5.2 BLOCK TRUNCATION CODING (BTC)

Edward J. Delp, Martha Saenz, and Paul Salama
Video and Image Processing Laboratory
School of Electrical and Computer Engineering
Purdue University
West Lafayette, Indiana

5.2.1. Introduction and Historical Overview

The problem of how one stores and transmits a digital image has been a topic of research for more than 40 years and was initially driven by military applications and NASA. The problem, simply stated, is how does one efficiently represent an image in binary form? This is the image compression problem. It is a special case of the source coding problem addressed by Shannon in his landmark paper [1] on communication systems. What is different about image compression is that techniques have been developed that exploit the unique nature of the image and the observer. These include the spatial nature of the data and of the human visual system. The "efficiency" of the representation depends on two properties of every image compression technique: data rate (in bits/pixel) and distortion in the decompressed image. The date rate is a measure of how much bandwidth one would require to transmit the image or how much space it would take to store the image.[1] Ideally one would like this to be as small as possible. If the decompressed image is exactly the same as the original image, the technique is said to be lossless. Otherwise the technique is lossy and the decompressed image has distortion or coding artifacts in it. Depending on the application, one can often trade distortion for data rate, hence if a user is willing to accept images with more distortion the data rate can often be lower.

Statistical and structural methods have been developed for image compression [2], the former being based on the principles of source coding with emphasis on the algebraic structure of the pixels in

[1] One can also use the "compression ratio" when describing data rate efficiency. The authors find this term to be imprecise and prefer to use data rate in bits/pixel.

an image, whereas the latter methods exploit the geometric structure of the image. In recent years there has been a great deal of activity in formulating standards for image and video compression. The results being the JPEG and MPEG standards discussed in chapters 5.5 and 6.4. Most statistical image compression methods are implemented by segmenting the image into non-overlapping blocks, since dividing the images into blocks allows the image compression algorithm adapt to local image statistics. The disadvantage, however, is that the borders of the blocks are often visible in the decoded image[2].

In this chapter we describe a lossy image compression technique known as *Block Truncation Coding (BTC)*. In the simplest possible terms: BTC is a block-adaptive binary encoder scheme based on moment preserving quantization. The basic concepts of BTC were born on March 17, 1977 in the office of O. Robert Mitchell at Purdue University during a conversation between Mitchell and his Ph.D. student Edward J. Delp. Delp and Mitchell discussed many ideas relative to how one could exploit statistical moments in the context of image compression. Delp began working on this concept as part of his Ph.D. thesis.[3] The first papers on BTC appeared at the IEEE International Conference on Communications in 1978 [3] and 1979 [4]. The first journal articles also appeared in 1979 [5, 6] along with Delp's thesis [7]. Since 1977 a great deal of work has been done on BTC. There has been more than 200 journal papers, 400 conference papers, 40 Ph.D. thesis and one book [8] published on BTC. BTC was a final candidate for the JPEG compression standard in 1987[4].

In the next section we will describe the basic BTC algorithm followed by a description of moment preserving quantization. We then describe various extensions to BTC and applications.

5.2.2. Basics of BTC

The basic BTC algorithm is a lossy fixed length compression method that uses a Q level quantizer to quantize a local region of the image. The quantizer levels are chosen such that a number of the moments of a local region in the image are preserved in the quantized output. In its simplest form, the objective of BTC is to preserve the sample mean and sample standard deviation of a grayscale image. Additional constraints can be added to preserve higher order moments. For this reason BTC is a block adaptive moment preserving quantizer.

---

[2] The reader might be familiar with this problem when selecting a low "quality factor" when using JPEG.

[3] The term "Block Truncation Coding" was coined by Delp in early 1978.

[4] See page 302 of [9]

The first step of the algorithm is to divide the image into non-overlapping rectangular regions. For the sake of simplicity we let the blocks be square regions of size n x n, where n is typically 4. For a two level (1 bit) quantizer, the idea is to select two luminance values to represent each pixel in the block. These values are chosen such that the sample mean and standard deviation of the reconstructed block are identical to those of the original block. An n x n bit map is then used to determine weather a pixel luminance value is above or below a certain threshold. In order to illustrate how BTC works, we will let the sample mean of the block be the threshold; a "1" would then indicate if an original pixel value is above this threshold, and "0" if it is below. Since BTC produces a bitmap to represent a block, it is classified as a binary pattern image coding method [10]. The thresholding process makes it possible to reproduce a sharp edge with high fidelity, taking advantage of the human visual system's capability to perform local spatial integration and mask errors. Figure 1 illustrates the BTC encoding process for a block. Observe how the comparison of the block pixel values with a selected threshold produces the bitmap.

By knowing the bit map for each block, the decompression/reconstruction algorithm knows whether a pixel is brighter or darker than the average. Thus, for each block two gray scale values, a and b, are needed to represent the two regions. These are obtained from the sample mean and sample standard deviation of the block, and are stored together with the bit map. Figure 2 illustrates the decompression process. An explanation of how a and b are determined will be given below.

For the example illustrated in Figures 1 and 2, the image was compressed from 8 bits per pixel to 2 bits per pixel (bpp). This is due to the fact that BTC requires 16 bits for the bit map, 8 bits for the sample mean and 8 bits for the sample standard deviation. Thus, the entire 4x4 block requires 32 bits, and hence the data rate is 2 bpp. From this example it is easy to understand how a smaller data rate can be achieved by selecting a bigger block size, or by allocating less bits for the sample mean or the sample standard deviation [5, 7]. We will discuss later how the data rate can be further reduced.

