Fingerprint Image Compression and the Wavelet Scalar Quantization Specification

Remigius Onyshczak
National Institute of Standards and Technology
Gaithersburg, MD 20899, U.S.A.

Abdou Youssef
Department of Computer Science
The George Washington University
Washington, DC 20052, U.S.A.

Abstract

Due to the large number and size of fingerprint images, data compression has to be applied to reduce the storage and communication bandwidth requirements of those images. In response to this need, the FBI developed a fingerprint compression specification, called Wavelet Scalar Quantization (WSQ). As the name suggests, the specification is based on wavelet compression. In this chapter, we will review the WSQ specification, and discuss its most important theoretical and practical underpinnings. In particular, we present the way wavelet compression generally works, and address the choice of the wavelet, the structure of the subbands, the different quantizations of the various subbands, and the entropy coding of the quantized data. The performance of WSQ will be addressed as well.

1 Introduction

In the beginning of the 1990s, the United States Department of Justice, Federal Bureau of Investigation, was confronted with a challenge. It had a rapidly growing collection of 35 million inked fingerprint cards in its repository, a considerable backlog of fingerprint cards needing to be classified, and a steadily increasing volume of fingerprint identification requests. The FBI needed a modern system to capture, classify, and identify fingerprints. It called upon other federal government agencies and private companies to help it create a system capable of gathering fingerprints at state and local law enforcement agencies, transmitting the digital fingerprint images to a central location where they could be classified in real time, matched to known fingerprints in a central database, and stored for future reference. The most current techniques in digital image capture and fingerprint pattern matching promised to make such a system possible, but one other critical piece in such a system had to be developed: image compression.

Digital fingerprint images of sufficient resolution for use in legal proceedings are quite large,

presenting problems in rapidly transmitting and economically storing the images. In addition, there are 500 million such images to be stored in and retrieved from a central database and possibly transmitted to state, local and other Federal government law enforcement agencies. To address these issues of transmission speed, bandwidth, and storage capacity, image compression is the logical approach [27].

A crucial question in such situations is the kind of compression to employ. Considering the high demand on reconstruction quality in fingerprint identification, lossless compression may seem the only option. However, lossless image compression techniques offer at most a 2-to-1 or 3-to-1 compression ratio, not sufficient to ease the transmission bandwidth requirements or reduce the amount of media needed to store the volume of fingerprint images at the FBI's central processing facility. A lossy image compression technique had to be developed which could achieve greater than 10-to-1 compression ratios while preserving the essential characteristics of the fingerprint image for both mechanized pattern matching and fingerprint identification by human examiners. The compression technique had to be non-proprietary, allowing it to be used by vendors supplying fingerprint image gathering, storage, retrieval, and processing hardware to local, state, and federal law enforcement agencies.

The FBI, together with the help and cooperation of the other agencies and private firms, developed a lossy compression technique for fingerprint images which it described in a document entitled "WSQ Gray-scale Fingerprint Image Compression Specification" [3, 4, 5, 12]. This chapter presents the theory behind, a description of, and practical implementation considerations of WSQ.

It should be noted that there are several techniques and standards for compressing general images, including fingerprints. Among the best-known and most widely used image compression standards are JPEG [20] and JPEG 2000 [24], and examples of well-known compression techniques include transform coding [7], DCT-based compression [21], subband coding [33], and fractal compression [2]. Also, there have been studies and development of compression techniques for fingerprint images specifically, such as model-based compression [15, 11] and JPEG adapted to fingerprint images [29]. However, none of those standards and techniques has been officially adopted by law enforcement for fingerprint compression. Instead, WSQ is now the primary specification for fingerprint compression, used by the FBI and other law enforcement agencies and departments. Therefore, we focus primarily on WSQ, and compare its performance with JPEG.

This chapter is organized as follows. The next section discusses the general framework of wavelet compression, which is the theory underlying WSQ. Section 3 addresses the various choices to be made by the designer for a complete wavelet compression system, and identifies briefly the choices made by WSQ. Section 4 elaborates more on the WSQ specification, and section 5 discusses the WSQ implementation experience of the first author. The last section concludes the chapter.

2 General Framework of Wavelet Compression

Wavelet-based compression has received considerable attention in the 1990s [1, 31], and has been adopted by various important standards such as JPEG 2000 [24] and MPEG4 [8]. The reasons for the high interest in wavelet compression are largely due to the competitive compression ratios that can be achieved at high quality without the bothersome blocking artifacts of the

JPEG-style discrete-cosine-transform (DCT) compression [7, 21].

Like any transform-based compression algorithm, wavelet compression works in three stages: transform, quantization, and entropy coding. In quantization and entropy coding, the same choices are available to wavelet compression as to other transform-based compression techniques. For quantization, one can choose between scalar [13] and vector quantizers [14], and from each category one can select from several types that will be discussed later. Similarly, for entropy coding, a variety of lossless bit-packing techniques can be used, such as run-length encoding (RLE) [23], Huffman coding [16], RLE+Huffman, or arithmetic coding [26], to name a few of the most commonly used ones. The choices made by WSQ will be discussed later in the chapter.

In the transform stage, however, wavelet compression differs significantly from other transform-based compression techniques. Specifically, in standard transform-based compression such as DCT, the transform is a matrix-vector multiplication, where the vector represents the signal to be transformed and compressed, while the matrix is fixed for any given signal size. In wavelet compression, on the other hand, the transform is a pair of filters, where one is low-pass and the other is high-pass, and the output of each filter is downsampled by two. Each of those two output signals can be further transformed similarly, and this process can be repeated recursively several times, resulting in a tree-like structure, called the decomposition tree. Unlike in DCT compression where the transform matrix is fixed, in wavelet transform the designer has the choice of what filter-pair to use and what decomposition tree structure to follow.

To aid the reader in understanding wavelet compression, the basic concepts of filters and downsampling are reviewed in the next subsection. (For additional details, the reader can refer to any digital signal processing text such as [18].) The rest of the section defines wavelets and wavelet transforms, and outlines the wavelet based compression scheme.

2.1 Linear Filters

A (linear) filter is a process characterized by a (finite or infinite) sequence (f_k), called the filter coefficients. The filter takes as input any sequence (signal) (x_k) and outputs another sequence (y_k) which is *the convolution* of the (x_k) and (f_k). That is, for all index values k, $y_k = \sum_r f_{k-r} x_r$ where r ranges over all values for which f_{k-r} is nonzero.

To understand the effect of a filter, it is better to view it from the frequency domain perspective using the z-transform and the Fourier transform. The *z-transform* of any sequence (a_k) is the complex function $A(z) = \sum_k a_k z^k$, where k ranges over all values for which a_k is nonzero, and z is a complex variable. The *Fourier transform* of (a_k) is the function $A(e^{-i\theta})$, where θ is a real variable. Note that $A(e^{-i\theta})$ is periodic of period 2π. The value $A(e^{i\theta})$ is referred to as *the frequency content of the signal (a_k) at frequency θ*. In the range from $-\pi$ to π, the values of $A(e^{i\theta})$ at θ near 0 are typically referred to as *the low-frequeny contents* (or simply the low frequencies) of the signal, and those at θ away from 0 and near $\pm\pi$ are referred to as *the high-frequeny contents* (or simply the high frequencies) of the signal. Note also that if the signal sequence (a_k) is finite (say that $k = 0, 1, ..., N-1$), then the *discrete Fourier transform* of (a_k) is a new sequence (A_k) where $A_k = A(e^{-i\frac{2k\pi}{N}})$, for $k = 0, 1, ..., N-1$.

The relationship between the input (x_k) and the output (y_k) of a filter (f_k) can be easily expressed using the z-transform: $Y(z) = F(z)X(z)$, that is, the z-transform of (y_k) is the product of the z-transforms of (f_k) and (x_k). In particular, $Y(e^{-i\theta}) = F(e^{-i\theta})X(e^{-i\theta})$, that is,

the Fourier transform of the output is the product the Fourier transforms of the input and the filter.

Therefore, the frequencies in the output signal (y_k) of a filter are the frequencies of the input signal (x_k) but scaled (i.e., multiplied) by the frequencies of the filter coefficients (f_k). In particular, if the low frequencies of the filter are equal to 1 and the high frequencies are all equal to 0, then the filter effect is the preservation of the low frequencies of the input (x_k) and the zeroing out of the high frequencies of the input. In such a case, the filter is called a *low-pass filter*. Conversely, if the low frequencies of the filter are 0 and the high frequencies are all 1, then effect of the filter is to zero out the low frequencies of the input and preserve the high frequencies. In this case, the filter is called a *high-pass filter*.

In intutive terms, the low frequencies of a signal generally represent the overall outline of the shape of the signal, while the high frequencies represent the fine details of the signal shape. Thus, a low-pass filter preserves the broad features of a signal (such as an image) but eliminates much of the details, whereas a high-pass filter eliminates much of the texture and and preserves the outlines (edges) of an image.

In addition to filtering, the wavelet transform and inverse transform involve downsampling by 2 and upsampling by 2. The *downsampling by 2 of a signal* (u'_k) is simply the dropping of the odd-index terms of the sequence, resulting in sequence (u'_{2k}) consisting of the even-indexed terms only. Thus, the output of filtering (x_k) into (u'_k) and then dowsampling by 2 is $(u_k) = (u'_{2k})$ where $u_k = u'_{2k} = \sum_i f_{2k-i}x_i$. The *upsampling by 2 of a signal* (u_k) is achieved by interspersing a zero between every two consecutive sample values of the signal. That is, the upsampling by 2 of (u_k) results in a signal (w_k) where $w_{2k} = u_k$ and $w_{2k+1} = 0$ for all k. In particualr, if (u_k) was the result of downsampling by 2 of (u'_k), then the upsampling by 2 of (u_k) is (w_k) where $w_{2k} = u'_{2k}$ and $w_{2k+1} = 0$ for all k.

2.2 Definition of Wavelets

A function $\psi(t)$ is called a wavelet if there exists a corresponding function $\phi(t)$, called a scaling function, and four sequences of real (or complex) numbers (g_k), (h_k), (p_k) and (q_k), all satisfying the following equations:

$$\phi(t) = \sum_k p_k \phi(2t-k), \ and \ \psi(t) = \sum_k q_k \phi(2t-k)$$

$$\phi(2t-n) = \frac{1}{2}\sum_k [g_{2k-n}\phi(t-k) + h_{2k-n}\psi(t-k)]$$

where for most wavelets of interest to compression the four sequences are real and finite. Furthermore, in order for ϕ and ψ to form a legitimate wavelet system, certain constraints have to be satisfied by the four sequences. Some of these constraints are

$$\sum_k g_k = \sum_k p_k = 2, \ and \ \sum_k h_k = \sum_k q_k = 0.$$

Other more stringent constraints are known as the perfect reconstruction (PR) conditions in the subband coding community [30, 31, 33]. The PR conditions can be stated succinctly by means of the z-transform:

$$G(z)P(z) + H(z)Q(z) = 4 \ and \ G(-z)P(z) + H(-z)Q(z) = 0$$

where $G(z),\ H(z),\ P(z)$ and $Q(z)$ are the z-transforms of $(g_k),\ (h_k),\ (p_k)$ and (q_k).

