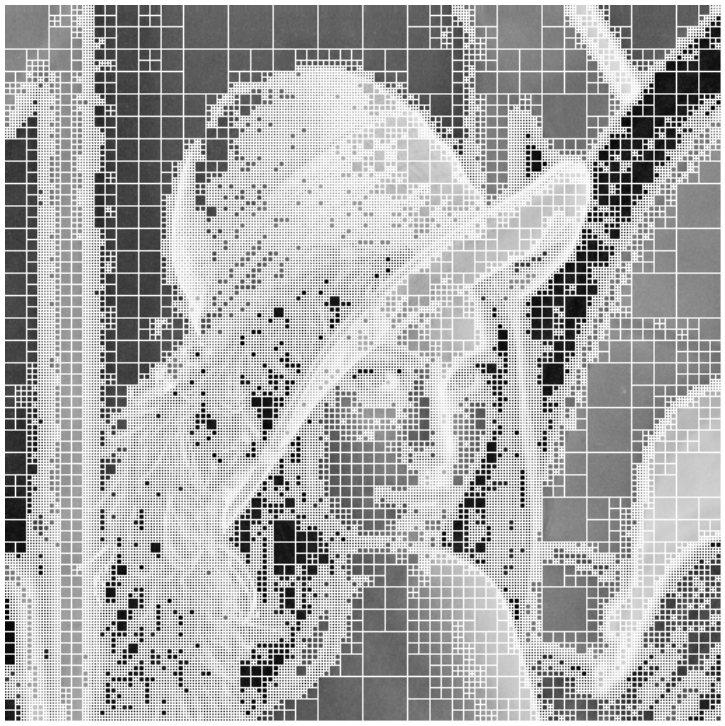


FIGURE 11. Hierarchical decomposition of Lena.

By adjusting the variance threshold v_{th}, and the block sizes m_1, m_2, one can direct the benefit of the hierarchical decomposition to achieve a low bit rate or high image quality. The block sizes in the hierarchy were set to m_1=8, m_2=2 by Kamel *et al.* [27] and m_1=32, m_2=4 by Roy and Nasrabadi [54]. Nasiopoulos *et al.* [41] proposed a two-level hierarchy with m_1=4, m_2=2. They use *range* (see formula 27) as threshold criterion, instead of variance.

The high MSE-values rarely originate from the largest blocks unless the variance threshold is too high. Therefore, if a low MSE-value is the primary criterion of the image quality, the maximal block size of 32*32 is recommended. The selection for the smallest block size m_2 causes a more dramatic change in MSE. If one wants to achieve very high quality images the smallest block size must be set to m_2=2. The variance threshold v_{th} can be considered as a finetuning parameter.

Although the hierarchical decomposition seems to be an excellent idea, it has some drawbacks also. Several non-hierarchical methods (Section 4) preserve direct access to the compressed file, but a straightforward implementation of HBTC does not. This also means that the fault tolerance has decreased. Moreover, large uniform areas may cause blocky appearance in the reconstructed image.

6. EXTENSIONS OF BTC

In this section, several external pre- and post-processing techniques will be discussed. Some of them aim at an improved visual quality of the reconstructed image, while others simply strive for lower bit rates. BTC's application to color images and video compression will be shortly discussed.

A) Smoothing filter

A simple smoothing operation of the original image is proposed by Mitchell and Delp [37]. The purpose of the smoothing is to reduce noise in the original image which causes pixel values near the threshold to be quantized to the wrong value. Each pixel $x_{i,j}$ of a block is replaced by a weighted sum of itself and the four adjacent pixels.

$$x'_{i,j} = 0.6 \cdot x_{i,j} + 0.1 \cdot \; x_{i,j-1} + x_{i-1,j} + x_{i+1,j} + x_{i,j+1} \tag{30}$$

B) Blocky appearance

Blocky appearance is a common feature of block coding techniques like BTC. The relatively small size of blocks (4*4) keeps the effect almost unnoticeable for human eyes. However, when large block sizes are applied in the hierarchical variant, the effect becomes more visible. Another source for the blocky appearance is the *skipping uniform blocks* -technique presented in Section 4.3, where pixels of low contrast blocks are represented by their average.

For reducing the blocky appearance Nasiopoulos *et al.* [41] proposed the use of smoothing filter as a postprocessing technique. The reconstructed image is scanned using an 8*8-window. If the contrast in the window is smaller than a predefined smoothness criterion, the area is smoothed by a 3*3 averaging filter. Here each pixel $y_{i,j}$ of the window is replaced by the value $y'_{i,j}$.

$$y'_{i,j} = \frac{1}{9} \cdot \sum_{i=-1}^{1} \sum_{j=-1}^{1} y_{i,j} \tag{31}$$

The method takes care of the blocky appearance of the reconstructed image, but the smoothed regions now look flat and artificial. In order to give back to the image the lost *texture*, Gaussian random noise is added to the averaged areas.

C) Discrete Cosine Transform

A hybrid of BTC and DCT is proposed by Delp and Mitchell [10]. First a highly compressed image is obtained by taking the two-dimensional discrete cosine transform over 16*16 pixel blocks. Only the eight *non-dc coefficients* in the low frequency section of each block are retained. A difference image is constructed by subtracting the transform coded image from the original. The basic BTC algorithm is then applied to this difference image. Both BTC and transform codes are sent to the decoder with bit rate 1.63 + 0.25 = 1.88 bpp.

The hybrid method was argued for using the different properties of the methods. While BTC preserves edges, they tend to be ragged (the staircase effect). On the other hand, transform coding usually produces smooth edges. In the hybrid formulation, the aim is to improve visual quality of the image. At the same time, MSE-value is reported to decrease [10].

D) Differential Pulse Code Modulation

Differential pulse code modulation (DPCM) is a well-known signal coding technique and it has been widely used in image compression [8, 44, 50]. In DPCM, each pixel value is first predicted on the basis of previously coded pixels. The result of the prediction is then subtracted from the actual pixel values, and the difference $e_{i,j} = x_{i,j} - \hat{x}_{i,j}$ is coded.

Delp and Mitchell [11] propose a composite method of DPCM and BTC. First an *open-loop* prediction is performed to construct the mean value and standard deviation of the differences ($\bar{e}$,σ_e), see Fig. 12. In this stage, quantization levels *a* and *b* are also calculated, because they are needed for the predictor. Then the quantization procedure is performed by applying DPCM with a *closed-loop* prediction. Prediction function is as follows.

$$\hat{x}_{i,j} = 0.8 \cdot y_{i-1,j} - 0.6 \cdot y_{i-1,j-1} + 0.8 \cdot y_{i,j-1} \tag{32}$$

Because the ranges of $\bar{e}$ and σ_e are smaller than those of $\bar{x}$ and σ, they can be coded with fewer bits. With the block size 8*8, Delp and Mitchell have found that 4+3 bits suffice; however, the detailed bit allocation table is not given. It has been shown that while $\bar{x}$ of a block is preserved, the second moment is not [11].

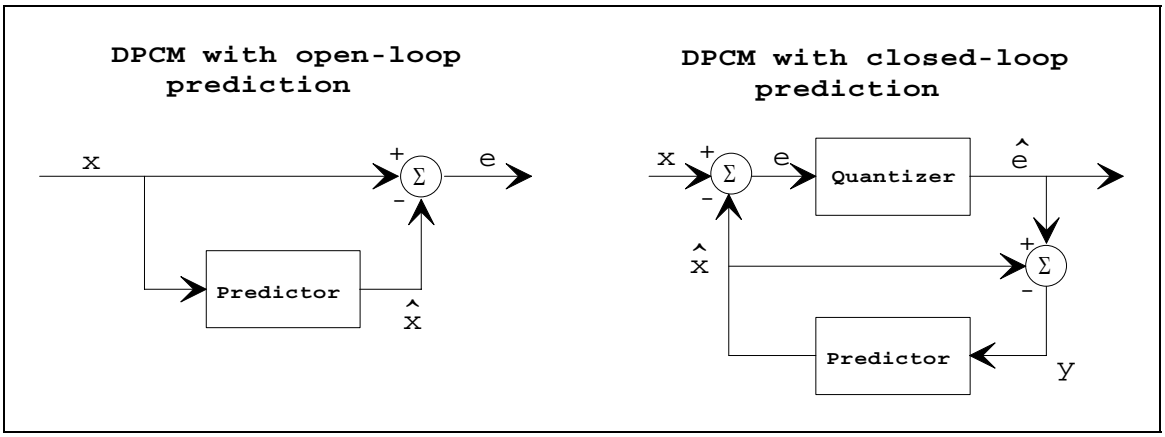


FIGURE 12. Differential Pulse Code Modulation.

A similar coding scheme is proposed by Lu *et al.* [34]. They, however, use a simple prediction function $\hat{x}_{i,j} = y_{k,j}$. Here $y_{k,j}$ denotes the nearest pixel of the same row, but outside of the current block. Thus this pixel is already known by the decoder, and the closed-loop prediction can be applied. The bit allocation system of the method was presented in Section 4.2.

E) N-level quantization

The basic idea of block truncation coding, two-level quantization, can be derived to a general *n*-level quantizer [12, 33, 35, 56]. Delp and Mitchell have proposed a moment preserving quantizer, which is shown to be related to the Gauss-Jacobi mechanical quadrature problem [12]. MSE-optimal quantizer has been given by Max [35]. These quantizers, however, presuppose that the probability distribution of the signal is either uniform, *Gaussian*, or

Laplacian. In Lloyd's quantizer [33] all quantization levels are given by an iterative algorithm.

Monet and Dubois [40] have proposed a uniform quantization method in which the overhead information of a block is first coded, consisting of the pair (x_{min}, x_{max}). The pixel values of a block are then scaled and normalized within this interval. They are finally quantized so that the number of quantization levels in a block is fixed regardless of the variance of the block.

More specific approaches has been taken by several authors concentrating on the 3-level quantization [13, 14, 19, 53]. Goeddel and Bass [19] proposed a 3-level quantizer to be used along with the original BTC. For each block, both quantizations are determined, but only the one with the smaller MSE-value is sent to decoder. Ronsin and DeWitte [53] use both 3- and 4-level quantization. They later developed another algorithm, in which the edge blocks are coded by a 3-level and the non-edge blocks by a 2-level quantizer [13]. The same idea was also applied by Efrati *et al.* [14].

Vector quantization of the quantization matrix is performed in all of these approaches. The codebook is designed to contain edge blocks [14]; or the end points of the edges are coded by an *edge following algorithm* [13, 53]. Wang and Mitra propose a more sophisticated algorithm by using *block pattern models* [67].

Ceng and Tsai [6] proposed the use of a 4-level quantizer without increasing the number of parameters. All four quantization levels can still be derived from the pair (*a*,*b*). Wu and Coll [75] has given a non-uniform *n*-level quantizer, where each of the quantization levels are iteratively determined aiming at minimal MAE. They also introduced a predictive entropy coding scheme for coding the overhead in the special case of a 4-level quantizer. Uhl [62] has extended the idea of multilevel quantization and proposed a hybrid algorithm. The blocks are classified and for each class, a different quantization is applied. They use 1, 2, 4, and 8-level quantizers, and a copy operation depending on the type of block.