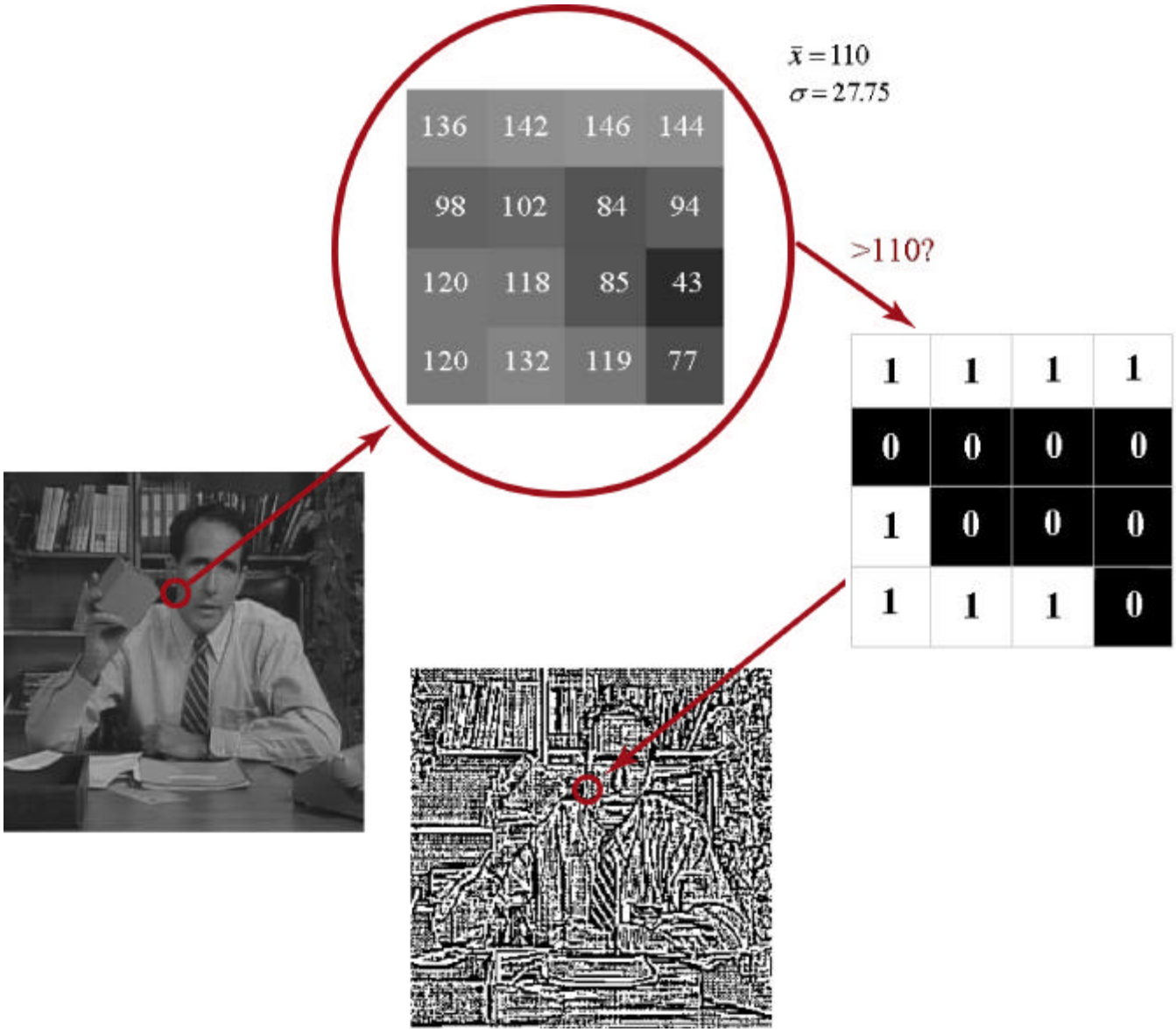

Figure 1. Illustration of the BTC compression process.

To understand how a and b are obtained, let k be the number of pixels of an nxn block ($k=n^2$) and $x_1, x_2, \ldots, x_k$ be the intensity values of the pixels in a block of the original image. The first two sample moments m_1 and m_2 are given by

$$m_1 = \bar{x} = \frac{1}{k}\sum_{i=1}^{k} x_i$$
$$m_2 = \frac{1}{k}\sum_{i=1}^{k} x_i^2 \quad , \qquad (1)$$