2.3 Wavelet Transforms

Given a wavelet defined through the fours sequences (g_k), (h_k), (p_k) and (q_k), the wavelet transform of an input data sequence (x_k) of length N transforms the sequence (x_k) into another sequence (y_k) of same length N. The first $\lceil \frac{N}{2} \rceil$ terms of (y_k) form a sequence (u_k) called a low-frequency subband, and the remaining $\lfloor \frac{N}{2} \rfloor$ terms of (y_k) form a second sequence (v_k) called a high-frequency subband, computed as follows:

$$u_k = \sum_n \frac{g_{2k-n}}{\sqrt{2}} x_n \; and \; v_k = \sum_n \frac{h_{2k-n}}{\sqrt{2}} x_n.$$

That is, (u_k) is derived by first passing the (x_k) through a linear filter of coefficients $(\frac{g_k}{\sqrt{2}})$, yielding a sequence (u'_k) where $u'_k = \sum_n \frac{g_{k-n}}{\sqrt{2}} x_n$, and then downsampling (u'_k) by two, i.e., $u_k = u'_{2k}$. The subband (v_k) is derived similarly using the filter $(\frac{h_k}{\sqrt{2}})$. Note that the first filter is low-pass because $\sum_k g_k$ is nonzero, while the second filter is high-pass because $\sum_k h_k = 0$.

Because of the equations of the PR conditions, the original sequence (x_k) can be perfectly reconstructed from the two subbands (u_k) and (v_k) by the following equation:

$$x_k = \sum_n \left[\frac{p_{k-2n}}{\sqrt{2}} u_n + \frac{q_{k-2n}}{\sqrt{2}} v_n \right],$$

which is equivalent to the following three steps: (1) upsampling (u_k) by 2 and then filtering through a linear filter of coefficients $(\frac{p_k}{\sqrt{2}})$, (2) upsampling (v_k) by 2 and then filtering through a linear filter of coefficients $(\frac{q_k}{\sqrt{2}})$, and finally (3) adding the two resulting sequences. This whole reconstruction process implements the inverse wavelet transform. Figure 1 shows graphically both the transform (or analysis) stage and the reconstruction (or synthesis) stage. The dotted lines, labeled "process", in the figure, refers in our context to the other stages of compression, that is, quantization and entropy coding, which are discussed later.

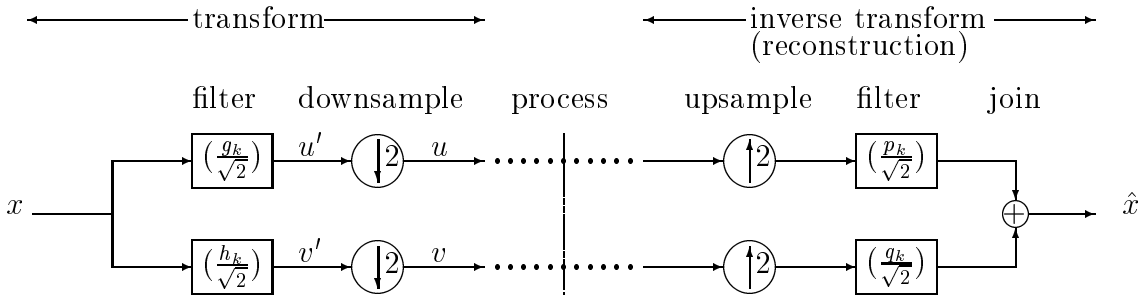


Figure 1: **The Subband Coding Scheme Corresponding to Wavelets**

Note that the low-pass subband (u_k) contains basically the same features as the original signal, albeit with slightly fewer details. Those missing details are indeed what is stored in the high-frequency subband (v_k). Since the subband (u_k) still has most of the original features, its data elements must still carry considerable correlation. This warrants further transforms on (u_k). Indeed, in actual wavelet-based compression systems, the wavelet transform is applied several times on the subbands, resulting in many subbands which are then quantized and entropy-coded. Figure 2 shows a typical wavelet coding/decoding structure.

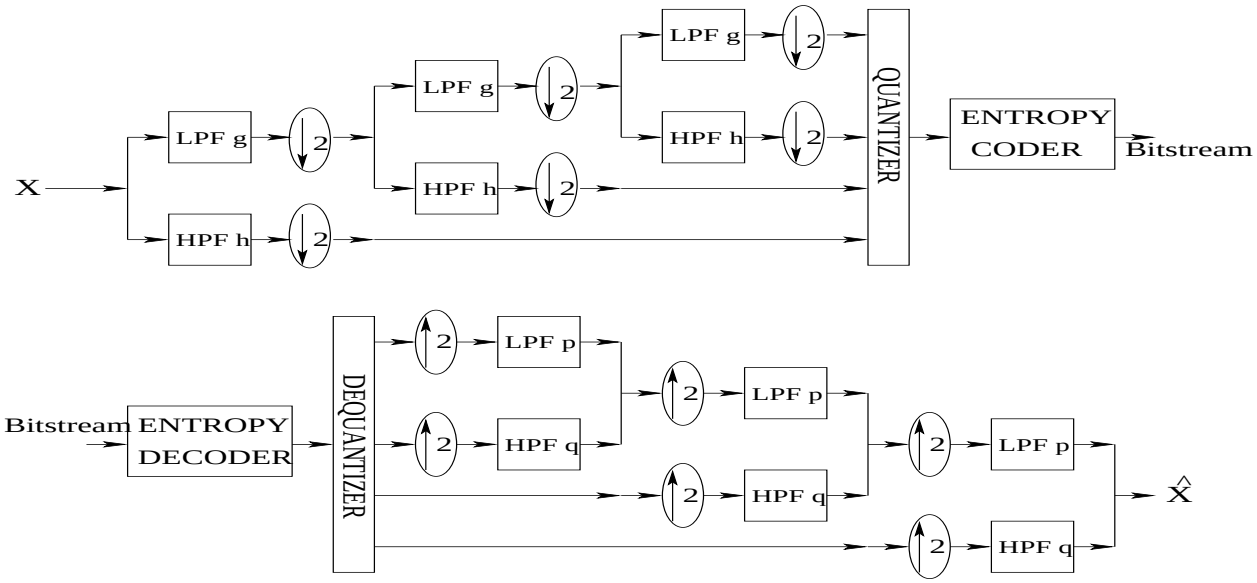


Figure 2: **An Example structure of a Typical Wavelet Coder (above) and its Corresponding Decoder (below)**

In the case of multidimensional data (e.g., images), the transforms are typically performed on the various dimensions independently. The reconstruction (i.e., decompression) follows the exact reverse process: entropy decoding, dequantization, and inverse wavelet transforms applied as many times as, and in the opposite order of, the direct transform.

3 Choices and Tradeoffs in Wavelet Compression

Clearly, many different decomposition tree structures can be followed. Also, many different choices of wavelets (i.e., wavelet filters) are available to choose from. Similarly, for the next two stages of the compression scheme, namely, quantization and entropy coding, the designer can choose from a variety of techniques. These choices and some of the tradeoffs will be discussed in this section, and the choices made by WSQ will be identified.

3.1 Choice of Good Wavelets

As mentioned earlier, the designer has the choice of which wavelet to use, that is, which 4-filter set to employ. One of the earliest questions addressed in wavelet compression was which wavelets yield good compression performance [9, 25, 32, 34]. This subsection addresses this question briefly.

First of all, biorthogonal wavelets are better than non-biorthogonal ones for compression because the corresponding filters have the very desirable property of being linear-phase filters, that is, symmetric or antisymmetric. In other terms, biorthogonal wavelets do not exhibit frequency aliasing artifacts, while other wavelets do.

Still, there is an unlimited number of biorthogonal wavelets, and many families have been

defined and shown to yield good compression performance. One of the extensive studies of best-wavelet selection was conducted by one of the authors on a very large set of wavelets [34]. This large set and the wavelet evaluation process and findings are outlined next.

To start, it is worthwhile to mathematically characterize biorthogonal wavelets. The perfect reconstruction equations, the standard choice of $P(z) = zH(-z)$ and $Q(z) = -zG(-z)$, and the linear-phase (symmetry) property of the filters together imply that for $z = e^{-i\omega}$,

$$P(z)G(z) = cos^{2N}(\frac{\omega}{2}) \left[\sum_{k=0}^{N-1} \begin{pmatrix} N-k+1 \\ k \end{pmatrix} sin^{2k}(\frac{\omega}{2}) \quad + \quad sin^{2N}(\frac{\omega}{2}) T(cos(\omega)) \right]$$

where N is an arbitrary positive integer and T is an arbitrary odd polynomial ($T(-x) = -T(x)$) [6, 10]. Wavelet designers have many choices for $T(x)$ and N, and, even for a fixed T and N, the designer can find many choices for P and G that satisfy the equations just stated.

In [34], $T(x)$ was taken to be simply 0, in which case the filters (g_k) and (h_k) have a total of $4N$ coefficients. All biorthogonal filters where the total number of coefficients of (g_k) and (h_k) range from 4 to 56, corresponding to $N = 1, 2, ..., 14$, were generated and evaluated. The total number of those wavelets is 4297. The reason the total filter length was capped at 56 was because longer filters incur high computation cost. The evaluation was based on the frequency response of the filters as well as on direct experimentation, that is, each 4-filter set was tested by compressing several test images and evaluating the reconstruction quality at various compression ratios.

The findings of this extensive study revealed that out of the 4297 wavelets tested, only about 18 of them were good for compression. The rest had inferior reconstruction quality, and in most cases the quality was very unacceptable. Interestingly, the wavelet specified in WSQ turned up among the 18 good wavelet. Furthermore, although the WSQ wavelet was not the absolute best, its filter lengths were the shortest (thus most efficient) and its performance was very close to the best wavelet.

3.2 Choices of Decomposition Trees

As just mentioned, many different tree structures can be followed. For example, the tree depth can vary depending on the size of the signal, the amount of correlation of the data, and the desired speed of the system, among others. Also, the high-frequency subband can be transformed further, although its tree need not be so elaborate because high-frequency subbands do not have much correlation left in them to warrant many additional decompositions.