The properties of these quantizers are briefly summarized in Table 2. The overhead refers to the storage requirements, supposing that the quantizers are adaptively applied for each block, as in the framework of block adaptive quantization methods. The overhead of the quantization matrix is expressed as bit per pixel, and in principle, is equal to the 2-base logarithm of the number of quantization levels in the quantizer. The quantization parameters require one byte per parameter (per block). Uniform quantizers need only two parameters, while non-uniform quantizers often require one parameter per each quantization level value (like Lloyd's quantizer). This might be somewhat impractical in block adaptive coding methods like BTC. However, a practical implementation [75] limits the number of levels to four.

TABLE 2. Properties of different quantizers.

Quantizer:	**Refe-rence.:**	**Quantization parameters:**	**Quant. levels:**	**Overhead**	
				Quant. param.:	**Quant. matrices:**
BTC	[10]	(*a*,*b*)	2	2	1
3-level BTC	[53]	(*a*,*b*)	3	2	log 3
4-level BTC	[6]	(*a*,*b*)	4	2	2

n-level BTC	[75]	one value/level	N	N	$\log N$
Block adaptive quantizer	[40]	(x_{min}, x_{max})	N	2	$\log N$
Lloyd quantizer	[33]	one value/level	N	N	$\log N$

F) Color images

The most common representation of color images, RGB, divides the image into three separate color planes, namely red, green, and blue. This color system is based on the trichromatic nature of the human vision [21]. There exists a high degree of correlation between the planes *R*, *G* and *B*, and therefore a *color space conversion* from RGB to YUV (or YIQ) is usually performed [46]. *Y* represents the *luminance* of the image, while *U,V* (*I,Q*) consists of the color information, i.e. *chrominance*:

$$\begin{aligned} Y &= 0.3 \cdot R + 0.6 \cdot G + 0.1 \cdot B \\ U &= B - Y \\ V &= R - Y \end{aligned} \tag{33}$$

$$\begin{aligned} I &= 0.74 \cdot V - 0.27 \cdot U \\ Q &= 0.48 \cdot V + 0.41 \cdot U \end{aligned} \tag{34}$$

Color images are considered as a three-component generalization of gray scale images, and thus they can be compressed by extending the existing algorithms. Lema and Mitchell [31] propose an algorithm in which the three components of the YIQ-space are separately coded by the absolute moment BTC. As the human visual system is most sensitive to the variations of *Y*, and less to *I* and least to *Q*, the chrominance planes are subsampled. It means, that only the average value of each 2*2-block in the *I*-plane and the average value of each 4*4-block in the *Q*-plane is retained. In the decompression phase, the size of the planes are recovered by using *bilinear interpolations*. The method yields a bit rate of 2.13 (1.62+0.41+0.10) bpp.

Wu and Coll [74] proposed the use of a single bit map for all three components, given by the quantization of the *Y*-component. The quantization levels of each component are further compressed by DCT, as presented in Section 4.2. High-frequency components of the DCT domain are discarded in the chrominance components so, that it correspond to a 3*3 averaging filter for the *I*-signal and a 5*5 averaging filtering for the *Q*-signal. Several variations of the basic idea are considered in [74], including both RGB and YIQ color space images. A bit rate of 1.71 bpp has been reported for the latter.

G) Video images

A straightforward application of BTC to video images proposed by Healy and Mitchell [24]. They extended the block to consists of pixels from consecutive frames by using block size of 4*4*3. A *registration algorithm* is used for detecting global translations, like camera movements. 2D-BTC is performed on movement blocks and edge regions. Bit rate can be reduced by coding only one 2D bit plane for the 3D-blocks. Further reduction is possible by skipping the coding of one or two frames, and coding only the blocks consisting of some

activities. All of these ideas are applied adaptively so that more accurate representation is used whenever necessary for preserving the image quality.

Roy and Nasrabadi [54] have experimented also with the hierarchical BTC on video images. Bit rates have been reported down to 0.39 bpp for the CCITT test sequence "Walter".

7. BLOCK TO BLOCK ADAPTIVE BTC

In an adaptive compression scheme one can change the compression method from block to block. The adaptation may concern the quantization method, bit plane coding method, or both of them. In principle, one could even change the whole compression system, for example use BTC for one block and another coding system for the next block. Several attempts of this kind apply multilevel quantizers (see Section 6) along with the original BTC [6, 13, 14, 19, 62].

Griswold *et al.* [22] vary the quantization system so that a different set of moments (r'th and $2r$'th moments) is preserved. For each block, the BTC is tentatively applied with all values of r=1, 2, 3, 4, and the one with minimum MSE-value is selected. Quantization levels (a,b) are calculated already in the encoding phase, and therefore decoder does not need to know the choice of r.

Mitchell and Delp [37] apply the adaptation only to the coding of the bit plane:

Standard deviation:	**Bit plane coding:**	**Bit rate:**
High	Full resolution	16 bits
Medium	Prediction technique	8 bits
Low	Uniform block	0 bits

Standard deviation is separately coded from the bit plane and assumed to be sent first. The decoder thus knows implicitly whether the bit plane follows, and by how many bits.

Adaptive compression coding (ACC) is proposed by Nasiopoulos *et al.* [41], see Fig. 13. If the range of a 4*4 block is smaller than a given smoothness threshold (18, 30, or 42) it is represented by its average. Blocks that are above the threshold but below 120 are coded by absolute moment BTC. If the difference of quantization levels (b-a) is less than or equal to 20, the bit plane is coded by vector quantization including only edge blocks. Hierarchical decomposition is applied to the remaining high activity blocks. Small blocks are either copied, or compressed by the one-level quantization, see Section 4.3. Overhead information is needed to inform the decoder of the coding system used.

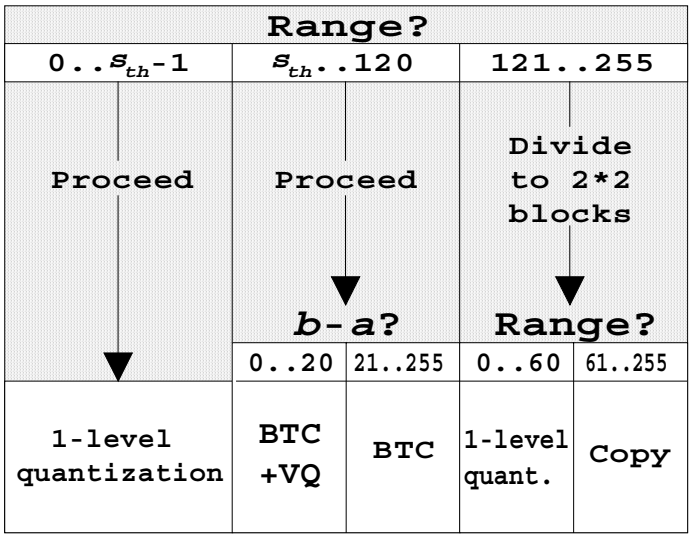


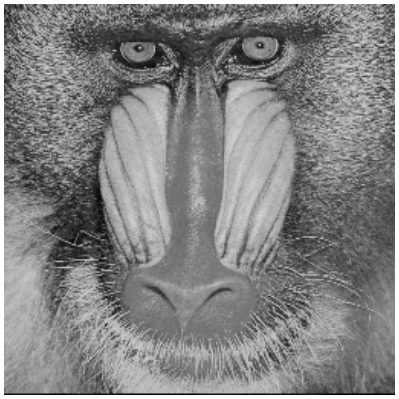
FIGURE 13. Adaptive compression coding system [41].

The adaptive compression coding system is indeed a very promising BTC variant. The coding scheme corresponds to most of the conclusions in Section 8. ACC, however, does not consider the coding of the quantization data (*a*,*b*) at all, and thus the results given in [41] do not quite compete with the ones that are given in Section 8.

8. EXPERIMENTAL TESTS

We built an *integrated BTC software* for testing the various methods presented in this survey. Our object has been usability of the software, so that any particular method could be easily utilized by a choice of the parameters. Main interest is focused on the bit rate (bpp), and the image quality. The latter is usually evaluated by the mean square error function (MSE). We fix the bit rate and then compare the MSE-values. Because of the highly procedural approach used in the implementation, we give almost no attention to the running times. The set of test images includes 18 different 8-bit gray-scale images of varying sizes from 256*256 to 720*576, see Fig. 14. There are two version of Lena: the first one is the green-component (G), of the original RGB-image, while Lenna is the luminance component (Y). The first one is the most frequently used image in the literature, and thus will be the primary test image in this survey.

Airplane (512 x 512)



Baboon (512 x 512)

Bigfoot (647 x 504)

Boats (720 x 576)

Bridge (256 x 256)

Camera (256 x 256)

Couple (256 x 256)

Girl (720 x 576)

Gold Hill (720 x 576)

FIGURE 14 a. Test images 1 to 9.

Lena (512 x 512)

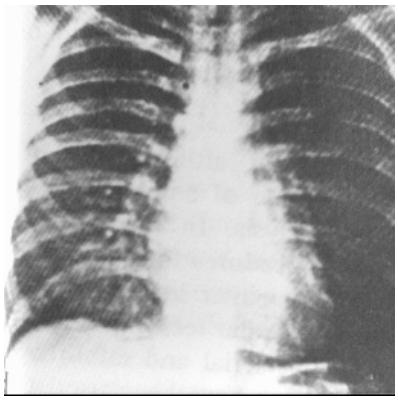
Lenna (512 x 512)

Peppers (512 x 512)

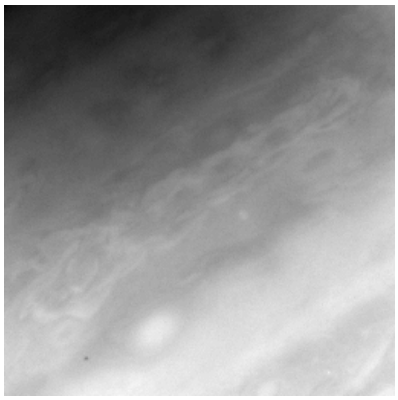
Sail (512 x 512)

Tiffany (512 x 512)

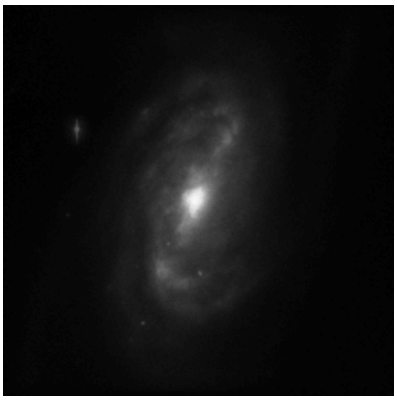
New Orleans (512 x 512)



Xray (512 x 512)



Jupiter (720 x 576)



Galaxy (512 x 512)

FIGURE 14 b. Test images 1 to 9.

8.1. QUANTIZATION METHOD

We start by comparing the different quantization methods of Section 4.1, see Table 3 for the selected methods. First we compress the test image Lena by all these methods. Quantization levels (a,b) are coded by 8+8 bits, and the bit plane is stored as such. The quality of the reconstructed images is shown in Table 4. The results indicate that the difference between the original 2nd moment preserving and the MSE-optimal quantization is remarkable. Magnifications of the reconstructed images are shown in Fig. 15. The coding artifacts are clearly seen in the magnifications; however, it is almost impossible to observe any subjective quality differences between the quantization methods.

The MSE-values are compared with the corresponding MSE-optimal quantization values in Fig. 16. The methods that preserve some set of moments are worst in the MSE-sense. The best candidates aside MSE-optimal quantization are the Lloyd and the one which uses *GB-heuristics* as a threshold. The latter is also easy to calculate, and is thus a good alternative to the MSE-optimal quantization. In the following sections, the MSE-optimal quantization will be the point of comparisons, unless otherwise noted.