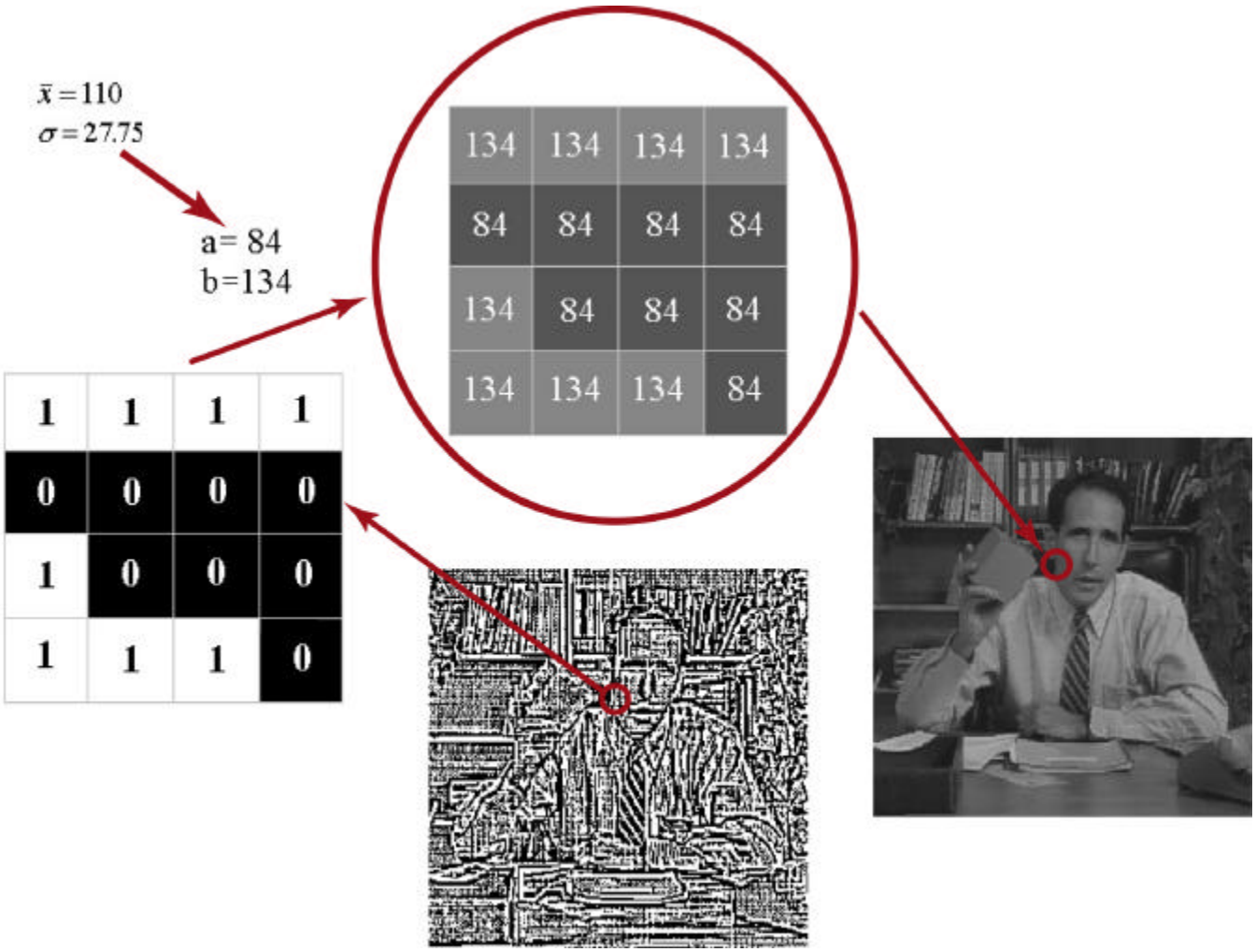

Figure 2. Illustration of the BTC decompression process.

and the sample standard deviation s is given by

$$s^2 = m_2 - m_1^2 \tag{2}$$

The one bit quantizer for a block and threshold, x_{th}, as shown in Figure 3, is defined by

$$Output = \begin{cases} b & if\ x_i \geq x_{th} \\ a & if\ x_i < x_{th} \end{cases} \quad for\ i = 1,2,\cdots,k\ . \tag{3}$$

As the example illustrated, the mean can be selected as the quantizer threshold. Other thresholds could also be used such as the sample median. Another way to determine the threshold is to perform an exhaustive search over all possible intensity values to find a threshold that minimizes a distortion measure relative to the reconstructed image [7].

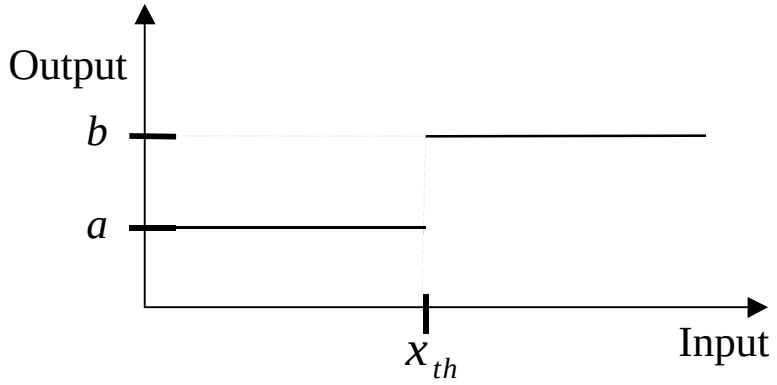

Figure 3. Binary Quantizer

Once a threshold, x_{th}, is selected the output levels of the quantizer (a and b) are found such that the first and second moments are preserved in the output. If we let q be the number of pixels in a block that are greater than or equal to x_{th} in value, we have:

$$\begin{aligned} k \cdot m_1 &= (k-q) \cdot a + q \cdot b \\ k \cdot m_2 &= (k-q) \cdot a^2 + q \cdot b^2 \end{aligned}. \tag{4}$$

Solving for a and b:

$$\begin{aligned} a &= m_1 - s\sqrt{\frac{q}{k-q}} \\ b &= m_1 + s\sqrt{\frac{k-q}{q}} \end{aligned}. \tag{5}$$