In images, as in 1D signals, various tree structures can be used. Figure 3 shows two tree structures for wavelet image compression. The first structure, due to Mallat [19], is well known, and the second is the structure used in the FBI's WSQ Specification.

The structure of the decomposition tree can be determined dynamically at compression time. The general approach is to measure the level of "energy" in each subband generated after each application of the transform. The term "energy" is often used in this context as a metaphor for the amount of details and variations present in the subband: the more variations and the finer the details, the more energy. This can be measured in various ways, such as the spectrum and the statistical variance of the subband. If the energy level of a subband is found to be above a certain threshold, that subband is transformed further (i,.e, decomposed

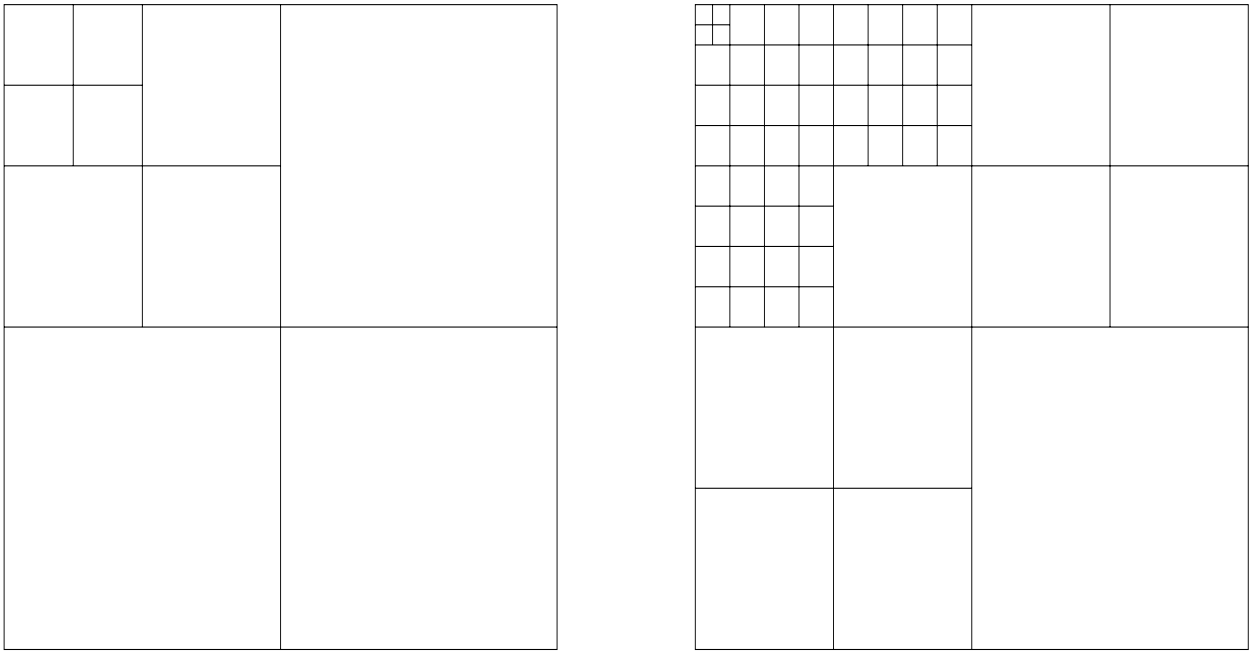


Figure 3: **Examples of Tree Decompositions of Images**

into smaller subbands), and then the same energy test is applied on the smaller subbands. While this dynamic approach can yield high compression ratios, its advantage is offset by the overhead of (1) extra execution time to measure the energy level, and (2) the extra bits needed to represent the shape of the resulting decomposition tree.

The logical alternative is to perform tests on several benchmark images in the intended application domain, such as fingerprints. From those tests, a reasonably optimal decomposition (on average) can be deduced and used thereafter on all images in the application. This has the advantage of avoiding the overhead of dynamic decisions, while still producing nearly optimal results. This route was the one taken in WSQ.

3.3 Quantization in Wavelet Compression

Roughly speaking, quantization is a process of further discretization of the data, and can be done at the individual data element (e.g., pixel) level, or at a vector (or block) level. In the first case, it is called scalar quantization [13], and in the second case vector quantization [14].

A scalar quantizer is specified by (1) subdividing the entire range of the data to be quantized into intervals labeled with integer values, and (2) designating a reconstruction value for each interval. If interval i is denoted $(d_i, d_{i+1}]$, and its reconstruction value denoted r_i, where $d_i < r_i \leq d_{i+1}$, then quantizing a value x is nothing more than replacing x by the integer label i of the interval in which x falls ($d_i < x \leq d_{i+1}$). Dequantizing i amounts to replacing it by the reconstruction value r_i.

Different classes of scalar quantizers have been defined, most notably uniform quantizers and optimal Max-Lloyd quantizers [17]. In uniform quantizers, all the intervals have the same length, and the reconstruction values are the exact middle points of their respective intervals. Clearly, uniform quantizers are efficient to specify and efficient to quantize with. In optimal Max-Lloyd quantizers [17], the end-points of the intervals and the reconstruction values are determined so that the error between the original data and the reconstructed (i.e., quantized then dequantized) data is minimized, where the error measure is the Euclidean distance (or mean-square error). Max-Lloyd quantizers optimize reconstruction quality, and are fairly efficient to derive and apply. Nevertheless, they are not always used because they do incur additional overhead over uniform quantizers.

Scalar quantization works very well when the data elements are rather independent (i.e., uncorrelated). If, however, the data elements are correlated, scalar quantization causes considerable distortions because it quantizes the various elements independently, that is, irrespective of inter-element relations. When data is correlated, vector quantization is preferable because it maintains better the correlations and thus the patterns and features represented thereof.

Vector quantization is specified by a dictionary of patterns (vectors), which are relatively small 2D $n \times m$ blocks in the case of images. Quantizing an image involves subdiving it into contiguous nonoverlapping $n \times m$ blocks, and coding each block by the index of the best-matching block in the dictionary. Dequantizing is simply replacing each index in the quantized image by the corresponding dictionary block. A typical block size is 4×4 or 8×8, and the number of blocks in a dictionary range from a few hundred to a few thousand. Various clustering techniques are used to construct good dictionaries [14].

In the context of wavelet compression, the coefficients of low-frequency subband may still be quite correlated and thus a vector quantizer may yield better performance than a scalar quantizer. However, in practice, the decomposition tree is so deep that the low-frequency subband size is so small that little gain can be derived from vector quantization, in which case scalar quantization of that subband is adequate. As for all the other (high-frequency) subbands, if the wavelet chosen is a very good wavelet, the coefficients in all those subbands are highly decorrelated, making the use of vector quantization unnecessary and even undesirable because of the high overhead as compared to scalar quantization. Therefore, scalar quantization of the high-frequency subbands is the quantizer of choice.

Of course, not all the subbands should necessarily be quantized with the same scalar quantizer. Rather, each subband is better processed with a quantizer whose specifications are tailored to that subband's statistics. (Note that the more intervals are used, and thus the smaller the intervals, the less the error, but of course the less the compression ratio.)

The FBI WSQ Specification opted for subband-adaptive scalar quantization. The quantizer of each subband is nearly a uniform quantizer: all the intervals are of the same length except for the middle interval centered at 0, which is 40% larger than the other intervals. The reconstruction values are all near the middle points of their intervals. Those quantizers are subband-adaptive in that the length of most of the intervals are inversely proportional to the logarithm of the variance of the subband. The length of the intervals depends also on a scalar multiplicative value, which is set depending on the desired compression ratio. The more compression is desired, the smaller that scalar factor is.

3.4 Entropy Coding

Entropy coding is the process of representing information efficiently but losslessly. Many entropy coding techniques have been developed over the last several decades, such as run-length encoding (RLE) [23], Huffman coding [16], arithmetic coding [26], and Lempil-Ziv [35], among others. Entropy coders are used either as standalone lossless compression systems or as the last stage of lossy compression. In the latter case, which is the one of interest to us here, the entropy coder is used to represent the quantized data. Among the many entropy coders, the most commonly used ones in lossy compression are arithmetic coding and RLE+Huffman.

Arithmetic coding encodes binary decisions as well as data bits using a probabilistic model of the information to be coded. It achieves optimal compression performance (in terms of bitrate or compression ratio). As a result, it has been adopted in many standards such as JPEG 2000, MPEG-4, and JBIG. It is also an option in JPEG. However, there are numerous patents on arithmetic coding. This complicates matters for many users, and hinders their ability to utilize arithmetic coding.

The situation with RLE and Huffman coding is much simpler as users are free to use them at will. Furthermore, they are easy to implement, fast to run, and yield very good compression performance. Therefore, they have been adopted in many standards such as JPEG and MPEG-2, as well as in the FBI WSQ specification.

RLE works well when the data to be coded consists of relatively long sequences of repeated values. Such sequences are called runs. RLE represents each run by a pair (n, v) where n is the length of the run, and v is the repeated value. In applications where only one type of runs is represented (e.g., runs of zeros only), the runs are coded simply by their lengths since the repeated value is implicit.

Huffman coding represents a data set of symbols (be they numeric, alphabetic or otherwise) by coding each symbol with a binary string whose length depends on the probability(or frequency) of occurrence of that symbol in the data set. The individual symbol codes are constructed in such a way that (1) the more frequently occurring a symbol is, the shorter its binary code is, and (2) no symbol binary code is a prefix of another symbol code. The alphabet symbol codes are derived as follows.

Suppose the symbols that make up the data set are drawn from an alphabet of m distinct symbols $\{a_1, a_1, ..., a_m\}$, and let p_i be the probability of occurrence of symbol a_i, for $i = 1, 2, ..., m$. The Huffman coding algorithm for coding the alphabet is a greedy algorithm that builds a binary tree. First, a node is created for each alphabet symbol; afterwards, the algorithm repeatedly selects two unparented nodes of smallest probabilities, creates a new parent node for them, and makes the probability of the new node to be the sum of the probabilities of its two children. Once the root is created, each left-pointing edge of the tree is labeled with 0, and each right-pointing edge is labeled with 1. Finally, each symbol is coded with the binary sequence that labels the path from the root to the leaf node of that symbol.

In WSQ, each quantized subband is turned into a one-dimensional sequence, then RLE is applied to code runs of zeros, and finally the run-lengths and other remaining data are coded with Huffman.