TABLE 3. Quantization methods.

	2nd mom.	**3rd mom.**	**Abs.mom.**	**GB+mean**	**Lloyd**	**MSE-opt.**	**MAE-opt.**
x_{th}	mean	3rd moment	mean	$(x_{max}+x_{min})/2$	$(a+b)$ / 2	MSE-opt.	MAE-opt.
(a,b)	moment preserving	moment preserving	mean values	mean values	mean values	mean values	medians

TABLE 4. Fidelity function values for reconstructed Lena.

	2nd mom.	**3rd mom.**	**Abs.mom.**	**GB+mean**	**Lloyd**	**MSE-opt.**	**MAE-opt.**
MSE	43.76	42.11	40.51	37.05	36.83	35.54	40.21
MAE	3.88	3.89	3.67	3.68	3.78	3.54	3.34
SNR	31.72	31.89	32.06	32.44	32.47	32.62	32.09

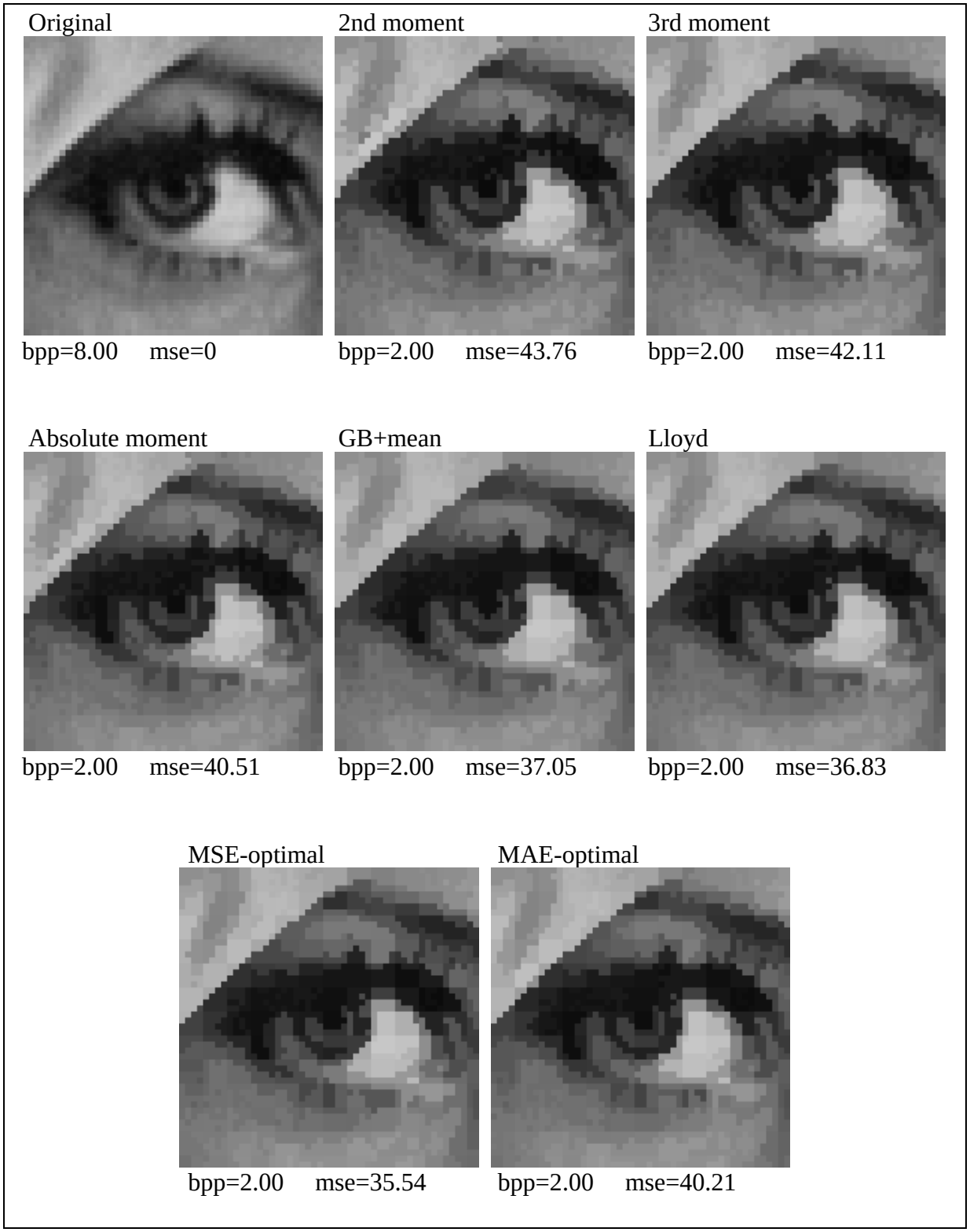


FIGURE 15. Eye of reconstructed Lena after different quantization methods.

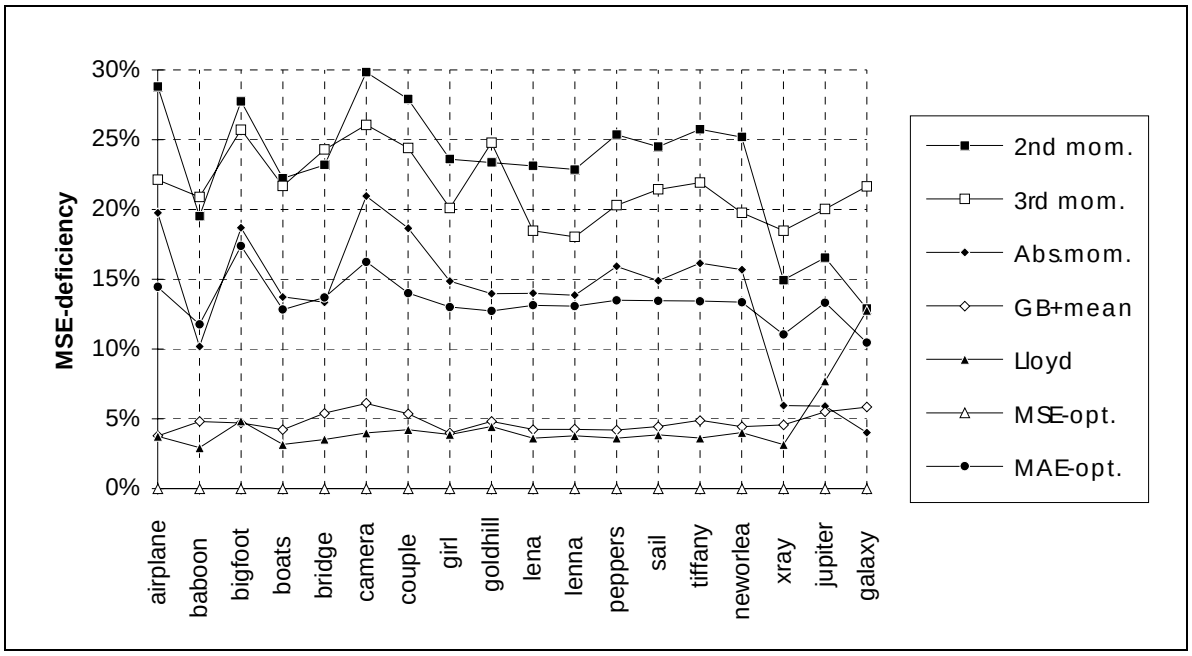
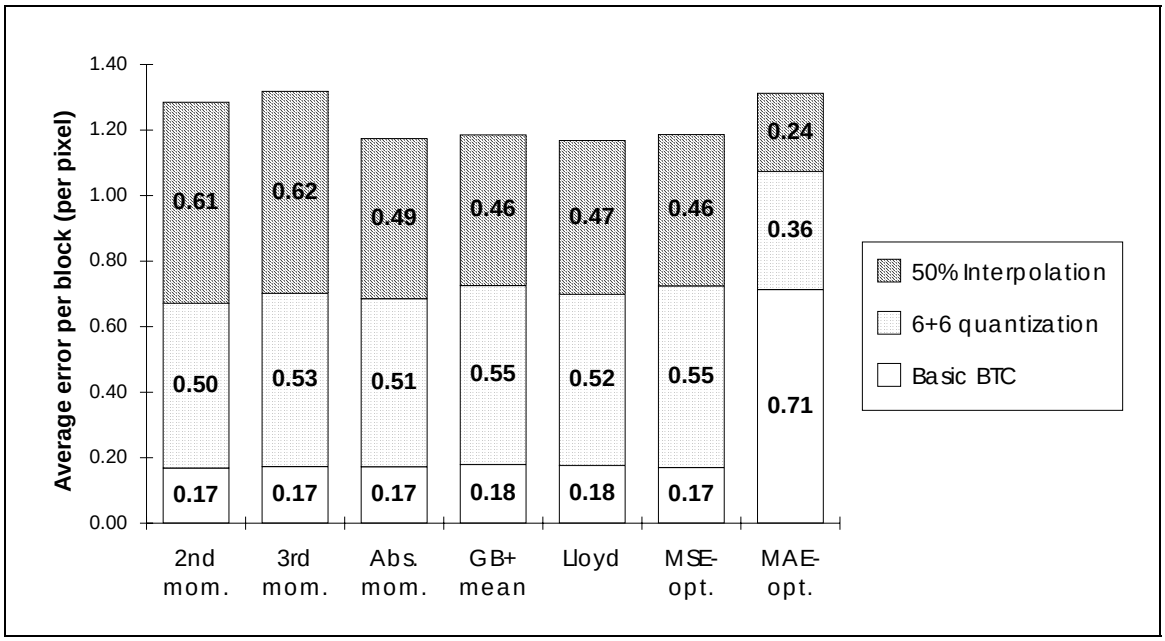


FIGURE 16. MSE-deficiency compared to the MSE-optimal quantization.

The basic idea behind the original BTC was to preserve certain moments. Despite the fact that the moment preserving quantizers can be written in closed form, the resulting quantization levels are only integer approximations of the formulas, and thus distorted by rounding errors. Fig. 17 and 18 demonstrate how well the two first moments are actually preserved in different quantization systems (white sections in the columns).

The illustrations show that the moments are not preserved precisely in any method. In fact, the moment preserving quantization does not preserve the first moment (mean) any better than the other quantization methods (except the MAE-optimal quantization). It will, however, preserve the second moment somewhat better, but the difference is hardly significant.

Moreover, if the quantization data and/or bit plane is further compressed, the moments are even less accurately preserved. In Fig. 17 and 18, there are also errors originating from quantization of (a,b) to 6+6 bits (gray sections in the columns) and from 50 % interpolation of the bit plane (dark sections in the columns). Despite all of that, the greatest deficiency of the moment preserving quantization is that it does not even try to preserve the moments beyond the block boundaries.



FIFURE 17. Error in the 1st moments of the reconstructed Lena.