Rather than selecting the threshold to be the mean, an additional constraint can be added to (4) in order to determine the threshold of the quantizer. This is done by preserving the third sample moment (m_3):

$$k \cdot m_3 = (k-q) \cdot a^3 + q \cdot b^3, \tag{6}$$

where m_3 is given by

$$m_3 = \frac{1}{k}\sum_{i=1}^{k} x_i^{\,3} \,. \tag{7}$$

Since q is defined as the number of x_i's greater than or equal to x_{th}, the threshold is then implicitly determined by q:

$$q = \frac{k}{2}\left[1 + A\sqrt{\frac{1}{A^2+4}}\right], \text{ where } \quad A = \frac{3\cdot m_1 m_2 - m_3 - 2\cdot m_1^{\,3}}{s^{\,3}} \,(s \neq 0)\,. \tag{8}$$

It is evident how each block can be described by the sample mean (m_1), the sample standard deviation (s), and a bitmap where the ones and zeros indicate whether the pixel values are above or below the threshold. The data rate is then determined by the block size k and the number of bits f that are allocated to the sample mean and sample standard deviation of a block. The data rate is then given by $\frac{k+f}{k} = 1 + \frac{f}{k}$ bits as shown in Figure 4. For instance for k=16 and using 10 bits to jointly quantize m_1 and s, the image would be compressed to $1 + \frac{10}{16} = 1.625$ bits per pixel (bpp).

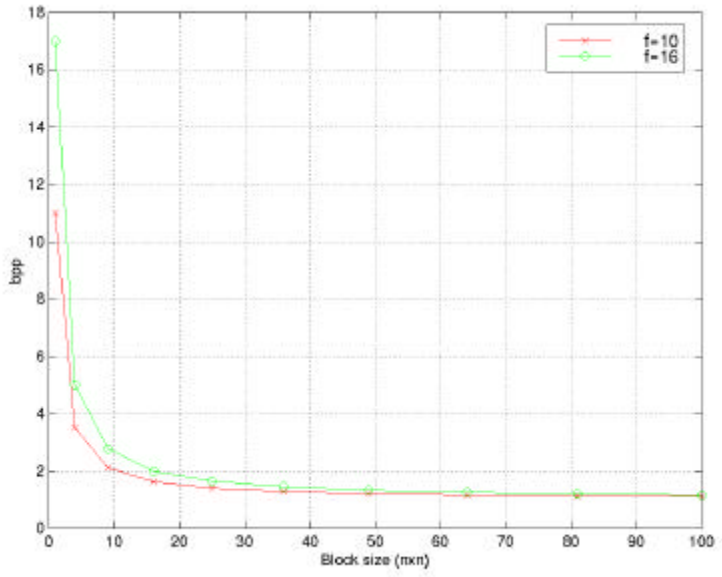

Figure 4. Data Rate vs. Block size

The issue of how many bits to assign to the sample mean and sample standard deviation was discussed in detail in [7,11]. The most important concept to note is that when the sample mean is small or large, the sample standard deviation must be small given the dynamic range of the pixel values. One

can exploit this and assign fewer bits to the sample standard deviation. In [11] it was shown that one could also use spatial masking models to reduce the number of bits assigned to the mean and standard deviation with 10 bits typically being enough to jointly quantize both values. The performance of BTC when the first three moments are preserved is illustrated in Figure 5. The image shown in Figure 5b is compressed to data rate of 1.625 bpp.

Another advantage to BTC is that channel errors do not propagate in the decompressed image due to the fact that BTC produces a fixed length binary representation of each block. Figure 5c shows the performance of BTC in the presence of channel errors when the channel has a bit error probability of 10^{-3}.

a. Original Image

b. Image Compressed to 1.625 bpp

c. Performance of BTC in the presence of channel errors.

Figure 5. BTC with errors

Other techniques can be used to design a one bit quantizer, for instance, one can use a fidelity criterion such as mean square error (MSE) or mean absolute error (MAE). If we let $y_1, y_2, \ldots, y_k$, be the x_i's sorted in ascending order, that is the order statistics of x_i's (see chapter 4.4), the MSE is then given by

$$MSE = \sum_{i=1}^{k-q-1} (y_i - a)^2 + \sum_{i=k-q}^{m} (y_i - b)^2 \ . \tag{6}$$

By minimizing the MSE, *a* and *b* are

$$\begin{aligned} a &= \frac{1}{k-q}\sum_{i=1}^{k-q-1} y_i \\ b &= \frac{1}{q}\sum_{i=k-q}^{m1} y_i \end{aligned} . \tag{7}$$

When minimizing the MAE (8), the values of *a* and *b* are given in (9).

$$MAE = \sum_{i=1}^{k-q-1} |y_i - a| + \sum_{i=k-q}^{m} |y_i - b|, \tag{8}$$

$$\begin{aligned} a &= median\ of\ (y_1, y_2, \ldots, y_{k-q-1}) \\ b &= median\ of\ (y_{k-q}, \ldots, y_{k1}) \end{aligned} . \tag{9}$$

A comparison between the use of MSE, MAE, and BTC is given in [5].

The main feature of BTC is the simplicity of its implementation, particularly due to the low decompression complexity. Due to the block nature of the algorithm the boundaries of adjacent blocks can sometimes be visible. The artifacts produced by BTC are usually seen around edges and in low contrast areas containing a sloping grayscale. In some images edges may appear to be ragged despite being sharp and some sloping gray levels may exhibit false contours [5].