3.5 Other Approaches to Wavelet Compression

In recent years, wavelet techniques for lossless compression and for lossy-to-lossless transmission/compression have been developed [28, 22]. Prior to this development, the filter coefficients of the wavelets used in compression were non-integer numbers that make their use lossy because of the limited precision of computer hardware. Later on, integer wavelets were developed, especially what is called the lifting scheme [22], allowing for lossless compression.

Furthermore, because of the multiresolution nature of wavelet representations, they can be used for progressive transmission, also called lossy-to-lossless. The idea is to format the bitstream in such a way that any number leading bits in it allows for a reconstruction of the whole image at a resolution level that depends on the number of those leading bits. This allows the user to control the tradeoff between image quality and amount of bits transmitted. It also allows for error tolerance and data loss in image transmission. For example, if some bits of the transmitted bitstream are lost or corrupted by noise, the receiver-decoder can use the bits that preceded the error/loss to reconstruct the whole image at whatever resolution affordable by those bits.

JPEG 2000 has adopted this scheme of wavelet compression for still images. WSQ, however, does not have provisions for lossless or lossy-to-lossless coding. The initial intent of WSQ, namely, to save not only on transmission bandwidth but also on storage, makes both of these modes undesirable because the resulting compression ratio would still be too modest as compared to the WSQ target compression ratios of at least 10 to 1 and above.

4 Fingerprint Image Compression: - The WSQ Specification

4.1 Intended Use of WSQ Compression

The WSQ compression technique developed by the FBI and other entities was designed to compress source fingerprint images between ratios of 10 to 1 and 20 to 1. At these compression ratios, sufficient friction ridge and pore detail is retained for the purposes of identification, by fingerprint matching hardware and by human latent fingerprint examiners. The technique is designed to discard information which is not necessary for the reconstruction of a fingerprint image usable by a latent fingerprint examiner to make a positive identification and by devices which classify the fingerprint pattern and extract minutia by mechanized means. Minutia, that is, the friction ridge endings and bifurcations, are the features by which fingerprint patterns are distinguished from one another.

4.2 The Source Fingerprint Image

This lossy compression technique produces best results when the source fingerprint image is a result of scanning an inked or chemical process fingerprint image from a card, or the output image produced by a livescan fingerprint capture device with a spatial resolution from 500 to 520 samples per inch in both the vertical and horizontal directions and an intensity resolution of 8 bits. The source image is also required to be continuous tone (i.e. having a reasonably

diverse intensity histogram with significant contribution from pixel values other than white (integer value 255) and black (integer value 0). The reasons for these requirements are:

- Using less than the required spatial resolution results in a source image with features too small to produce a significant result in the wavelet analysis, resulting in loss of these features in the quantization step of the WSQ encoding process.
- A source image with less than 8 bits per pixel in intensity resolution will also suffer from high loss of information in the quantization step of the WSQ encoding process.
- Source images with most of their pixel intensities consisting of white and black will exhibit excessive "ringing" or "echos" in the reconstructed image, again, resulting from information discarded in the quantization step of WSQ encoding.

Using a source fingerprint image with the correct characteristics will produce a reconstructed fingerprint image remarkably similar to the source fingerprint image.

Before wavelet analysis, the 8-bit image pixel values must be shifted and scaled, in that order. The statistical mean of the pixel data is obtained, then subtracted from all of the pixel values in the source image. Finally the shifted image pixel data is divided by a value equal to the absolute furthest shifted pixel value from 0. This ensures no input data to the wavelet analysis process extends beyond the -1.0 to 1.0 range.

4.3 Wavelet Transform and Quantization in WSQ

The WSQ wavelet is a biorthogonal wavelet. The low-pass analysis filter has nine coefficients, and the high-pass analysis filter has 7 coefficients. These short filters make the transform rather fast. The coefficients of these filters are:

Low-Pass Analysis Filter Coefficients	High-Pass Analysis Filter Coefficients
0.037828455506995	
-0.02384946501938	0.064538882628938
-0.11062440441842	-0.040689417609558
0.37740285561265	-0.41809227322221
0.85269867900940	0.78848561640566
0.37740285561265	-0.41809227322221
-0.11062440441842	-0.040689417609558
-0.02384946501938	0.064538882628938
0.037828455506995	

The wavelet transform is as described in Section 2.3. The decomposition tree with labeled subbands is shown in Figure 4.

The quantization is uniform scalar quantization of the subbands except that the interval centered at 0 is 40% larger than the rest of the intervals, and the reconstruction value of each interval $(d_i, d_{i+1}]$ is $d_i + 0.56(d_{i+1} - d_i)$ rather than $d_i + 0.5(d_{i+1} - d_i)$. The precise length of the intervals varies from subband to subband, and depends on the variance of the subband.

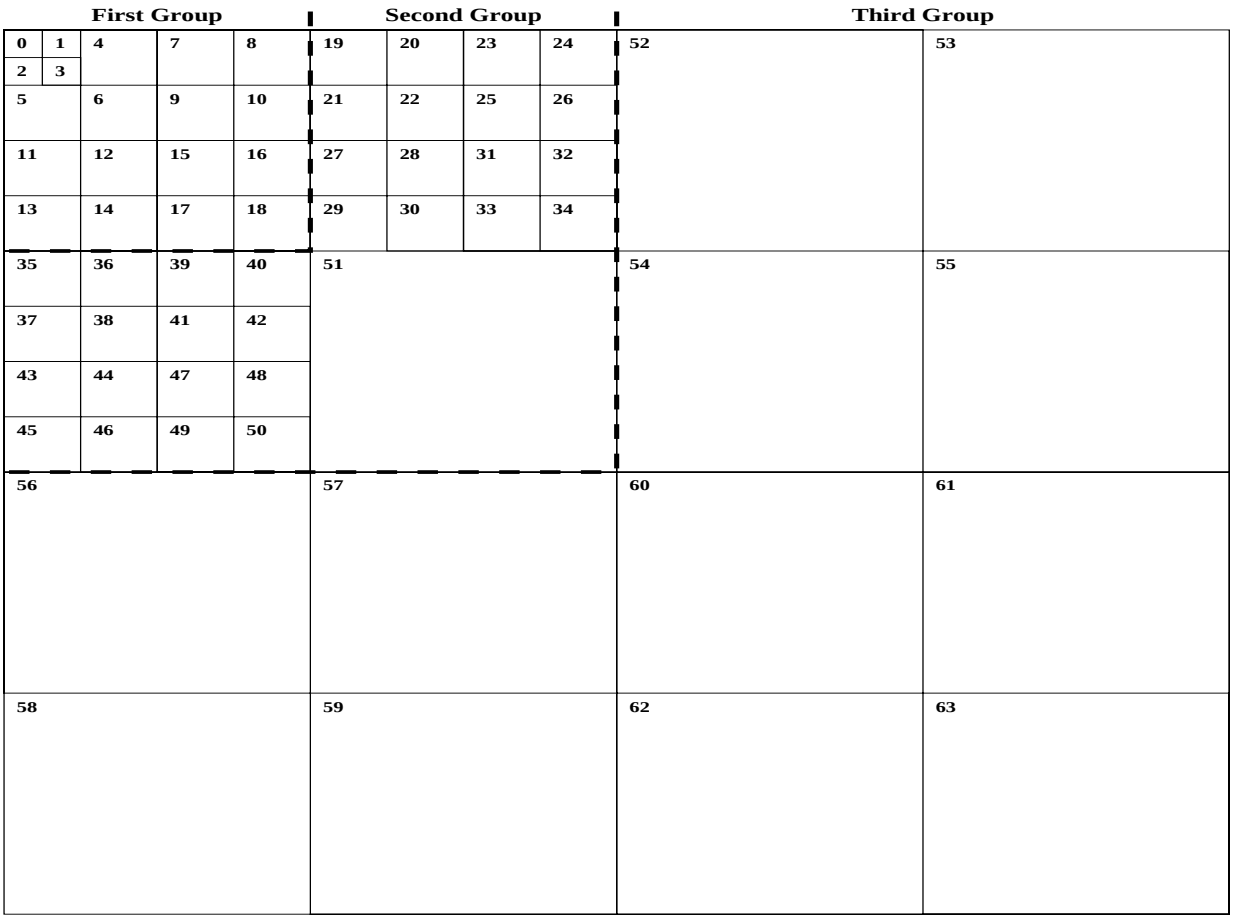


Figure 4: **The WSQ Decomposition Tree with Labeled Subbands**

4.4 Run-Length Encoding and Decoding in WSQ

This encoding scheme uses integers, either 8-bit or 16-bit, to represent long sequences of zeros (called runs) in a subband which result from scalar quantization. Only sequences of zeros are coded with RLE. The lengths of these zero-runs are encoded with Huffman encoding as described in the next subsection. During the run length decoding process, the integers which indicate a zero run are decoded from the Huffman encoded bitstream and zeros are deposited in the subband in preparation for wavelet synthesis. More information about this is found in the section about Huffman Coding.

4.5 Huffman Coding and Decoding

Huffman encoding, used as the last step in WSQ lossy compression, is based on assigning the shortest binary codes to the most frequently occurring symbols in the Huffman input table, which is derived by a process described in subsection 3.4. Table 1 shows the codes of each of the 254 valid input symbols for the Huffman coder. These input symbols are fixed in the WSQ Specification. A Huffman table transmitted with the entropy coded bitstream consists of

Table 1: **Huffman Coder Input Symbols**

Symbol	Description
1	Zero run of length 1
2	Zero run of length 2
⋮	
99	Zero run of length 99
100	Zero run of length 100
101	escape for positive 8-bit coefficient index
102	escape for negative 8-bit coefficient index
103	escape for positive 16-bit coefficient index
104	escape for negative 16-bit coefficient index
105	escape for 8-bit zero run
106	escape for 16-bit zero run
107	coefficient index value -73
108	coefficient index value -72
⋮	
180	(same as symbol 1)
⋮	
253	coefficient index value 73
254	coefficient index value 74

the number of each code length (1 through 16) and the Huffman input symbol order. This is sufficient to generate the decoding table at the decoder. A maximum of eight Huffman tables may be included with each WSQ compressed file.

Huffman coding is simply the concatenation of variable length codes, each representing a symbol, to create a bitstream. The boundaries of codes can, and most often will, occur within bytes in a WSQ file. No Huffman code can begin with any other Huffman code. This makes decoding analogous to traversing a binary tree, concluding at a leaf containing the symbol the code represents. The code associated with each symbol is generated using the population distribution of the input symbols.