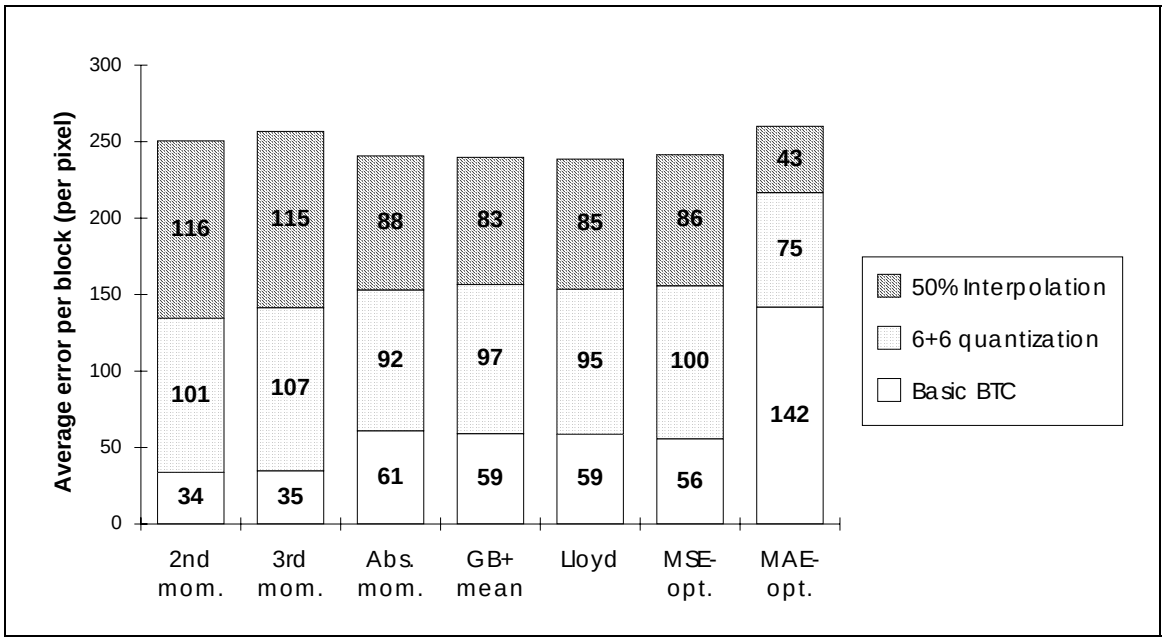


FIGURE 18. Error in the 2nd moments of the reconstructed Lena.

8.2. CODING THE QUANTIZATION DATA

There are several choices of quantization data to be coded, but only (*a*,*b*) is independent of the selected quantization method. It also minimizes the computation required by the decoder. The pair (*a*,*b*) contains redundancy but it can easily be removed by representing the

quantization data by (a,b-a). On the other hand, the use of ($\bar{x}$,σ), or ($\bar{x}$,α) takes advantage of the *skewness information* included in the bit plane, and therefore a less accurate representation would be possible with the same MSE. However, the extra degdaration generated by the rounding errors causes the overall performance for (a,b) to be slightly better [16] in the sense of bit rate and MSE.

Next we will show the effect of the number of bits allocated to (a,b) on the MSE-value. Table 5 summarizes the results with a set of test images. It turns out that the number of bits can be reduced to 6+6 with only a small increase in MSE. Similar results hold for the other test images, too.

TABLE 5. MSE-values of different accuracy.

No. of bits:	**8+8**	**7+7**	**6+6**	**5+5**	**4+4**	**3+3**	**2+2**
Airplane	34.51	34.78	35.78	39.86	55.67	130.91	342.96
Baboon	118.22	118.48	119.48	123.50	139.67	204.85	431.69
Boats	27.97	28.24	29.25	33.22	48.24	109.46	445.86
Bridge	87.67	87.92	88.92	92.88	109.01	173.86	488.38
Lena	35.54	35.80	36.80	40.74	56.57	118.29	376.27
New Orleans	14.21	14.47	15.46	19.41	34.96	94.60	432.14
X-ray	14.69	14.98	16.16	21.23	43.01	140.17	892.06
Galaxy	0.82	1.10	1.99	6.63	28.66	76.56	187.32

Several promising methods for coding the quantization data were introduced in Section 4.2. We consider here only two of them, Entropy coding (EC) and FELICS, see Table 6. In FELICS the intrablock redundancy is not properly taken into consideration. In spite of this, the results are surprisingly close to EC, and sometimes even better. Moreover, the implementation of FELICS is very simple, and thus it is a strong candidate for coding the quantization data.

TABLE 6. Performance of various quantization data coding schemes.

Accuracy:	**8+8 bits**		**6+6 bits**	
Coding method:	**EC**	**FELICS**	**EC**	**FELICS**
Airplane	9.70	9.57	6.34	6.30
Baboon	12.26	12.21	8.03	8.03
Boats	9.60	9.49	6.38	6.26
Bridge	13.02	12.54	8.51	8.45
Lena	10.58	10.69	6.69	6.75
New Orleans	10.29	9.82	6.50	6.21
X-ray	8.98	9.54	5.65	6.11
Galaxy	4.08	5.26	2.94	4.26

8.3. CODING THE BIT PLANE

We implemented four of the bit plane coding methods of Section 4.3. The interpolative BTC proposed by Zeng [76, 77] includes two levels of interpolation in which 50 % or 25 % of bits of the bit planes are stored. We modified the 50 % variant so that a 75 % interpolation scheme was obtained, too.

Other methods considered are the 1-dimensional median filtering [3], vector quantization [61], and the prediction technique [37]. They may not be the best representatives of their kind, but we try to get an idea of their power relative to each other. In the VQ implementation, we experimented several different search techniques, but we report here only the results for the extensive (MSE-optimal) search. The codebook was generated by the generalized Lloyd algorithm, and the initial codebook was selected heuristically as the 256 most frequent matrices, as proposed in [17].

The MSE-values for different methods and test images are summarized in Table 7. Magnified parts of the reconstructed Lena are shown in Fig. 19. The interpolative method (with 50 % and 75 % schemes) was the most promising in the MSE-sense in our tests.

TABLE 7. MSE-values according to different bit plane coding methods.

Method:	**Storing**	**Int-75%**	**Int-50%**	**Int-25%**	**Filtering**	**VQ**	**Prediction**
bpp:	*1.00*	*0.75*	*0.50*	*0.25*	*0.75*	*0.50*	*0.50*
Airplane	34.51	37.11	39.60	75.43	40.14	42.72	120.62
Baboon	118.22	169.60	217.80	387.47	170.83	234.54	417.59
Boats	27.97	30.99	34.13	76.61	39.91	35.61	90.00
Bridge	87.67	112.96	138.40	247.74	127.52	141.94	271.34
Lena	35.54	38.85	41.93	63.83	46.46	48.29	101.48
New Orleans	14.21	15.73	17.31	27.65	17.50	19.84	42.49
Xray	14.69	17.95	21.33	48.76	20.61	18.47	33.05
Galaxy	0.82	1.11	1.39	1.77	1.28	1.33	1.95

There is an obvious way to improve the VQ and the filtering schemes. The quantization level values (a,b) are calculated according to the original partition of a block, however, the VQ or median filtering stage performs a new (indirect) partition. Therefore it makes sense to recalculate (a,b) according the new partition. In fact, the VQ search should be done on the basis of the original block, and not on the basis of the bit plane given by a quantization method which is ignored later on. This gives an order of 10 % improvement in the MSE, see [17].

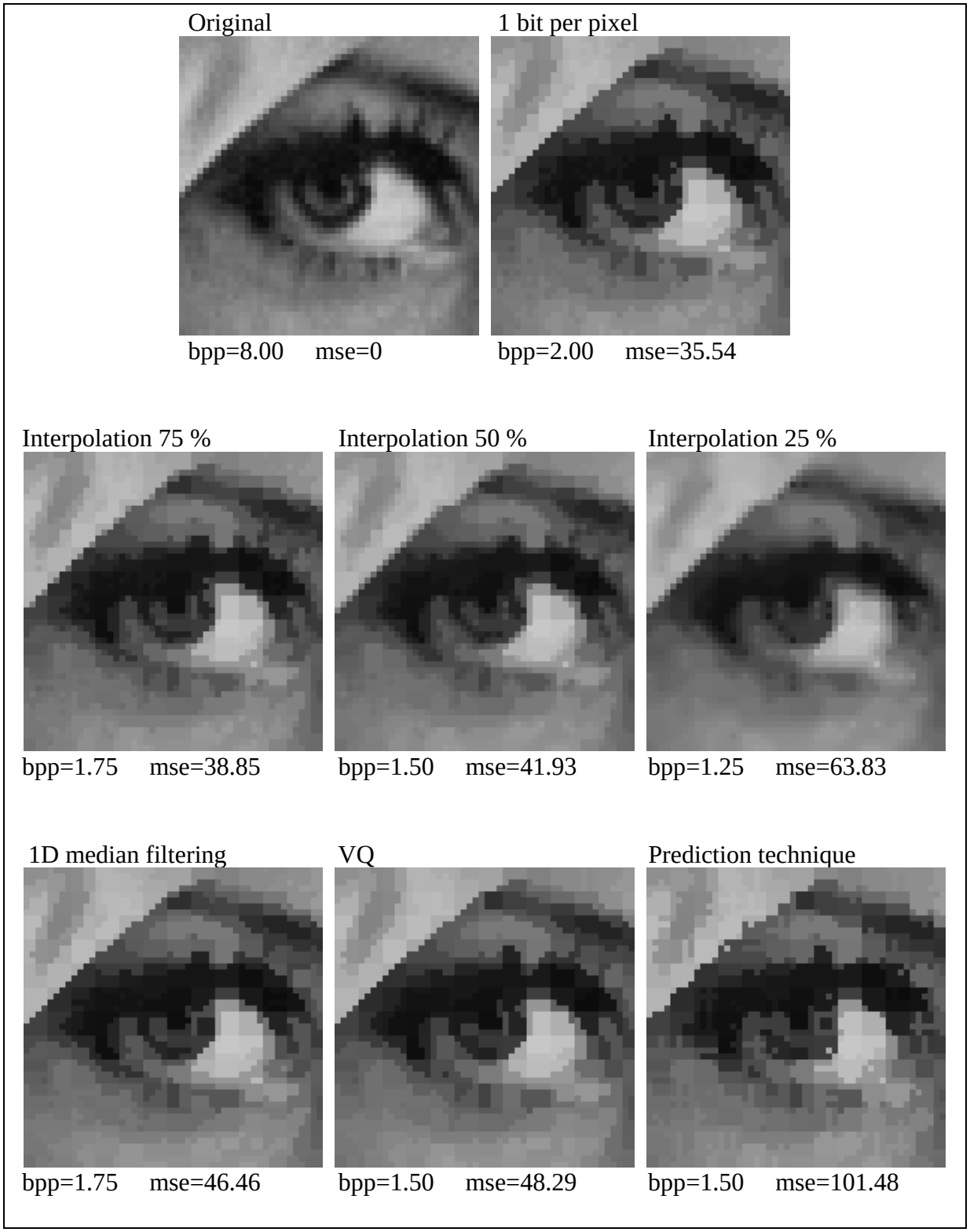


FIGURE 19. Eye of Lena with different bit plane coding methods.

8.4. DECOMPOSITION OF THE IMAGE

In most BTC variants the block size is fixed to 4*4, however, any size and shape of blocks can be applied. Here we consider only square-like blocks. The size of a block has an immediate effect on the bit rate and the image quality, see Table 8.

TABLE 8. Compression results for Lena with different block sizes.

Block size	**2*2**	**3*3**	**4*4**	**5*5**	**6*6**	**7*7**	**8*8**
MSE	7.69	23.56	35.54	46.44	56.17	64.62	72.36
bpp	5.00	2.79	2.00	1.66	1.47	1.36	1.25

In hierarchical decomposition, we apply the quadtree division and a block size which is a power of two. Different selections of the block sizes are compared in Fig. 20 and 21. The smallest block size is either 4*4 or 2*2, and it has a major effect both on the bit rate and the image quality. The largest block size can vary from 4*4 to 32*32 and its influence is only marginal for most of the images. It is also shown that the images containing more homogenous areas are easiest for the hierarchical BTC and thus gives the lowest bit rates.