5.2.3. Moment Preserving Quantization

In this section we will develop the Moment Preserving Quantizer (MP) quantizer. We will show that quantizers that preserve moments can be derived in closed form when the input probability density function is symmetric and the number of levels is relatively small. We will discuss how a MP quantizer can be formulated as the classical Gauss-Jacobi mechanical quadrature problem.

Since the advent of the use of pulse code modulation (PCM) systems there has been great interest in the design of quantizers. It was observed that non-uniform quantizers possessed properties that could be used to achieve results such as lower mean square error or enhanced subjective performance in the areas speech and image compression. These types of quantizers are designed for a particular input probability distribution function relative to a particular performance index or fidelity criterion. The most popular fidelity criterion used is that of the mean square error (MSE) between the input and output with the quantizer designed to minimize the mean square error. Other pointwise measures have also been proposed, such as the mean absolute error criterion. Studies have shown that pointwise fidelity criteria cannot be used reliably in image coding [12].

Preserving the moments of the input and output of a quantizer has been proven to be a very successful approach for image coding [5, 11]. Block Truncation Coding, as described in the previous section, uses a small number of levels and a non-parametric form of a moment preserving quantizer. By non-parametric we mean that the quantizer was designed to fit the actual data; no *a priori* probability distribution function is assumed. We will approach the problem by first examining a two level MP quantizer and then generalize the result to *Q* levels.

Let the random variable X denote the input to the quantizer, whose probability distribution function is $F(x)$, $x \in [c,d]$. The interval $[c,d]$ can be finite, infinite, or semi-infinite. Let Y denote the random variable at the output of the quantizer. For a two level quantizer, the random variable Y is discrete and takes on the values $\{y_1, y_2\}$ with probabilities $P_1 = \text{Prob}(Y = y_1)$ and $P_2 = \text{Prob}(Y = y_2)$. The output Y takes on the value y_1 whenever the input x is below some threshold x_{th}, otherwise the output is y_2. Therefore, in general, to design any two-level quantizer one must choose the two output levels y_1 and y_2 (designated by *a* and *b* in the previous section), and the input threshold x_{th} as illustrated in Figure 3. It is necessary that the quantizer preserve the first three moments of the input, otherwise one of the three parameters would have to be known (or guessed) initially [7]. To specify the quantizer one must solve the following equations for y_1, y_2, and x_{th}:

$$\begin{aligned} E[Y] &= E[X] = y_1 P_1 + y_2 P_2 \\ E[Y^2] &= E[X^2] = y_1^2 P_1 + y_2^2 P_2 \\ E[Y^3] &= E[X^3] = y_1^3 P_1 + y_2^3 P_2 \\ P_1 + P_2 &= 1, \end{aligned} \tag{10}$$

where the expectation operator is defined by

$$E[X^i] = \int_c^d x^i \, dF(x) \qquad \text{and} \quad y_1 \le x_{th} \le y_2 .$$

We shall assume throughout this presentation that the moments exist and are finite. (10) can be rewritten as

$$\begin{aligned} m_1 &= y_1 F(x_{th}) + y_2 [1 - F(x_{th})] \\ m_2 &= y_1^2 F(x_{th}) + y_2^2 [1 - F(x_{th})] \\ m_3 &= y_1^3 F(x_{th}) + y_2^3 [1 - F(x_{th})], \end{aligned} \tag{11}$$

where $m_i = E\left[X^i\right]$,

$$P_1 = \text{Prob}\left(X \le x_{th}\right) = F\left(x_{th}\right),$$

and

$$P_2 = \text{Prob}\left(X > x_{th}\right) = 1 - F\left(x_{th}\right).$$

By solving (11) for y_1, y_2, and x_{th}, the quantizer obtained is such that the first three moments of X and Y are identical. To find x_{th} we shall assume that $F^{-1}(\cdot)$ exists.

Without loss of generality we shall further assume that $m_1 = 0$ and $m_2 = 1$, i.e. X is zero mean and unit variance. (11) then becomes

$$\begin{aligned} 0 &= y_1 F\left(x_{th}\right) + y_2\left[1 - F\left(x_{th}\right)\right] \\ 1 &= y_1^2 F\left(x_{th}\right) + y_2^2\left[1 - F\left(x_{th}\right)\right] \\ m_3 &= y_1^3 F\left(x_{th}\right) + y_2^3\left[1 - F\left(x_{th}\right)\right]. \end{aligned} \tag{12}$$

By solving the first two equations for y_1 and y_2 in terms of $F\left(x_{th}\right)$ and using these solutions in the last equation we arrive at the desired results

$$\begin{aligned} y_1 &= -\sqrt{\frac{1 - F\left(x_{th}\right)}{F\left(x_{th}\right)}} = -\sqrt{\frac{P_2}{P_1}} \\ y_2 &= \sqrt{\frac{F\left(x_{th}\right)}{1 - F\left(x_{th}\right)}} = \sqrt{\frac{P_1}{P_2}} \\ F\left(x_{th}\right) &= \frac{1}{2} + \frac{m_3}{2}\sqrt{\frac{1}{4 + m_3^2}} \end{aligned} \tag{13}$$

This result is interesting in that the quantizer can be written in closed form. The above result in (13) also indicates that the threshold x_{th}, is nominally the *median* of X and not the mean as one would expect. The third moment m_3 is in general a signed number and can be thought of as a measure of skewness in the probability distribution function. This result indicates that the threshold is biased above or below the median according to the sign and magnitude of this skewness. These results are similar to those of BTC in the previous section, the difference being that BTC uses sample moments [5]. It should be noted that at this point we have no guarantee that $y_1 \le x_{th} \le y_2$. This problem will be addressed below.