Huffman coding starts at the upper-left corner of subband 1, proceeds left to right then top to bottom exhausting all bin index values in subband 1. Subbands 2 through 64 are processed in turn. At the end of Huffman coding, the rest of the byte is filled with '1' bits to signal the end of the bitstream to the decoder. As Huffman coding proceeds in the encoder, any byte consisting of all '1' bits is immediately succeeded by a byte consisting of all '0' bits. This eliminates any confusion between these Huffman generated all '1' bytes and valid marker codes. (See the subsection on markers to further explain the makeup and use of markers.)

Table 2: **WSQ Marker Codes**

Second Byte	Description
1010 0000	Start of Image
1010 0001	End of Image
1010 0010	Start Frame
1010 0011	Start Subband Group
1010 0100	Start Wavelet Filter Tap Table
1010 0101	Start Quantization Tables
1010 0110	Start Huffman Tables
1010 0111	Restart Interval
1010 1000	Comment

4.6 Progressive Transmission

To facilitate the possibility of transmitting a half-resolution or quarter-resolution image using the WSQ Specification, frequency subbands have been separated into 3 groups. Decoding the first group to completion yields an image one-quarter the resolution of the original source image. A half-resolution image can be obtained by decoding the first two subband groups. Decoding all subband groups to completion reconstructs the full-sized image. This feature is thought to find application when a human is looking through many images to find candidate matches.

The fact that a smaller portion of the total data is transmitted and processed to produce fractional resolution images quickens the search process and eliminates unnecessary processing and transmission bandwidth usage. This segmentation also allows the calculation of 3 independent Huffman coding tables to further optimize the compression performance of WSQ.

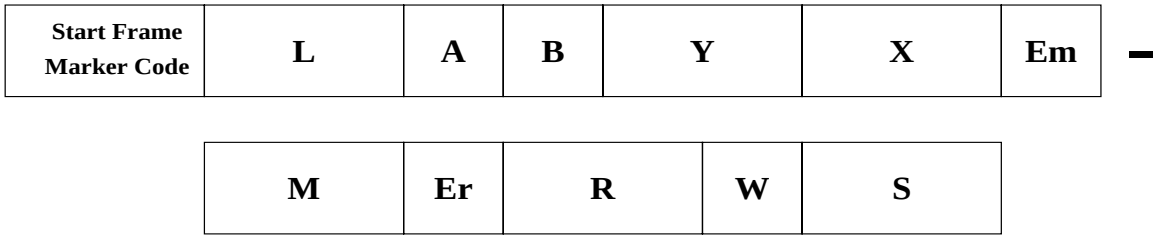
4.7 Marker Codes

Marker codes are two-byte sequences, the first of which is a byte with all bits set to 1, ending with a byte indicating the type of marker. Some marker codes allow the WSQ encoded file to be quickly searched for items of interest such as the Huffman tables, wavelet filter coefficients (i.e., tap values), and image dimensions. Others allow recovery from bit or byte errors encountered as a result of bad storage media, or interference during transmission.

A description of the marker secondary byte contents and their meaning appears in Table 2. Note that the second byte will never be all '0' bits because this sequence is reserved for the purpose of indicating an all '1's byte generated by the Huffman coder. Some of the more useful marker codes are Start Frame, Start Subband Group, Start Wavelet Filter Tap Table, Start Quantization Tables, and Start Huffman Tables.

It is necessary to include information in the WSQ compressed file to enable the decoder to reconstruct an image with the same dimensions, brightness and contrast as the original source image. The Start Frame marker indicates the point after which this data is found. Figure 5 shows the Start Frame header information content and sequence.

Each subband group is marked by a Start Subband Group marker code to enable progressive



L - indicates the length of the rest of the information
A - scanner BLACK calibration value
B - Scanner WHITE calibration value
Y - number of lines in the source image
X - number of columns in the source image
Em - the decimal point in M is shifted left Em places
M - source image pixel value mean subtracted before scaling
Er - the decimal point in R is shifted left Er places
R - the source image was divided by R before transformation
W - specifies the WSQ encoder algorithm used on this image
S - specifies the software implemantation that encoded this image

Figure 5: **Start of Frame Information**

transmission and reconstruction, and allows for multiple Huffman code tables to be used, one for each subband group. Figure 6 details the content of the Start Subband header.

The Wavelet Filter Tap information must be provided with the encoded bitstream to provide some indication of the synthesis high-pass and low-pass filter tap values needed to reconstruct the image. A Start Wavelet Filter Tap Table marker signals the start of this group of information. Figure 7 shows the content of this header.

The quantization bin centers and sizes used in the WSQ encoder must be supplied to the decoder to enable the recovered integer bin index values to be converted to approximate wavelet coefficients for processing by the wavelet synthesis stage. Figure 8 shows how these bin centers and sizes are transmitted using the Start Quantization Table marker code.

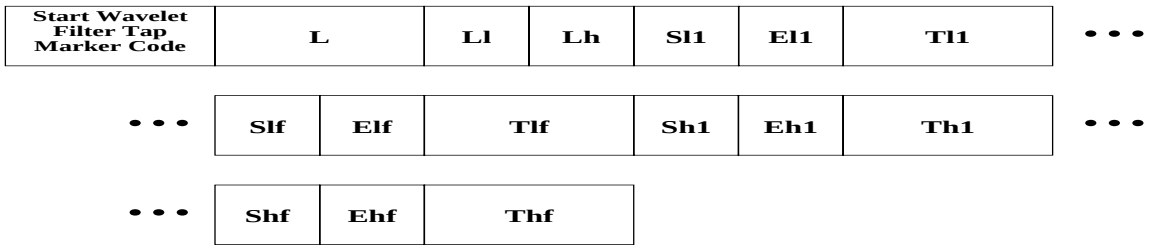
Figure 9 shows the content and arrangement of information in the Start Huffman Table header. This information enables huffman decoder tables to be constructed at the WSQ Decoder and is crucial to extracting bin index integers from the entropy coded bitstream.

5 Implementation of WSQ

The practical portion of this chapter is based on experience with the reference WSQ encoder/decoder pair which was designed and tested by the first author at the National Institute of Standards and Technology (NIST) in Gaithersburg, Maryland, USA. The encoder/decoder pair was implemented in the C programming language.

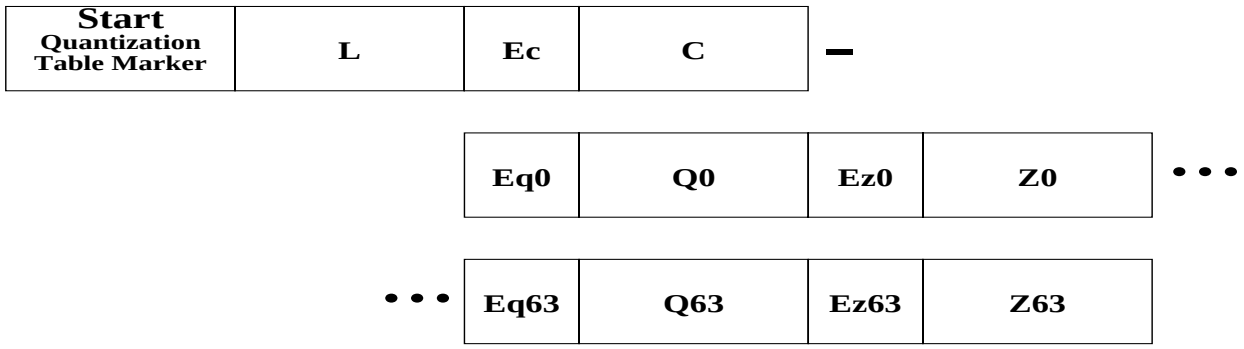
L - length of Start Subband Group marker header
H - huffman code table number used in this subband group

Figure 6: **Subband Group Information**



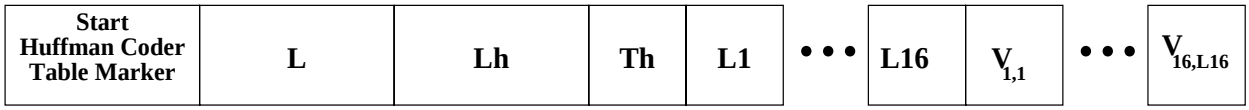
L - indicates the length of the rest of the information
Ll - number of taps in wavelet lowpass analysis filter
Lh - number of taps in wavelet highpass analysis filter
Sl1 - sign of first wavelet lowpass analysis tap (0 = positive)
El1 - move decimal point left El1 places in Tl1
Tl1 - value of first wavelet lowpass analysis filter tap
.
.
.
Slf - sign of last wavelet lowpass analysis tap (0 = positive)
Elf - move decimal point left Elf places in Tlf
Tlf - value of last wavelet lowpass analysis filter tap
Sh1 - sign of first wavelet highpass analysis tap (0 = positive)
Eh1 - move decimal point left Eh1 places in Th1
Th1 - value of first wavelet highpass analysis filter tap
.
.
.
Shf - sign of last wavelet highpass analysis tap (0 = positive)
Ehf - move decimal point left Ehf places in Thf
Thf - value of last wavelet highpass analysis filter tap

Figure 7: **Wavelet Filter Tap Information**



L - Total length of information in this header
Ec - decimal point shifts left Ec0 digits in C
C - quantizer bin center parameter
Eq0 - decimal point shifts left Eq0 digits in Q0
Q0 - quantization bin size for subband 0
Ez0 - decimal point shifts left Ez0 digits in Z0
Z0 - quantization zero bin size for subband 0
.
.
.
Eq63 - decimal point shifts left Eq63 digits in Q63
Q63 - quantizer bin size for subband 63
Ez63 - decimal point shifts left Ez63 digits in Z63
Z63 - quantizer bin size for subband 63

Figure 8: Quantization Table Information



L − Total length of information in this header
Th − Huffman Table Identifier
L1 − Number of 1−bit huffman codes
.
.
.
L16 − number of 16−bit huffman codes
$V_{1,1}$ − symbol associated with this huffman code
.
.
.
$V_{16.L16}$ − symbol associated with last huffman code

Figure 9: Start Huffman Table Header

5.1 Processing Speed

The coding process consists of the transform stage, the determination of the quantizer parameter value, the quantization stage, and the entropy coding stage. To estimate the time it takes to code an $N \times M$ fingerprint image, the transform stage let
t_* = the time for a single floating-point multiplication operation
t_+ = the time for a single floating-point addition/subtration operation
t_d = the time for a single floating-point division operation
t_c = the time for a single floating-point comparison operation
t_{log} = the time for computing the natural logarithm.