The criterion for dividing a block can be either standard deviation, variance, or range of the block. Variance was used with a threshold of v_{th}=30 for the results in Fig. 20 and 21. A different threshold value can be given for each size of block, and it might be useful to apply a smaller threshold value to the larger blocks than to the smaller ones. In particular, the threshold for "4*4$\rightarrow$2*2" division can be used to adjust the method towards better image quality or lower bit rate.

Hierarchical decomposition is easy to implement, but the variable block size makes the coding of the bit plane somewhat more complicated. VQ implementation, for example, is tuned to 4*4-blocks only and therefore it excludes the use of 2*2-blocks. For larger blocks (32*32, 16*16, 8*8) the problem is solved by dividing the bit plane until 4*4 size is reached and then they are coded by the 4*4-block method. Interpolation is implemented as a two-phase algorithm, so it is independent of block size.

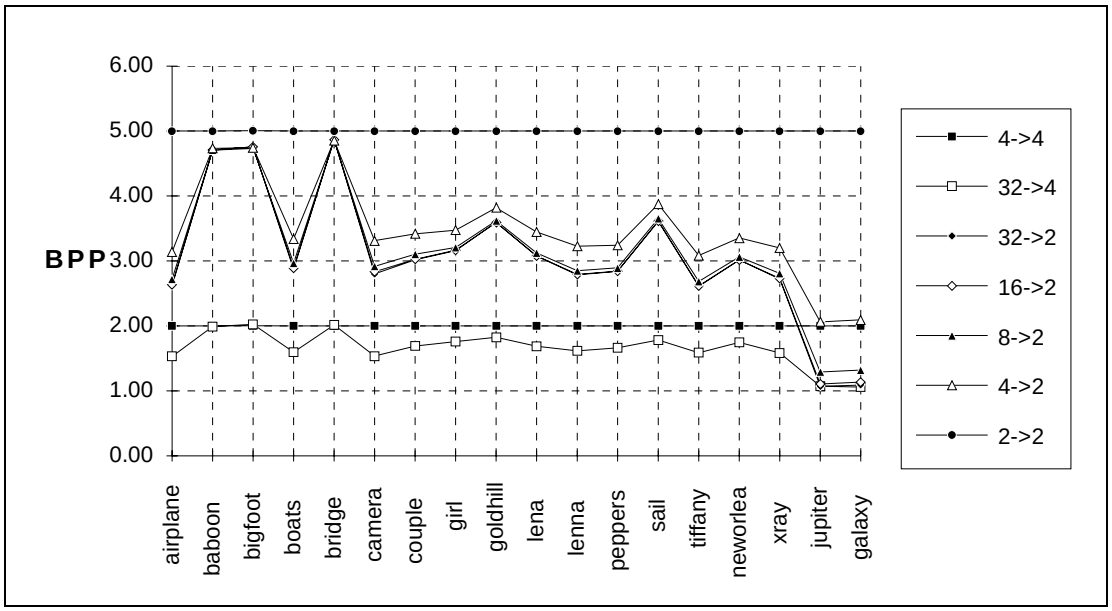


FIGURE 20. Compression efficiency versus block size.

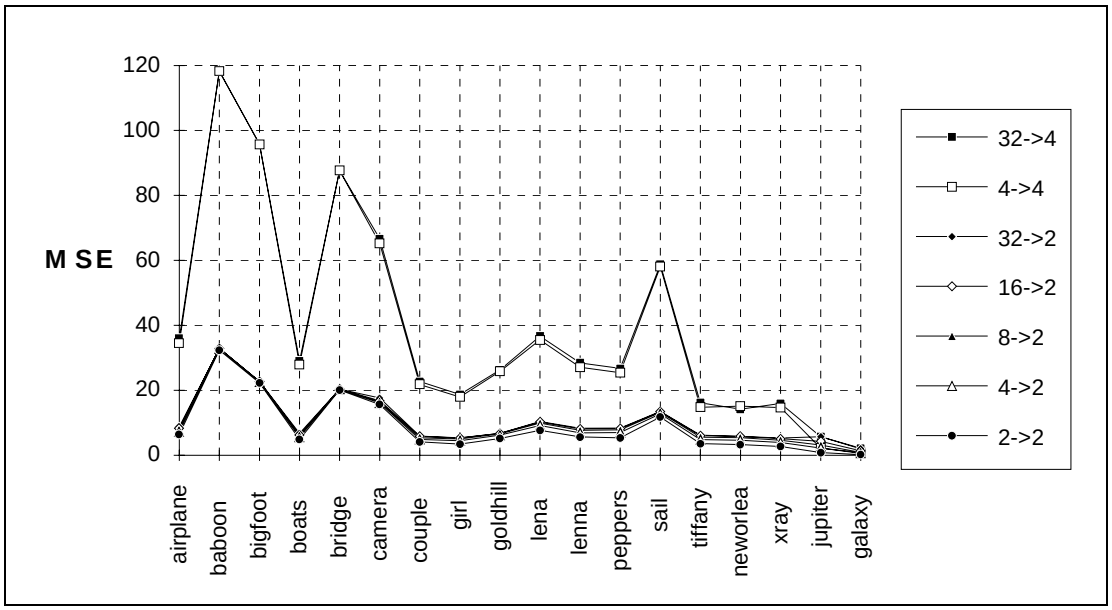
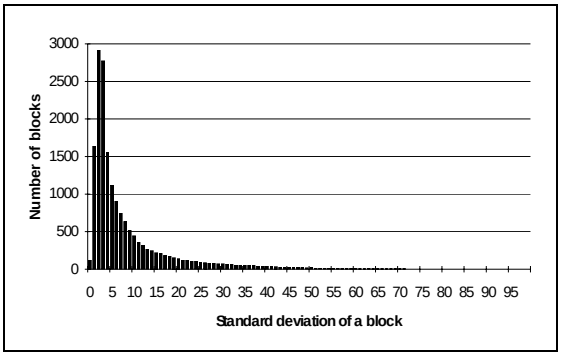


FIGURE 21. Image quality versus block size.

8.5. BLOCK TO BLOCK ADAPTIVE BTC

In Adaptive BTC, the blocks are classified according to the contrast of a block, and a different coding method is applied to different types of blocks. Fig. 22 and 23 illustrates the distribution of the blocks and their average MSE values (per pixel) according to standard deviation. The higher the variance, the greater the MSE contributed by the block. Even if there are only few such blocks, their net effect is remarkable: more than 50 % of the MSE-value originates from the 10 % of the high variance blocks. This indicates that another

compression method might be used for these. One can also reduce the number of the high variance blocks by the use of hierarchical decomposition, as was done in the previous section.



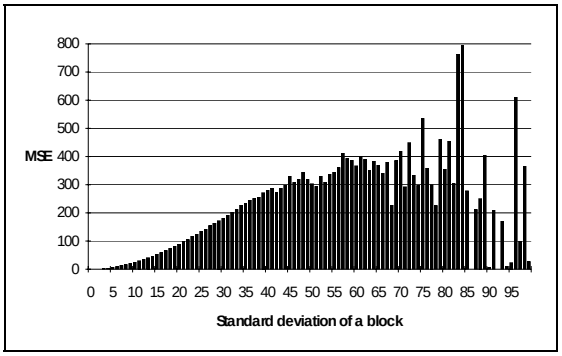


FIGURE 22. Distribution of 4*4 blocks. **FIGURE 23.** Average MSE of different blocks.

Fig. 24 illustrates the (absolute) additional MSE (compared to the basic BTC) originating from the coding of the bit plane. Even if the increase of MSE is greater in high variance blocks, there is no evidence that the variance of a block would reflect to the **relative** increase in MSE. A more important observation is that the superiority between any two given bit plane coding method is independent of the contrast of a block. The consequence of this is that there is no clear advantage of block to block adaptation in the bit plane coding method.

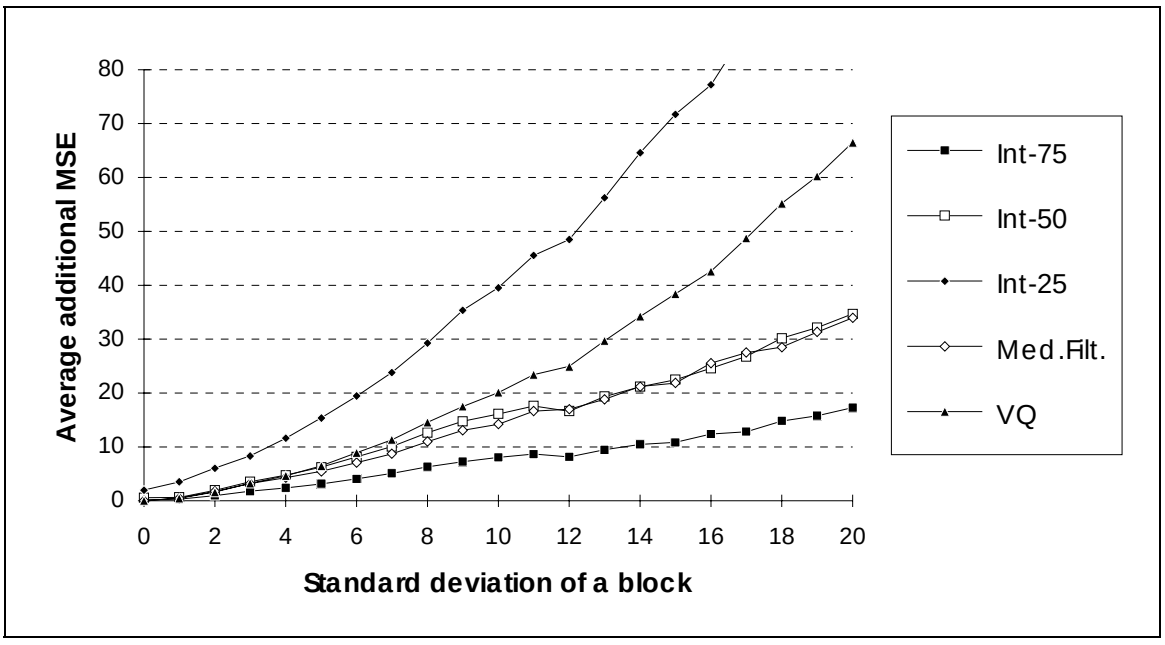


FIGURE 24. Additional MSE contributed by bit plane coding according to the block types.

We have restricted our study of the adaptive schemes to the bit plane coding only. It would still be possible to extend the adaptation to the other parts of the algorithm also, and even other related coding systems, see Sections 6 and 7.

8.6. COMPARISON TO JPEG

Next we will link together the best ideas of the BTC algorithm. Since the three phases of the BTC were studied separately, there is no guarantee that the combined algorithm is optimal in any sense, nor the best that could be constructed from these elements. In spite of that, we will compare it against JPEG. This is done by first fixing a certain bit rate and then comparing the MSE-values of these two methods.

The combined BTC algorithm consists of the components given in Table 9. Hierarchical decomposition is fully utilized: The corresponding minimum and maximum block sizes are 2*2 and 32*32. Standard deviation (σ) is used as a threshold criterion and is set to 6 for other level transitions than "4*4->2*2", for which the threshold is left as an adjusting parameter.