The MP quantizer can be generalized to *Q*-levels. One needs to recognize that for a *Q*-level quantizer there are *Q* output levels and *Q*-1 thresholds. So if we desire an *Q*-level MP quantizer we need to know the first 2*Q*-1 moments, i.e., the *Q*-level MP quantizer preserves 2*Q*-1 moments. This, as shown in [13], guarantees the uniqueness of the quantizer. For large *Q* this does lead to the problem of knowing a large set of moments for a given distribution.

To arrive at the desired quantizer we need to know Q output levels $\{y_1, y_2, \cdots, y_Q\}$ and $Q-1$ thresholds $\{x_1, x_2, \cdots, x_{Q-1}\}$ with $y_1 \le x_1 \le y_2 \cdots y_{Q-1} < x_{Q-1} \le y_Q$. We again assume $m_1 = 0$ and $m_2 = 1$, and solve

$$m_n = \int_c^d x^n dF(x) = \sum_{i=1}^{Q} y_i^n P_i \quad \text{for } n = 0,1,2,\cdots,2Q-1, \tag{14}$$

where

$$x_0 = c,$$
$$x_Q = d,$$
$$m_n = E\left[X^n\right],$$
$$P_i = F(x_i) - F(x_{i-1}) = \text{Prob}(Y = y_i).$$

For a large class of practical problems where $F(x)$ admits a probability density function $f(x)$ and if $f(x)$ is even, i.e. $f(x) = f(-x)$, then the complexity of (14) is simplified since $m_n \equiv 0$ for n odd and the quantizer itself is symmetric. For a symmetric probability density function a closed form solution has been obtained for $Q = 2,3,4$ [13].

Equation (14) can be recognized as a form of the Gauss-Jacobi mechanical quadrature [14]. The output levels, y_i, of a *Q*-level MP quantizer are the zeros of the Q^{th} degree orthogonal polynomial associated with $F(x)$. The P_i are the Christoffel numbers and the x_i and y_i alternate by the Separation Theorem of Chebyshev-Markov-Stieltjes [14]. A review of orthogonal polynomials, the Gauss-Jacobi mechanical quadrature, and the Separation theorem are presented in [13].

Table 1 shows the MP quantizer thresholds and output levels for an input that has a zero mean, unit variance Gaussian probability density function. MP quantizer thresholds and output levels for uniform, and Laplacian probability distribution functions and other distributions are given in [7, 13].

Table 1: Positive thresholds and positive output levels for a MP quantizer (Q=2-16) for a zero mean, unit variance Gaussian probability density function. (MSE=mean square error)

	Output Levels	**Thresholds**		**Output Levels**	**Thresholds**
Q=2 Entropy 1.00 MSE 0.4042	1.0	0.0	Q=10 Entropy 2.0748 MSE 0.0820	0.4849 1.4650 2.4843 3.5818 4.8595	0.0000 1.0137 2.0568 3.1702 4.4491
Q=3 Entropy 1.2516 MSE 0.2689	0.0000 1.7312	0.9673	Q=11 Entropy 2.1419 MSE 0.0745	0.0000 0.9288 1.8760 2.8651 3.9361 5.1880	0.4805 1.4537 2.4620 3.5449 4.7951
Q=4 Entropy 1.4423 MSE 0.2032	0.7419 2.3344	0.0000 1.6866	Q=12 Entropy 2.2032 MSE 0.06841	0.4444 1.3404 2.2595 3.2237 4.2718 5.5009	0.0000 0.9216 1.8615 2.8409 3.8979 5.1232
Q=5 Entropy 1.5936 MSE 0.1626	0.0000 1.3557 2.8570	0.7277 2.2820	Q=13 Entropy 2.2598 MSE 0.0631	0.0000 0.8567 1.7254 2.6207 3.5634 4.5914 5.8002	0.4409 1.3309 2.2429 3.1978 4.2324 5.4358
Q=6 Entropy 1.7188 MSE 0.1362	6.6167 1.8892 3.3242	0.0000 1.3338 2.8003	Q=14 Entropy 2.3123 MSE 0.0587	0.4126 1.2427 2.0883 2.9630 3.8869 4.8969 6.0874	0.0000 0.8509 1.7142 2.6026 3.5363 4.5512 5.7349
Q=7 Entropy 1.8255 MSE 0.1166	0.0000 1.1544 2.3667 3.7504	0.6081 1.8624 3.2648	Q=15 Entropy 2.3611 MSE 0.0547	0.0000 0.7991 1.6067 2.4324 3.2891 4.1962 5.1901 6.3639	0.4096 1.2352 2.0755 2.4435 3.8586 4.8560 6.0221
Q=8 Entropy 1.9185 MSE 0.1024	0.5391 1.6365 2.8025 4.1445	0.0000 1.1408 2.3364 3.6890	Q=16 Entropy 2.4069 MSE 0.0519	0.3868 1.1638 1.9519 2.7602 3.6009 4.4929 5.4722 6.6308	0.0000 0.7943 1.5977 2.4182 3.2683 4.1670 5.1485 6.2986
Q=9 Entropy 2.0008 MSE 0.0909	0.0000 1.0233 2.0768 3.2054	0.5332 1.6193 2.7694 4.0818			

For comparison purposes the mean square error of the quantizer and the entropy of the output are shown.