The transform stage takes time equal to

$$T_{transform} = (34t_* + 30t_+)NM$$

and the quantization stage takes time equal to

$$T_{quantization} = (10.5t_+ + 5.25t_* + 2.62t_c)NM + 284t_d + 184t_* - 56t_+ + 56t_{log}.$$

The entropy coding stage involves scanning the quantized image and finding runs of zeros, and then Huffman-coding the run lengths and the nonzero coefficients. The scanning time is proportional to NM, but the number of runs encountered varies from image to image and, within the same image, varies depending on the target bitrate. Due to this variability, it is hard to come up with a quantified time formula for the entropy coding stage. Rather, empirical time measurements will be provided below.

For the determination of the value of the quantizer parameter that would meet the target bitrate, the implementors of WSQ are free to follow any method they choose. The correctness of the decoder depends on the value of the parameter, not on the method of derivation of that value. Therefore, it is hard to give a time formula for this part of WSQ as well, but empirical performance measures are provided instead.

The decoding process consists of the entropy decoding stage, the dequantization stage, and the inverser trasnform stage. The entropy decoding stage takes time proportional to NM. The dequatization stage takes time equal to

$$T_{dequantization} = (t_* + 3t_+)NM,$$

and the inverse transform stage take time equal to

$$T_{inverse\ transform} = (17t_* + 13.77t_+)NM.$$

In terms of real time, WSQ is quite fast. We ran WSQ on various images on a Pentium II 733 MHz, and measured the speed. Table 3 gives representative figures of encoding and decoding speeds for two fingerprint images of typical sizes, at compression ratio of 15:1.

Table 3: **WSQ Speed on a Pentium 733 MHz at Compression Ratio of 15:1**

Image Dimensions	Encoding Speed	Decoding Speed
768×768	0.635 sec	0.994 sec
800×856	0.741 sec	1.061 sec

5.2 Compression Performance of WSQ: Compression Ratio vs. Quality

The compression ratio range of the WSQ specification is 10:1 to 20:1. In this range, experience with many images has shown that the visual quality of the reconstructed images is very good: adequate for fingerprint identification. Figures 10 illustrates the WSQ performance at compression of 15:1 (15 to 1), and Figures 11 and 12 show the performance at mush higher compression ratios (60:1 and 120:1). As the figures show, the quality is very good at 15:1; it also holds up well at much higher compression ratios, although echoing and smearing effects begin to appear and adversely limit fingerprint identification.

Examining the performance limits of WSQ at high and low ends of compression ratios is, indeed, of interest. At very high compression ratios in the mid-hundreds, the limit was reached when the size of the additional information included in the WSQ file (e.g. huffman tables, etc) dwarfed the encoded bitstream information and compression ratio could not be achieved. Near this compression rate upper limit the reconstructed image showed ringing or echoing at edges and blurring within the fingerprint image, as already apparent in Figure 12. The blurring within the fingerprint obscured minutia and made even classification impossible. The echoing at edges created the opportunity for false minutia detection by automated methods.

At the low end of compression ratios the limit was noted when, during scalar quantization, large wavelet coefficients in the lower subbands needed to be represented by integers larger than the 16-bit maximum the WSQ specification allows. Near this low compression ratio limit, the image, as expected, showed very little difference from the source image. These differences were only detectable using a pixel-by-pixel comparison of the two images. Because this low compression ratio limit is lower than compression ratios achieved by other lossless compression techniques (e.g., the lossless mode of JPEG), the WSQ technique is unsuitable for lossless compression.

To appreciate the performance of WSQ relative to other lossy compression systems, we compare the performance of WSQ with that of JPEG, which is one of the most widely used lossy image compression standards. Figures 13, 14 and 15 show the same fingerprint image reconstructed by both compression systems at compression ratios of 15:1, 30:1 and 45:1, respectively. To the untrained eye, JPEG seems to give the same performance at 15:1; however, the FBI latent examiners found that even at this relatively modest compression ratio, JPEG images hindered fingerprint identification. At higher compression ratios (e.g., 45:1), the blocking artifacts and loss of details in JPEG become more evident, as can be seen in Figure 15. To help the reader see the performance difference, Figures 17-19 show the error (difference) images between the original and the reconstructed for both compression systems at the three compression ratios of 15:1, 30:1 and 45:1. Note how the error image of JPEG is much more pronounced at compression ratios of 30:1 and 45:1 than those of WSQ.

5.3 Compliance Testing Using the WSQ Coder/Decoder

Vendors wishing to produce WSQ implementations must show compliance with the FBI's WSQ specification by participating in a Compliance Testing Program set up by Michael McCabe, NIST. The vendor obtains test files from NIST, processes them using their own implementation of the WSQ specification, and submits the files they produce to NIST for evaluation. The results are sent to the FBI for review and the FBI issues the Certificate of Compliance to the

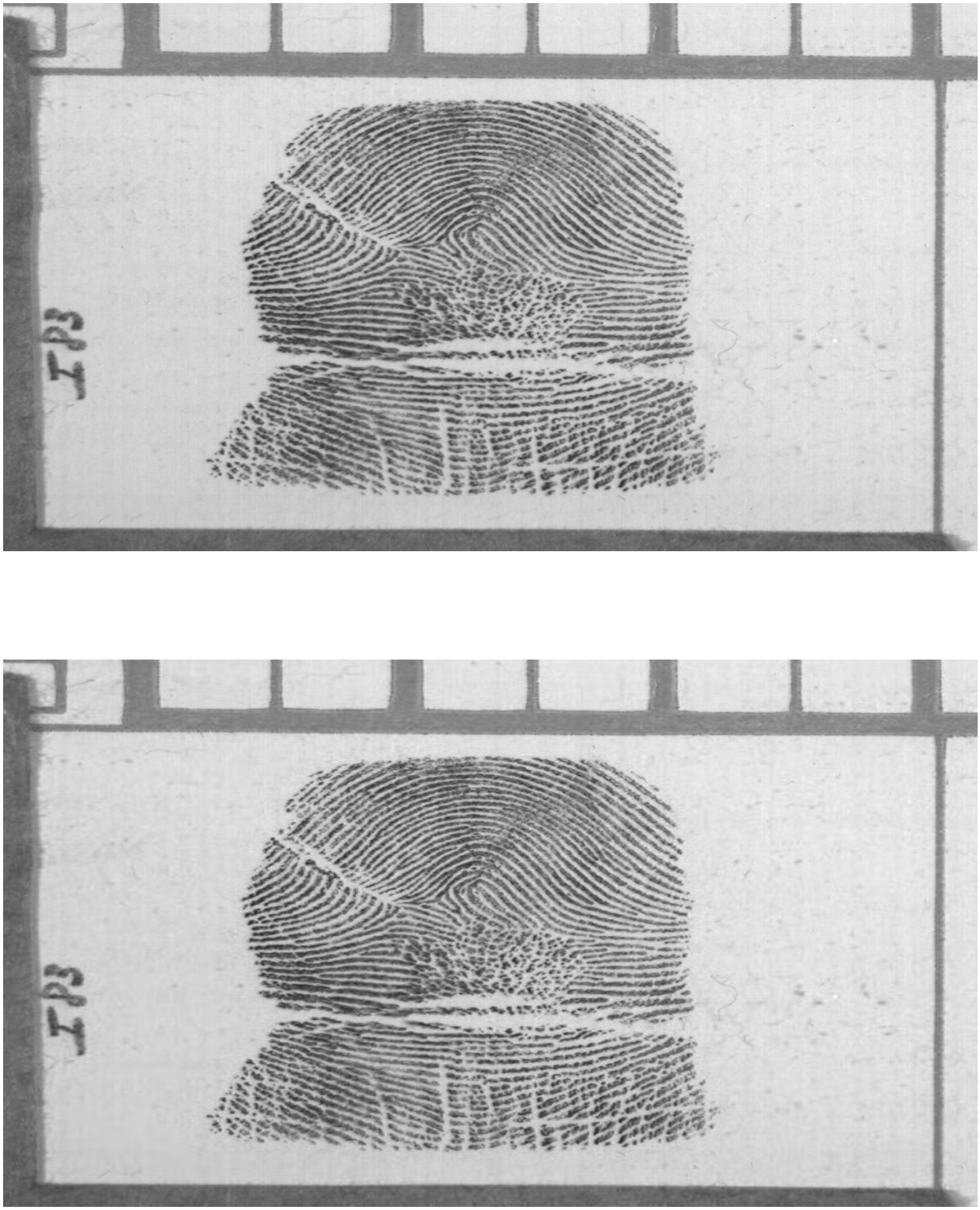


Figure 10: **An Original Fingerprint Image (above) and its WSQ-Recosntructed at 15:1 (below)**