TABLE 9. Elements of the combined BTC algorithm.

Phase:	**Methods:**
Block size	• Hierarchical (32→2)
Quantization system	• GB + means
Quantization data	• (*a*,*b*)
Number of bits	• 6+6
Coding the quant. data	• FELICS
Coding the bit plane	• 1 bpp • Interpolation • Skipping the bit plane

The bit plane is either stored as such, or its size is reduced by the interpolation technique. The level of interpolation is chosen according to the desired bit rate level. It will be referred by the proportion of coded bits which is either 50, 75, or 100 %. Bit planes of uniform blocks are skipped and the 1-bit quantization is applied to these blocks. The uniform block skipping threshold is used as a fine-tuning parameter.

Fig. 25 illustrates the performance of the combined BTC algorithm for Lena. The tests were performed by varying the threshold of the hierarchical decomposition from σ=0 to 50. Each curve represents one of the selected interpolation levels. In this way we are able to cover reasonably well the bit rates from 1.0 to 2.0 bits per pixel. Better image qualities are still possible at the cost of decreasing compression efficiency, but the increase of the bit rate serves the purpose only up to a limit. By the use of lossless image compression methods, such as lossless JPEG [46, 57] or FELICS [25], one can achieve bit rates of about 4.5 bpp (for Lena). At the other end, the image quality will decrease rapidly if very low bit rates are desired.

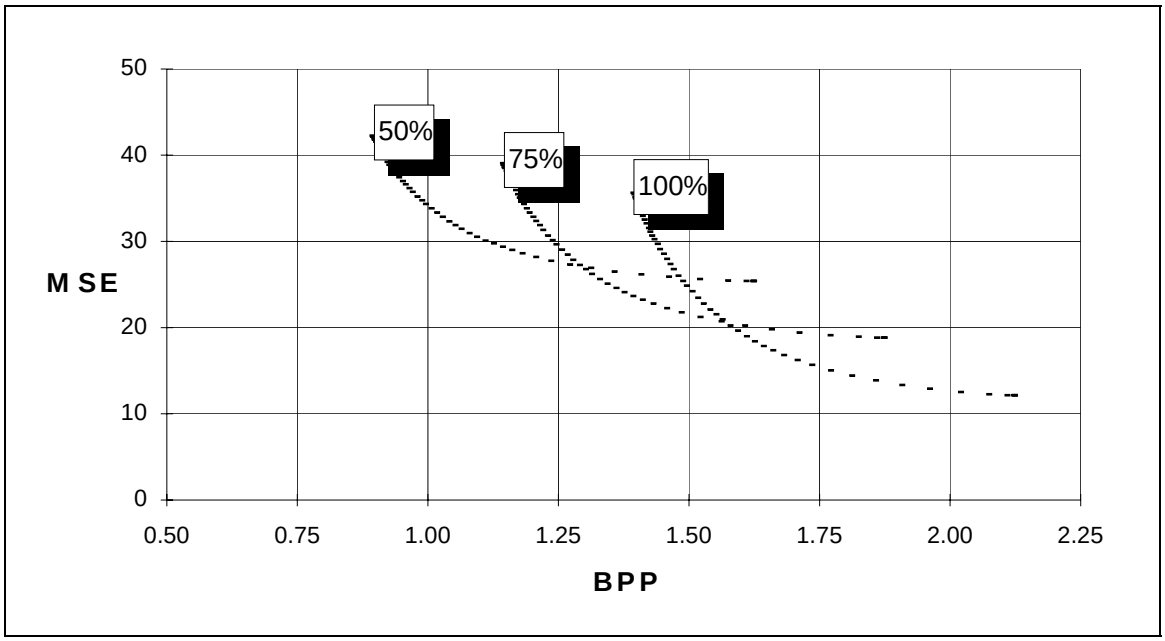


FIGURE 25. Performance of the combined BTC algorithm for Lena.

A comparison of JPEG and the combined BTC is summarized in Fig. 26-29. Here we use the same parameters that were used for the test results in Fig. 25 with one exception: we performed two different tests. In the first one, we used the skipping threshold $\sigma=0$, and in the second $\sigma=5$. The final curves of Fig. 26-29 are assembled from the results of these two so that they correspond to the best MSE-values for each bits per pixel selection. The non-continuation of the curve in Fig. 29 originates from the nature of the image Galaxy. It consists entirely of low contrast regions, and therefore the hierarchical decomposition cannot be used for adjusting its performance as much as is desirable (see Fig. 20 and 21 of Section 8.4).

In our tests, JPEG clearly outperforms BTC (see Fig. 30). At the bit rate of 2.0, the MSE of JPEG is 65 % from the MSE of BTC (for Lena), and at the bit rate of 1.0 bpp, it is only 53 %. The difference in the visual quality is yet unclear. Small differences in the MSE-value are not necessarily reflected in the visual quality and one must keep in mind, that the distortions caused by these two methods are not alike. Moreover, distortion is almost unnoticeable if one looks at the images in their natural sizes. Magnifications of Lena are shown in Fig. 31. We want to emphasize that there is a definite need for a better objective quality measure than MSE.

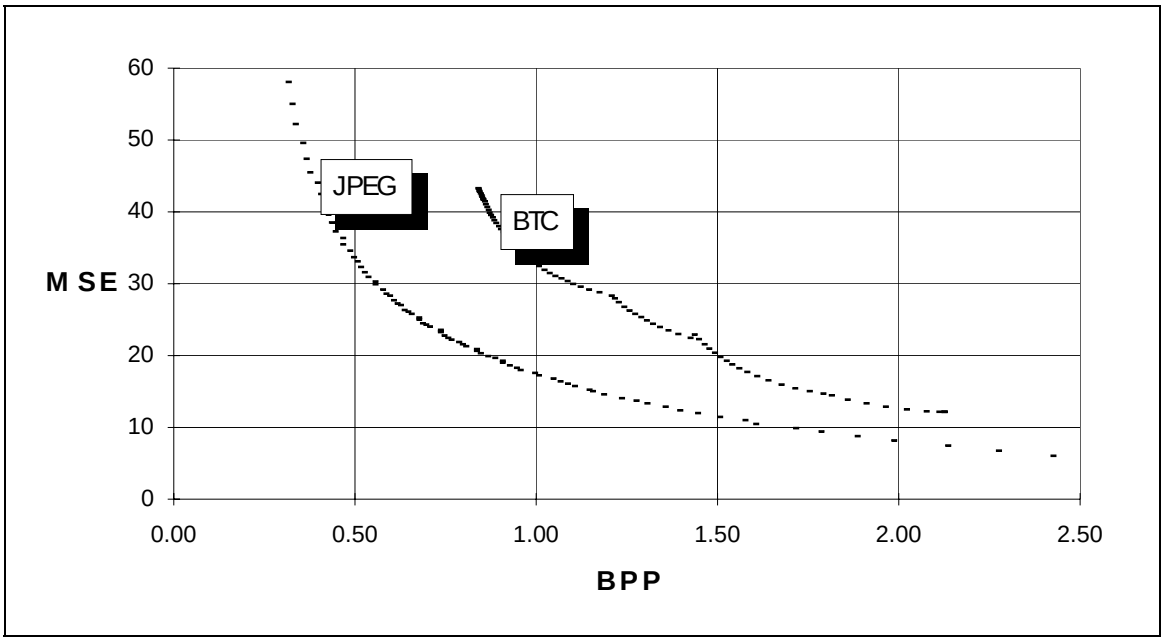


FIGURE 26. Comparison of JPEG versus BTC with test image Lena.

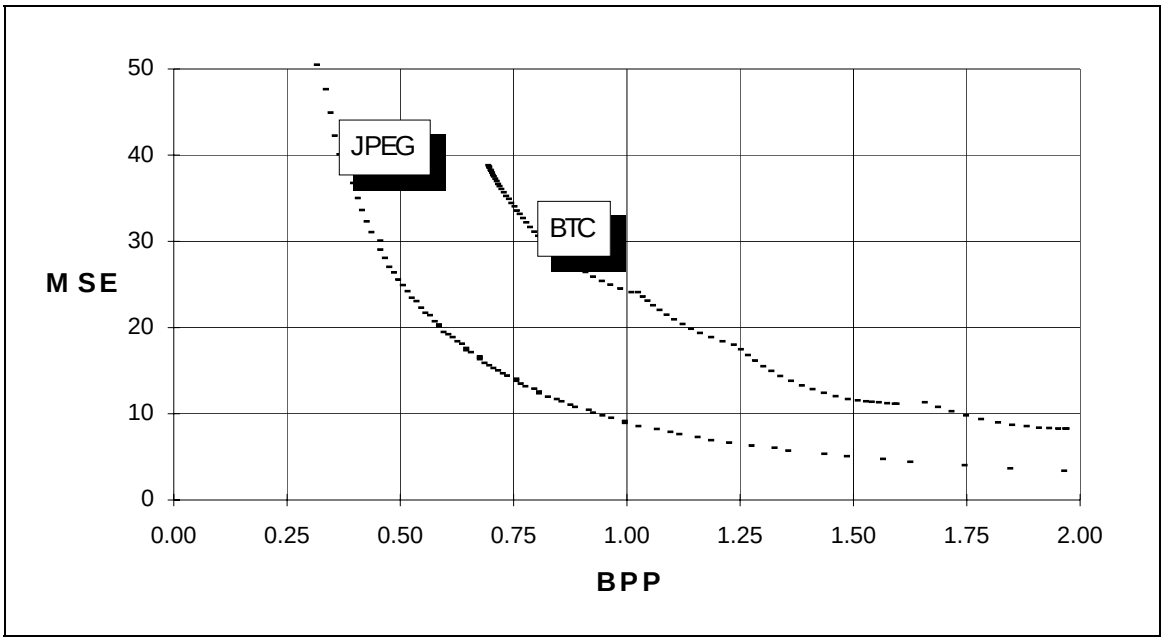


FIGURE 27. Comparison of JPEG versus BTC with test image Boats.

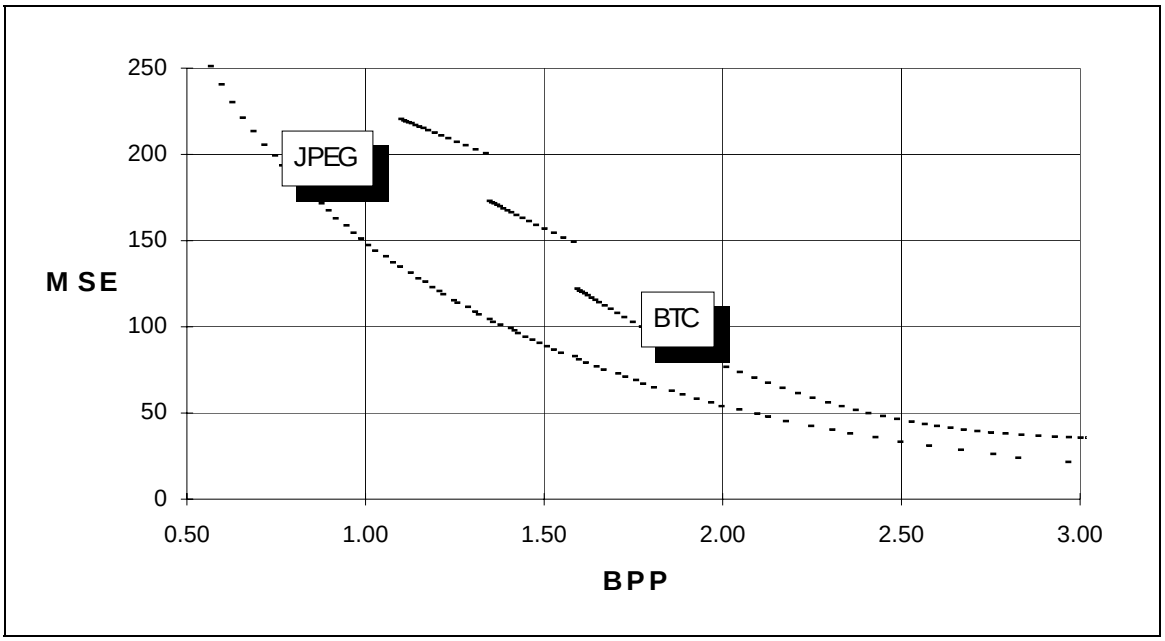


FIGURE 28. Comparison of JPEG versus BTC with test image Baboon.