The results for probability density functions on an infinite interval exhibit one of the disadvantages of the MP quantizer. The outputs at y_1 and y_Q have a tendency to spread much further from the origin than a minimum mean square error quantizer. What this says is that the quantizer assigns output levels that have a small probability of occurrence. These assignments of small probability output levels are reflected by the low values of the entropy for MP quantizers [13]. This indicates that it would be very hard to evaluate the MP quantizer for large values of Q (say larger than 30) because the output levels would be assigned such small probability of occurrence that one could have problems with computationally accuracy. Also it is no easy task to compute the zeros of a polynomial of high degree. These types of problems do not manifest themselves in the MSE quantizer due to the types of algorithms used to determine the output levels and input thresholds. Convergence properties of the MP quantizer for large Q are derived in [13]. It is also shown that the quantization error of the MP quantizer is negatively correlated with the input.

5.2.4. Variations and Applications of BTC

We will not attempt to list all the variations and extensions made to BTC over the years, rather we provide a general idea of the ways in which BTC has been used in image and video compression. Overviews of the many different variants of BTC are presented in [15, 16].

The first comparison study of the performance of BTC was done in 1980 [17]. In this study BTC was compared with the DCT and hybrid coding techniques in the context of high resolution aerial reconnaissance imagery. This study showed that at data rates from 1-3 bits/pixel (monochrome images) BTC performed very favorably compared to the other techniques.

After the initial work on BTC and moment preserving quantizers [13], the group at Purdue worked on several enhancements and extensions to the basic algorithm. These include coding graphics images [11], predictive coding [18], coding color images [19], the use of absolute moments [19], video [20, 21], and hardware implementations [22]. Figure 6 illustrates one of the recent applications of BTC in coding color images [23]. Here BTC is used in a multiresolution decomposition of the image to achieve a data rate of 1.89 bpp.

A great deal of work has been done on the use of absolute moments [24]. The use of absolute moments is interesting in that the mean square performance is better than the standard BTC approach. A very interesting recent paper by Ma [25] examines the earlier work done at Purdue by Lema and Mitchell [19] and argues that this work is often improperly cited. BTC has also been used with vector quantization, nonlinear filters and multilevel quantizers. Many video compression schemes have proposed using BTC including HDTV [26].

Due to its low complexity, BTC is attractive for hardware and/or software implementation. The first paper describing an integrated circuit approach was in 1978 [27], with more recent interest being in video [28]. Many software implementations have been proposed including Sun's CellB video format [29] that is used in their XIL library and as part of the multicast transport used on the Internet. The XMovie [30] architecture that has been suggested for multimedia systems is an extension of the DEC's Software Motion Pictures (SMP) [31] system based on BTC. Perhaps one of the most interesting recent extensions of BTC is in the area of binary pattern image coding [10] whereby the BTC bit plane is extended so that only certain patterns in each block are encoded. An excellent example of this approach is Visual Pattern Coding [32] which can preserve local gradient in each image block. These techniques have been shown to work quite well for video in multimedia application at data rates below 100 Kb/s.

Original Image

Encoded at 1.89 bpp

Figure 6. Illustration of the use of BTC in color image compression.

5.2.5 Conclusions

Block Truncation Coding has come a long way since March 1977. Despite the recent work in image video compression standards, BTC is still attractive in many applications that require low-complexity and moderate data rates. These include Internet video with software-only codecs, digital cameras and printers. On the research side there continues work on combining BTC with other techniques and approaches to improve performance. As in all research one never knows where this will lead. We have no doubt that BTC will be of interest to the research community and applications engineers well into the next century.

5.2.6 Acknowledgements

This work was partially supported by grants from the AT&T Foundation, the Rockwell Foundation and Lucent Technologies. Direct all correspondence relative to this chapter to E. J. Delp, at ace@ecn.purdue.edu, or http://www.ece.purdue.edu/~ace, or +1 765 494 1740.

5.2.7. References

1 C. L. Shannon, "A mathematical theory of communication," *Bell System Technical Journal,* vol. 27, July 1948, pp. 379-423, 623-656.

2 M. M. Reid, R. J. Millar, and N. D. Black, "Second-generation image coding: an overview," *ACM Computing Surveys*, vol. 29, no. 1, March 1997, pp. 3-29.

3 O. R. Mitchell , E. J. Delp, and S. G. Carlton, "Block truncation coding: a new approach to image compression," *Proceedings of the IEEE International Conference on Communications,* vol. 1, June 4-7 1978, pp. 12B.1.1-12B.1.4.

4 E. J. Delp and O. R. Mitchell , "Some aspects of moment preserving," *Proceedings of the IEEE International Conference on Communications,* vol. 1, June 10-14 1979, pp. 7.2.1-7.2.5.

5 E. J. Delp and O. R. Mitchell, "Image compression using block truncation coding," *IEEE Transactions on Communications*, vol. 27, no. 9, September 1979, pp. 1335-1341.

6 E. J. Delp, R. L. Kashyap, and O. R. Mitchell, "Image compression using autoregressive time series models," *Pattern Recognition*, Vol. 11, No. 5/6, 1979, pp. 313-323.