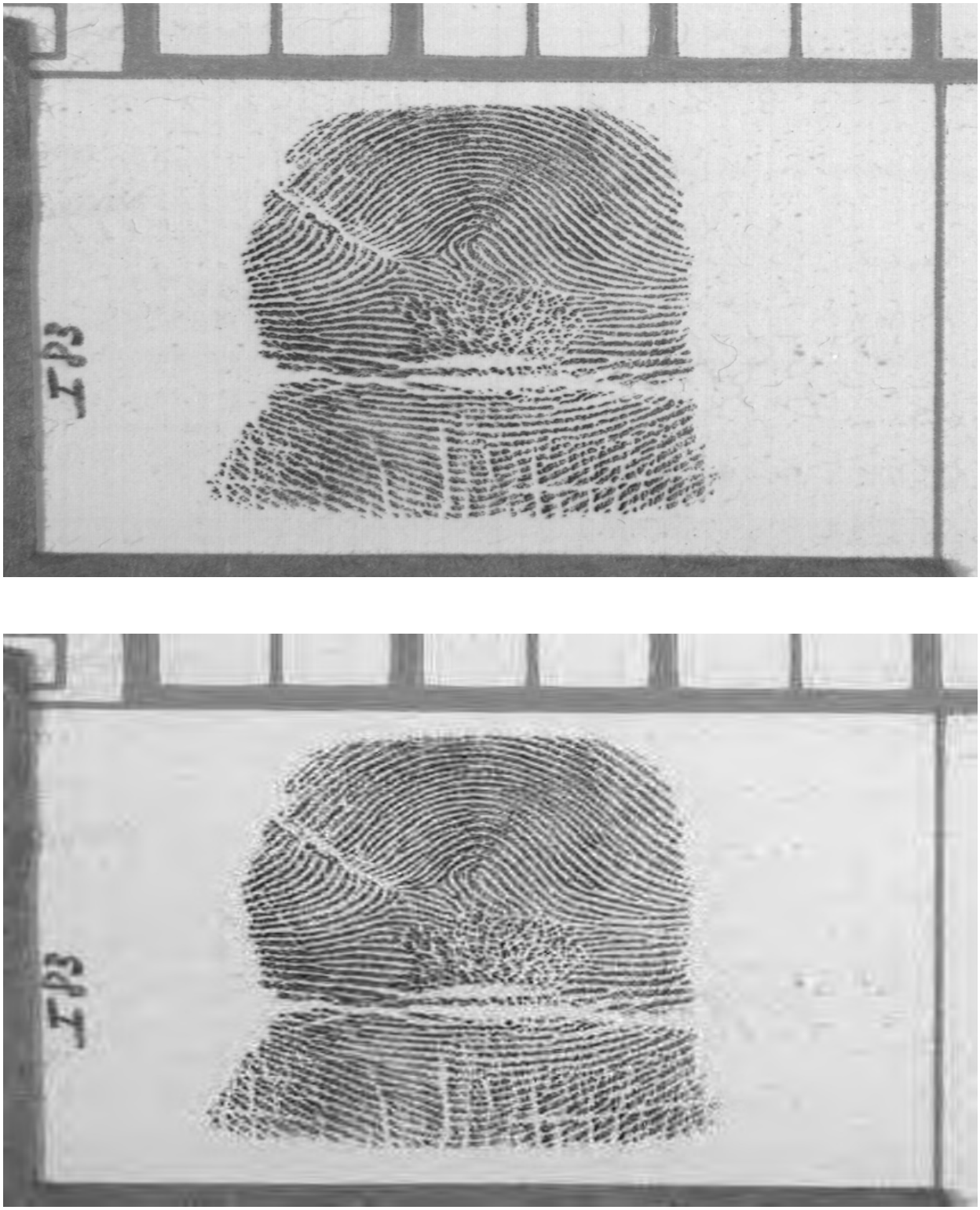


Figure 11: **Original (above) and WSQ-Recosntructed at 60:1 (below)**

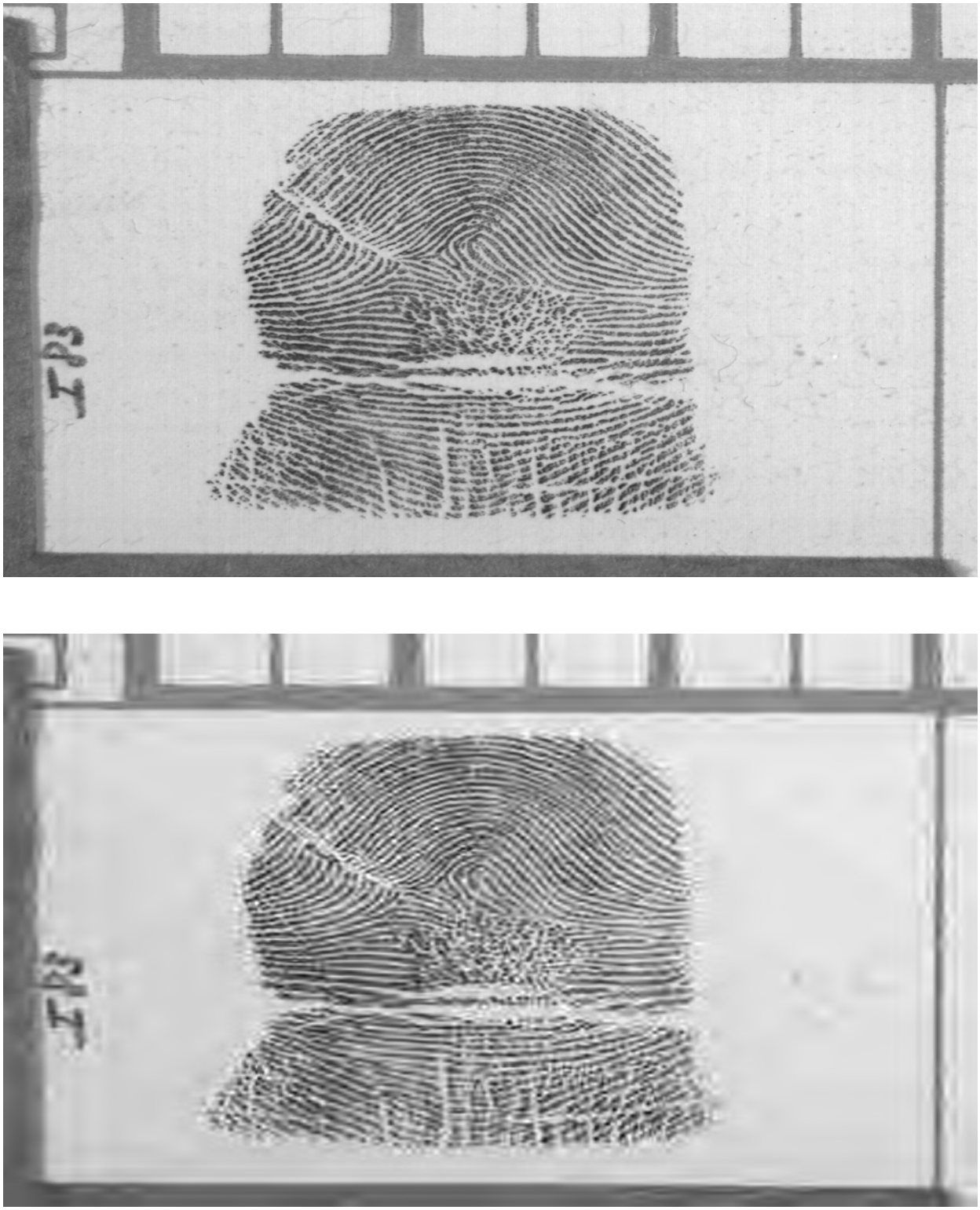


Figure 12: **Original (above) and WSQ-Recosntructed at 120:1 (below)**

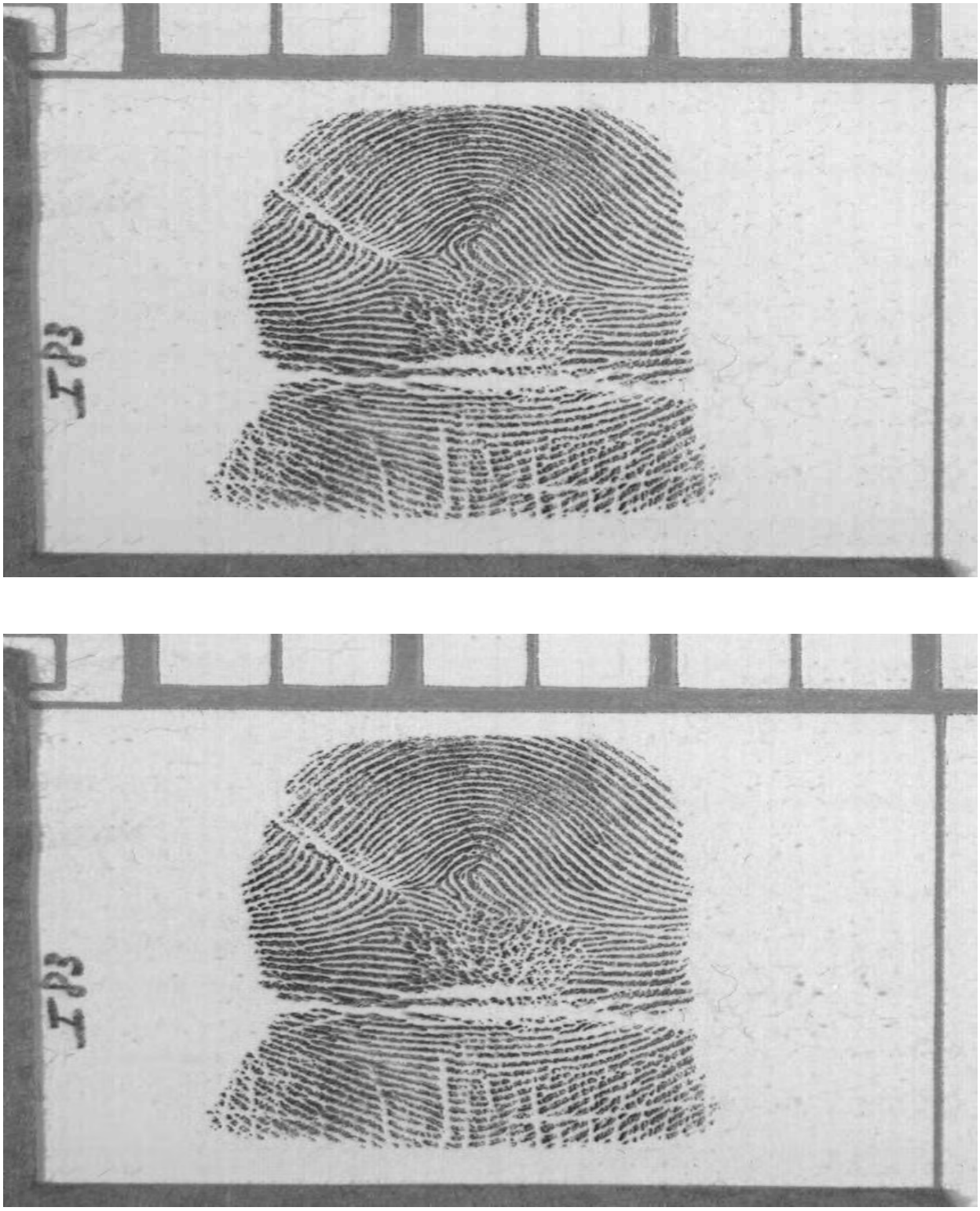


Figure 13: **JPEG at 15:1 (above) and WSQ at 15:1 (below)**