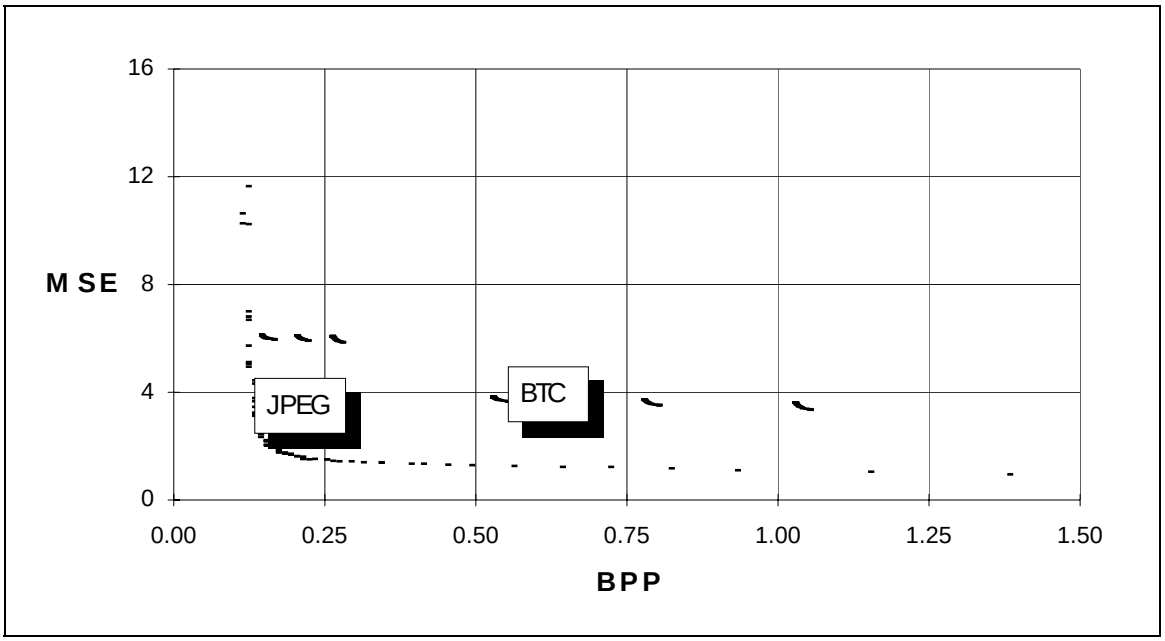


FIGURE 29. Comparison of JPEG versus BTC with test image Galaxy.

J P E G


bpp=1.00 mse=17.26


bpp=1.50 mse=11.48


bpp=2.00 mse=8.18

B T C


bpp=1.00 mse=32.47


bpp=1.50 mse=19.81


bpp=2.00 mse=12.52

FIGURE 30. Part of Lena when coded by JPEG and BTC.

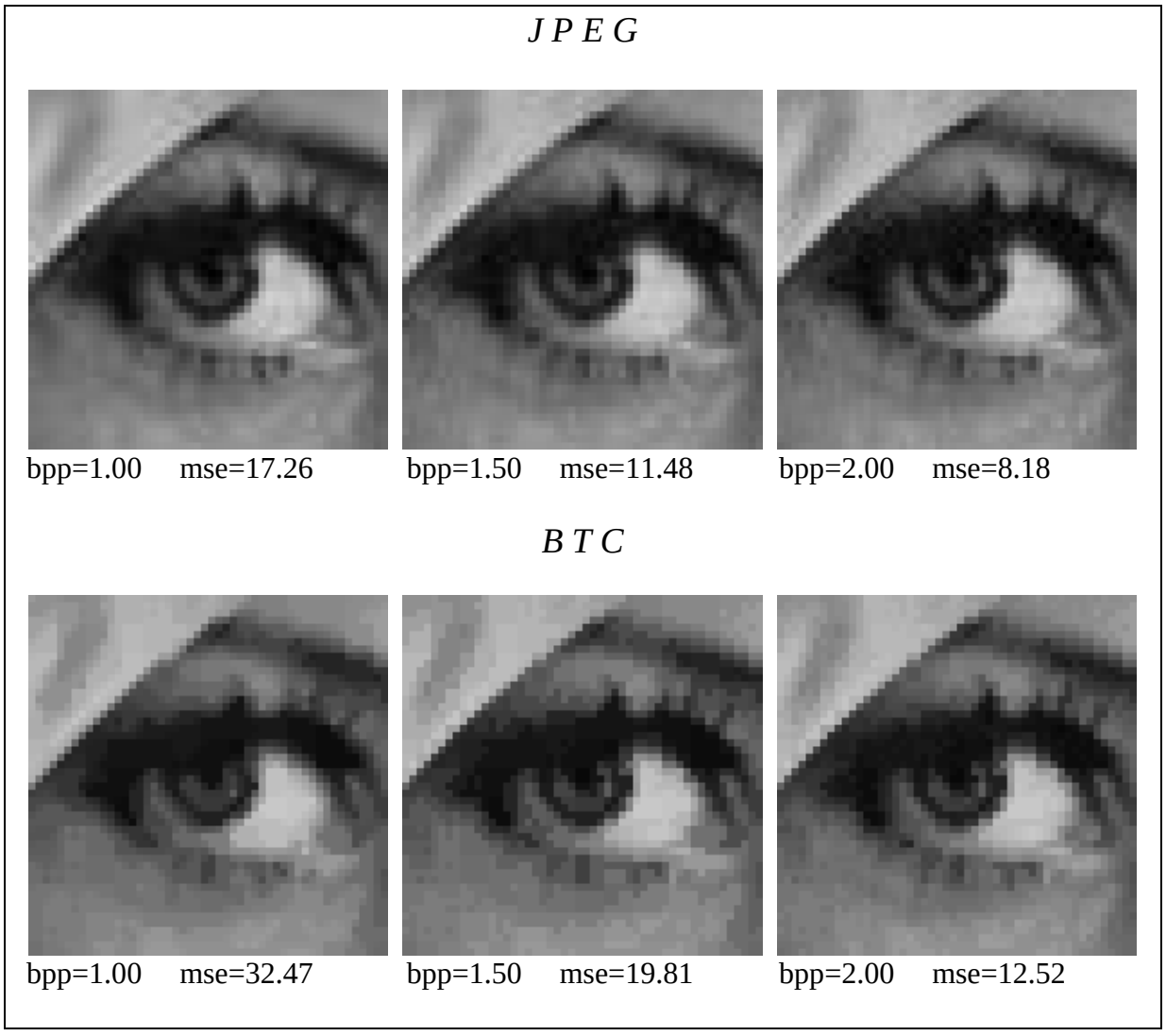


FIGURE 31. Eye of Lena when coded by JPEG and BTC.

9. CONCLUSIONS

We have done a comparative study of the variants of the block truncation coding. Our main observation was that one can remarkably improve the overall performance of BTC by the use of hierarchical decomposition, interpolation and FELICS coding. The most important change for improving the image quality was the use of 2*2-blocks in the high contrast regions. When compared to the performance of JPEG, the new combined block truncation coding is still weak if the image quality is measured by the mean square error.

There are still other aspects which make BTC a useful alternative for JPEG in some practical applications. All the selected algorithms are quite simple and easy to implement. The integration makes the implementation somewhat more complicated, but it is nevertheless probably much simpler than the JPEG implementation. Since we did not perform any throughput analysis, we can only estimate the running times.

The encoding phase of BTC is very fast, and the decoding phase is even faster. (Though it has to be said that the JPEG software used was not slow either.) One can maximize the throughput at the decoding phase by the use of (a,b) as the quantization data. It could be further improved by the use of vector quantization instead of interpolation. This would load the encoding more, but the computational demands would be much less at the decoding phase. FELICS coding is a fast and practical alternative to other lossless schemes, and therefore it is a good choice here. Moreover, it is utilized only twice per block. The hierarchical decomposition has only a small effect on the throughput, but is directly reflected in the number of blocks to be coded.

By the use of hierarchical decomposition and FELICS coding, one can reduce the interblock redundancy. This will eliminate direct access to the decoded file and thus there is no longer guarantee of fault tolerance.

REFERENCES

[1] **Ahmed N., Natarajan T., Rao K.R.** (1974) Discrete Cosine Transform. *IEEE Transactions on Computers*, **C-23**, 90-93.

[2] **Alcaim A., Oliveira L.V.** (1992) Vector Quantization of the Side Information in BTC Image Coding. *Proc. ICCS/ICITA 92*, Singapore, **1**, 345-349.

[3] **Arce G., Gallagher N.C. Jr.** (1983) BTC Image Coding Using Median Filter Roots. *IEEE Transactions on Communications*, **31**, 784-793.

[4] **CCITT Document No. 420** (1988) Adaptive Discrete Cosine Transform Coding Scheme for Still Image Telecommunications Services.

[5] **Chen W.H., Pratt W.** (1984) Scene Adaptive Coder. *IEEE Transactions on Communications*, **32**, 225-232.

[6] **Cheng S.-C., Tsai W.-H.** (1993) Image Compression Using Adaptive Multilevel Block Truncation Coding. *Journal of Visual Communication and Image Representation*, **4**, 225-241.

[7] **Colomb S.W.** (1966) Run-Length Encodings. *IEEE Transactions on Information Theory*, **12**, 399-401.

[8] **Connor D.J., Pease R.F., Scholes W.G.** (1971) Television Coding Using Two-Dimensional Spatial Prediction. *Bell Systems Technical Journal*, **50**, 1049-1061.

[9] **Delp E.J., Mitchell O.R.** (1979) Some Aspects of Moment Preserving Quantizers. *IEEE International Conference Communications*, (ICC'79), **1**, 10-14, 7.2.1-7.2.5..

[10] **Delp E.J., Mitchell O.R.** (1979) Image Coding Using Block Truncation Coding. *IEEE Transactions on Communications*, **27**, 1335-1342.

[11] **Delp E.J., Mitchell O.R.** (1991) The Use of Block Truncation Coding in DPCM Image Coding. *IEEE Transactions on Signal Processing*, **39**, 967-971.

[12] **Delp E.J., Mitchell O.R.** (1991) Moment Preserving Quantization. *IEEE Transactions on Communications*, **39**, 1549-1558.

[13] **DeWitte J., Ronsin J.** (1983) Original Block coding Scheme for Low Bit Rate Image Transmission. *Signal Processing II: Theories and Applications, Proc. EUSIPCO-83.* Elsevier, Amsterdam, 143-146.