7 E. J. Delp, *Moment preserving quantization and its application in block truncation coding,* Ph.D. thesis, School of Electrical Engineering, Purdue University, August 1979.

8 B. V. Dasarathy, *Image Data Compression: Block Truncation Coding*, IEEE Computer Society Press, Los Alamitos, California, 1995.

9 W. B. Pennebaker and J.L. Mitchell, *JPEG Still Image Compression Standard*, Van Nostrand Reinhold, New York, 1993.

10 A.A. Rodriguez, C. E. Fogg, and E. J. Delp, "Video compression for multimedia applications," in *Image Technology: Advances in Image Processing, Multimedia, and Machine Vision*, Jorge L. C. Sanz (Ed), Springer, 1996.

11 O. R. Mitchell and E. J. Delp, "Multilevel graphics representation using block truncation coding," *Proceedings of the IEEE*, vol. 68, no. 7, July 1980, pp. 868-873.

12 D. J. Sakrison, "On the role of the observer and a distortion measure in image transmission," *IEEE Transactions on Communications*, vol. COM-25, no. 11, November 1977, pp. 1251-1267.

13 E. J. Delp and O. R. Mitchell, "Moment preserving quantization," *IEEE Transactions on Communications*, vol. 39, no. 11, November 1991, pp. 1549-1558.

14 G. Szego, *Orthogonal Polynomials*, American Mathematical Society, vol. 23, 1975.

15 H. B. Mitchell, N. Zilverberg, and M. Avraham, "A comparison of different block truncation coding algorithms for image compression," *Signal Processing: Image Communications*, vol.6, no. 1, March 1994, pp. 77-82.

16 P. Franti, O. Nevalainen, and T. Kaukoranta, "Compression of digital images by block truncation coding: a survey," *Computer Journal*, vol.37, no.4, 1994, pp.308-332.

17 O. R. Mitchell, S. C. Bass, E. J. Delp, T. W. Goeddel, and T. S. Huang, "Image Coding for Photo Analysis," *Proceedings of the Society for Information Display*, vol. 21, no. 3, 1980, pp.279-292.

18 E. J. Delp and O. R. Mitchell, "The use of block truncation coding in DPCM image coding," *IEEE Transactions on Signal Processing*, vol. 39, no. 4, April 1991, pp. 967-971.

19 M. D. Lema and O. R. Mitchell, "Absolute moment block truncation coding and applications to color images," *IEEE Transactions on Communications*, vol. 32, no. 10, October 1984, pp. 1148-1157.

20 D. J. Healy and O. R. Mitchell, "Digital video bandwidth compression using BTC," *IEEE Transactions in Communications*, vol. 29, no. 12, December 1981, pp. 1809-1817.

21 M. D. Lema and O. R. Mitchell, “Compression of video sequences using AMBTC with motion compensated prediction,” *Proceedings of the IEEE International Conference on Communications,* Chicago, June 1985, pp. 29-33.

22 T. N. Mudge, E. J. Delp, L. J. Siegel, H. J. Siegel, “Image coding using the multimicroprocessor system PASM,” *Proceeding of IEEE Computer Society Conference on Pattern Recognition and Image Processing,* pp. 200-205. New York, 1982

23 L.A. Overturf, M. L. Comer, E. J. Delp, “Color image-coding using morphological pyramid decomposition,” *IEEE Transactions on Image Processing*, vol. 4, no. 2, February 1995, pp. 177-185.

24 K. K. Ma and S. A. Rajala, “New properties of AMBTC (absolute moment block truncation coding),” *IEEE Signal Processing Letters*, vol.2, no.2, Feb. 1995, pp.34-36.

25 K. K. Ma, “Put absolute moment block truncation coding in perspective,” *IEEE Transactions on Communications*, vol. 45, no. 3, March 1997, pp. 284-286.

26 N. M. Nasrabadi, C. Y. Choo, T. Harries, and J. Smallcomb, “Hierarchical block truncation coding of digital HDTV images,” *IEEE Transactions on Consumer Electronics*, vol. 36, no. 3, August 1990, pp. 254-261.

27 W. L. Eversole, D. J. Mayer, F. B. Frazee, and T. F. Cheek, “Investigation of VLSI technologies for image processing,” *Proceedings of the DARPA Image Understanding Workshop,* Pittsburgh, November 1978, pp. 191-195.

28 L-G. Chen, Y-C. Liu, T-D Chiueh, and Y-P. Lee, “A real-time video signal-processing chip,” *IEEE Transactions on Consumer Electronics*, vol. 39, no. 2, pp. 82-92. May. 1993

29 W. K. Pratt, *Developing Visual Applications XIL: An Imaging Foundation Library,* Sun, 1998.

30 R. Keller, W. Effelsberg and B. Lamparter, “Xmovie: architecture and implementation of a distributed movie system,” *ACM Transactions on Information Systems,* vol. 13, no. 4, October 1995, pp 471-499.

31 B. K. Neidecker-Lutz and R. Ulichney, “Software motion pictures,” *Digital Technical Journal,* Vol. 5, No. 2, Spring 1983, pp. 1-9.

32 B. Barnett and A. C. Bovik, “Motion-compensated visual pattern image sequence coding for full motion multisession videoconferencing on multimedia workstations,” *Journal of Electronic Imaging*, vol.5, no.2, April 1996, pp.129-43.