[14] **Efrati N., Liciztin H., Mitchell H.B.** (1991) Classified Block Truncation Coding-Vector Quantization: an Edge Sensitive Image Compression Algorithm. *Signal Processing, Image Commun.*, **3**, 275-283.

[15] **Eskicioglu A.M., Fisher P.S.** (1993) A Survey of Quality Measures for Gray Scale Image Compression. *Proceedings Space and Earth Science Data Compression Workshop*, Snowbird, Utah, 49-61.

[16] **Fränti P., Nevalainen O.** (1993) Block Truncation Coding with Entropy Coding. (submitted to *IEEE Transactions on Communications*)

[17] **Fränti P., Kaukoranta T., Nevalainen O.** (1994) A New Approach to BTC-VQ Image Compression System. *Proc. Picture Coding Symposium*, Sacramento, CA.

[18] **Gersho A., Gray R.M.** (1992) Vector Quantization and Signal Compression. *Kluwer Academic Publishers*.

[19] **Goeddel T., Bass S.** (1981) A Two Dimensional Quantizer for Coding Digital Imagery. *IEEE Transactions on Communications*, **29**, 60-67.

[20] **Goldberg M., Bouncher P.R., Shlien S.** (1986) Image Compression Using Adaptive Vector Quantization. *IEEE Transactions on Communications*, **34**, 180-187.

[21] **Gonzalez R.C., Woods R.E.** (1992) *Digital Image Processing*. Addison-Wesley.

[22] **Griswold N., Halverson D., Wise G.** (1987) A Note on Adaptive Block Truncation Coding for Image Processing. *IEEE Transactions on Accoustics, Speech and Signal Processing*, **35**, 1201-1203.

[23] **Halverson D., Griswold N., Wise G.** (1984) A Generalized Block Truncation Coding Algorithm for Image Compression. *IEEE Transactions on Accoustics, Speech and Signal Processing*, **32**, 664-668.

[24] **Healy D., Mitchell O.R.** (1981) Digital Bandwidth Compression Using Block Truncation Coding. *IEEE Transactions on Communications*, **29**, 1809-1817.

[25] **Howard P.G., Vitter J.S.** (1993) Fast and Efficient Lossless Image Compression. *IEEE Proceedings Data Compression conference*, Snowbird, Utah, 351-360.

[26] **Huang L., Bijaoui A.** (1991) An Efficient Image Compression Algorithm Without Distortion. *Pattern Recognition Letters*, **12**, 69-72.

[27] **Kamel M., Sun C. Guan L.** (1991) Image Compression by Variable Block Truncation Coding with Optimal Threshold. *IEEE Transactions on Signal Processing*, **39**, 208-212.

[28] **Kruger A.** (1992) Block Truncation Compression. *Dr. Dobb's Journal*, 48-55 and 104-106.

[29] **Kunt M., Bénard M., Leonardi R.** (1987) Recent results in high-Compression Image Coding. *IEEE Transactions on Circuits and Systems*, **34**, 1306-1336.

[30] **Kunt M., Ikonomopolous A., Kocher M.** (1985) Second Generation Image Coding Technique. *Proceedings of the IEEE*, **73**, 549-574.

[31] **Lema M.D., Mitchell O.R.** (1984) Absolute Moment Block Truncation Coding and its Application to Color Images. *IEEE Transactions on Communications*, **32**, 1148-1157.

[32] **Linde Y., Buzo A., Gray R.M.** (1980) An Algorithm for Vector Quantizer Design. *IEEE Transactions on Communications*, **28**, 84-95.

[33] **Lloyd S.P.** (1982) Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, **28**, 129-137.

[34] **Lu W.W., Gough M., Davies P.** (1991) Scientific Data Compression for Space: a Modified Block Truncation Coding Algorithm. *Proc. SPIE*, **1470**, 197-205.

[35] **Max J.** (1960) Quantizing for Minimum Distortion. *IRE Transactions on Information Theory*, **6**, 7-12.

[36] **Mitchell O.R., Bass S.C., Delp E.J., Goeddel T.W., Huang T.S.** (1980) Image Coding for Photo Analysis. *Proc. Soc. Inform. Display*, **21**, 279-292.

[37] **Mitchell O.R., Delp E.J.** (1980) Multilevel Graphics Representation Using Block Truncation Coding. *Proceedings of the IEEE*, **68**, 868-873.

[38] **Mitchell H.B., Dorfan M.** (1992) Block Truncation Coding Using Hopfield Neural Network. *Electronics Letters*, **28**, 2144-2145.

[39] **Moffat A.** (1991) Two-level Context Based Compression of Binary Images. *IEEE Proceedings Data Compression conference*, Snowbird, Utah, 382-391.

[40] **Monet P., Dubois E.** (1993) Block Adaptive Quantization of Images. *IEEE Transactions on Communications*, **41**, 303-306.

[41] **Nasiopoulos P., Ward R., Morse D.** (1991) Adaptive Compression Coding. *IEEE Transactions on Communications*, **39**, 1245-1254.

[42] **Nasrabadi N.M., King R.A.** (1988) Image Coding Using Vector quantization: A Review. *IEEE Transactions on Communications*, **36**, 957-971.

[43] **Neejärvi J.** (1992) *Analysis and Extensions of FIR-Median Hybrid and Morphological Filters*. Tampere University of Technology, Ph.D. Thesis.

[44] **Netravali A.N., Haskell B.G.** (1988) *Digital Pictures*. Plenum Press.

[45] **Pennebaker W.B., Mitchell J.L., Langdon G.G., Arps R.B.** (1988) An Overview of the Basic Principles of the Q-coder. *IBM Journal of Research and Development*, **32**, 717-726.

[46] **Pennebaker W.B., Mitchell J.L.** (1993) *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, New York.

[47] **Qiu G., Varley M.R., Terrell T.J.** (1991) Improved Block Truncation Coding Using Hopfield Neural Network. *Electronics Letters*, **27**, 1924-1926.

[48] **Qiu G., Varley M.R., Terrell T.J.** (1992) Image Compression by a Variable Size BTC Using Hopfield Neural Networks. *Colloquiumon 'Neural Networks for Image Processing. Applications'*, London, 10/1-6.

[49] **Rabbani M., Melnychuck P.W.** (1992) Conditioning Context for the Arithmetic Coding of Bit Planes. *IEEE Transactions on Signal Processing*, **40**, 232-236.

[50] **Rabbani M., Jones P.W.** (1991) *Digital Image Compression Techniques*. Bellingham, USA, SPIE Optical Engineering Press.

[51] **Rao K.R., Yip P.** (1990) *Discrete Cosine Transform*. New York, Academic Press.

[52] **Rice R.F.** (1979) Some Practical Universal Noiseless Coding Techniques. Jet Propulsion Laboratory, JPL Publication 79-22, Pasadena, CA.

[53] **Ronsin J., DeWitte J.** (1982) Adaptive Block Truncation Coding Scheme Using an Edge Following Algorithm. *Proc. ICASSP 82*, Paris, 1235-1238.

[54] **Roy J., Nasrabadi M.** (1991) Hierarchical Block Truncation Coding. *Optical Engineering*, **30**, 551-556.

[55] **Samet H.** (1990) *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Reading, MA.

[56] **Sharma D.J., Netravali A.N.** (1977) Design of Quantizers for DPCM Coding of Picture Signals. *IEEE Transactions on communications*, **25**, 1267-1274.

[57] **Tischer P.E., Worley R.T., Maeder A.J., Goodwin M.** (1993) Context-based Lossless Image Compression. *The Computer Journal*, **36**, 68-77.

[58] **Todd S., Langdon G.G., Rissanen J.** (1985) Parameter Reduction and Context Selection for Compression of Gray-Scale Images. *IBM Journal of Research and Development*, **29**, 188-193.

[59] **Tukey J.W.** (1971) *Exploratory Data Analysis*. Addison-Wesley.

[60] **Udpikar V., Raina J.** (1985) Modified Algorithm for the Block Truncation Coding of Monochrome Images. *Electronics Letters*, **21**, 900-902.

[61] **Udpikar V., Raina J.** (1987) BTC Image Coding Using Vector Quantization. *IEEE Transactions on communications*, **35**, 352-356.

[62] **Uhl T.J.** (1983) Adaptive Picture Data Compression Using Block Truncation Coding. *Signal Processing II: Theories and Applications, Proc. EUSIPCO-83*. Elsevier, Amsterdam, 147-150.

[63] **Wang Q.** (1992) *Analysis of Median Related and Morphological Filters with Application to Image Processing*. Tampere University of Technology, Ph.D. Thesis.

[64] **Wang Q., Gabbouj M., Neuvo Y.** (1993) Root Properties of Morphological Filters. *Signal Processing*, **34**, 131-148.

[65] **Wang Q., Neuvo Y.** (1992) BTC Image Coding Using Mathematical Morphology. *Proc. IEEE International Conference on Systems Engineering*, Kobe, Japan, 592-595.

[66] **Wang Q., Neuvo Y.** (1993) Deterministic Properties of Separable and Cross Median Filters with an Application to Block Truncation Coding. *Multidimensional Systems and Signal Processing*, **4**, 23-38.

[67] **Wang Y., Mitra S.K.** (1993) Image Representation Using Block Pattern Models and Its Image Processing Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **15**, 321-336.

[68] **Weitzman A., Mitchell H.B.** (1991) Fast Multi-Stage VQ Search for the BTC-VQ Algorithm. *Proceedings of 17th Convention of Electrical and Electronics Engineers in Israel*, Tel Aviv, Israel, 182-185.

[69] **Weitzman A., Mitchell H.B.** (1992) An Interblock BTC-VQ Image Coder. *Proceedings of 11th IAPR International Conference on Pattern Recognition*, III, Conference C, The Hague, Netherlands, 426-429.

[70] **Weitzman A., Mitchell H.B.** (1993) An Adaptive BTC-VQ Image Compression Algorithm Operating at a Fixed Bit-Rate. *Signal Processing, Image Commun.*, **5**, 287-294.

[71] **Wendt P.D., Coyle P.D., Gallagher N.C.** (1986) Stack Filters. *IEEE Transactions on Accoustics, Speech and Signal Processing*, **34**, 898-911.

[72] **Witten I., Neal R., Cleary J.** (1987) Arithmetic Coding for Data Compression. *Comm. ACM*, **30**, 520-539.

[73] **Wu Y., Coll D.C.** (1991) BTC-VQ-DCT Hybrid Coding of Digital Images. *IEEE Transactions on Communications*, **39**, 1283-1287.

[74] **Wu Y., Coll D.C.** (1992) Single Bit-Map Block Truncation Coding of Color Images. *IEEE Journal on Selected Areas in Communications*, **10**, 952-959.

[75] **Wu Y., Coll D.C.** (1993) Multilevel Block Truncation Coding Using Minimax Error Criterion for High Fidelity Compression of Digital Images. *IEEE Transactions on Communications*, **41**, 1179-1191.

[76] **Zeng B.** (1991) Two Interpolative BTC Image Coding Schemes. *Electronics Letters*, **27**, 1126-1128.

[77] **Zeng B., Neuvo Y.** (1993) Interpolative BTC Image Coding with Vector Quantization. *IEEE Transactions on Communications*, **41**, 1436-1438.

[78] **Zeng B., Gabbouj M., Neuvo Y.** (1991) A Unified Design Method for Rank Order, Stack, and Generalized Stack Filters Based on Classical Bayes Decision. *IEEE Transactions on Circuits and Systems*, **38**, 1003-1020.