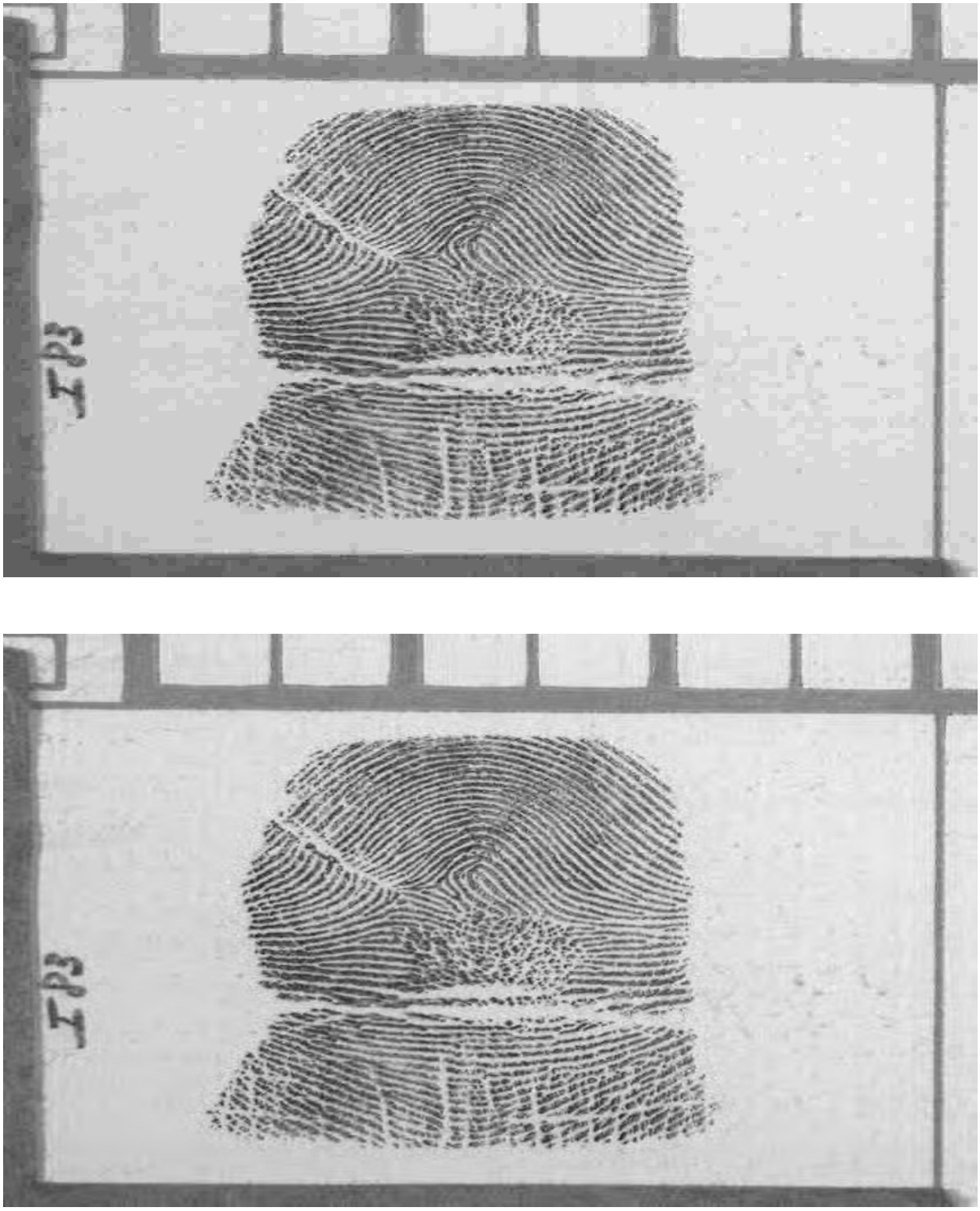


Figure 14: **JPEG at 30:1 (above) and WSQ at 30:1 (below)**

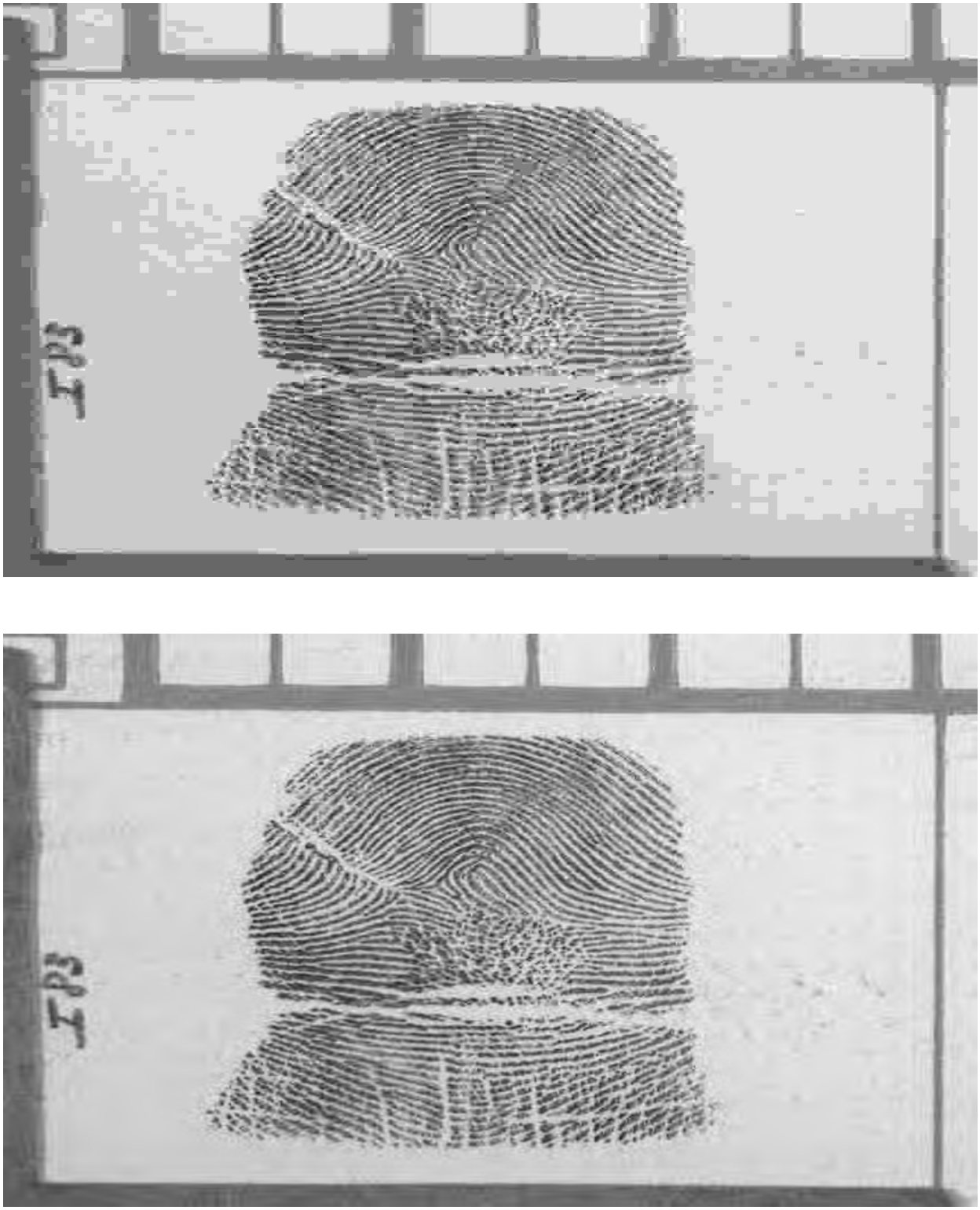


Figure 15: **JPEG at 45:1 (above) and WSQ at 45:1 (below)**

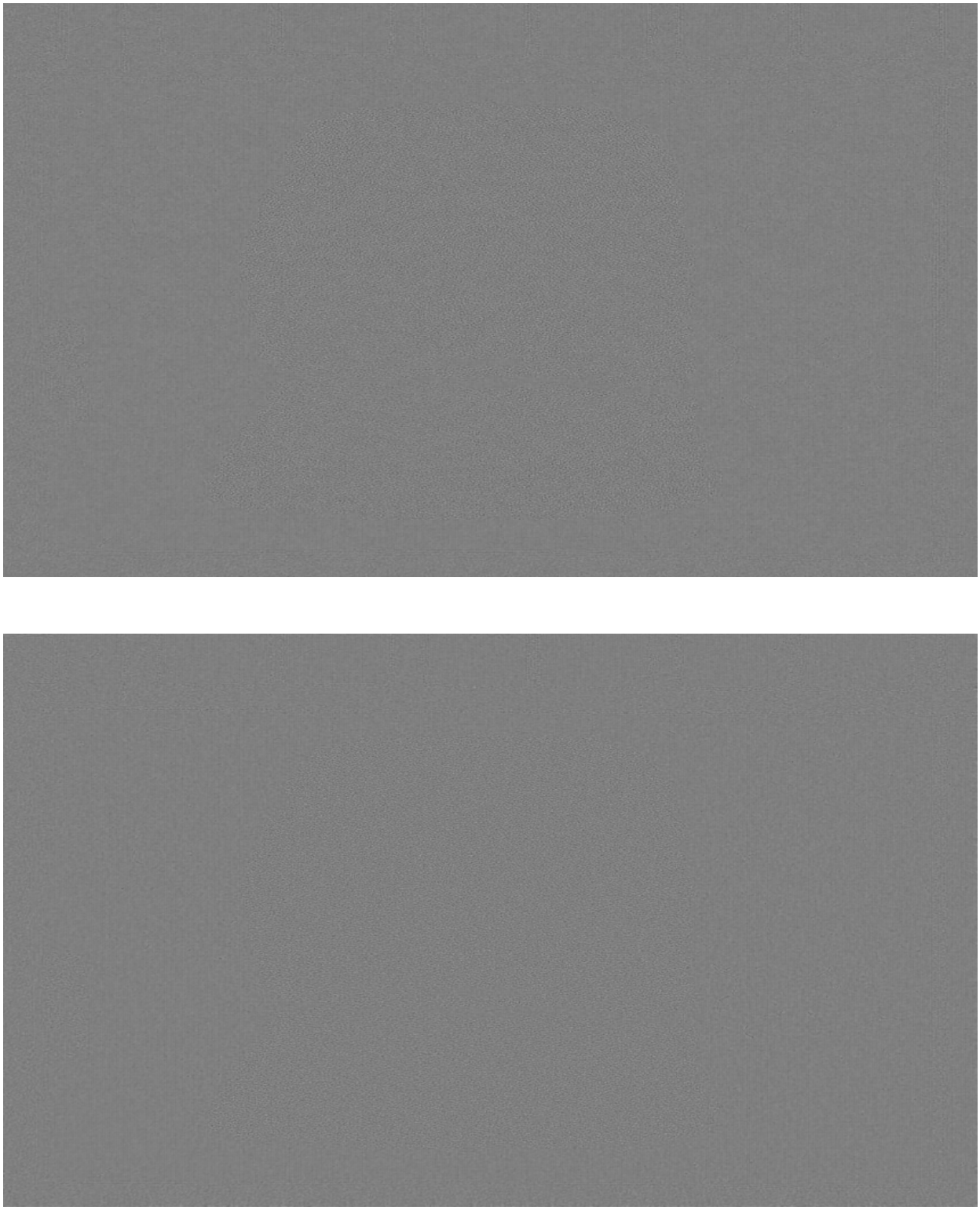


Figure 16: **The Error Images of JPEG at 15:1 (above) and WSQ at 15:1 (below)**

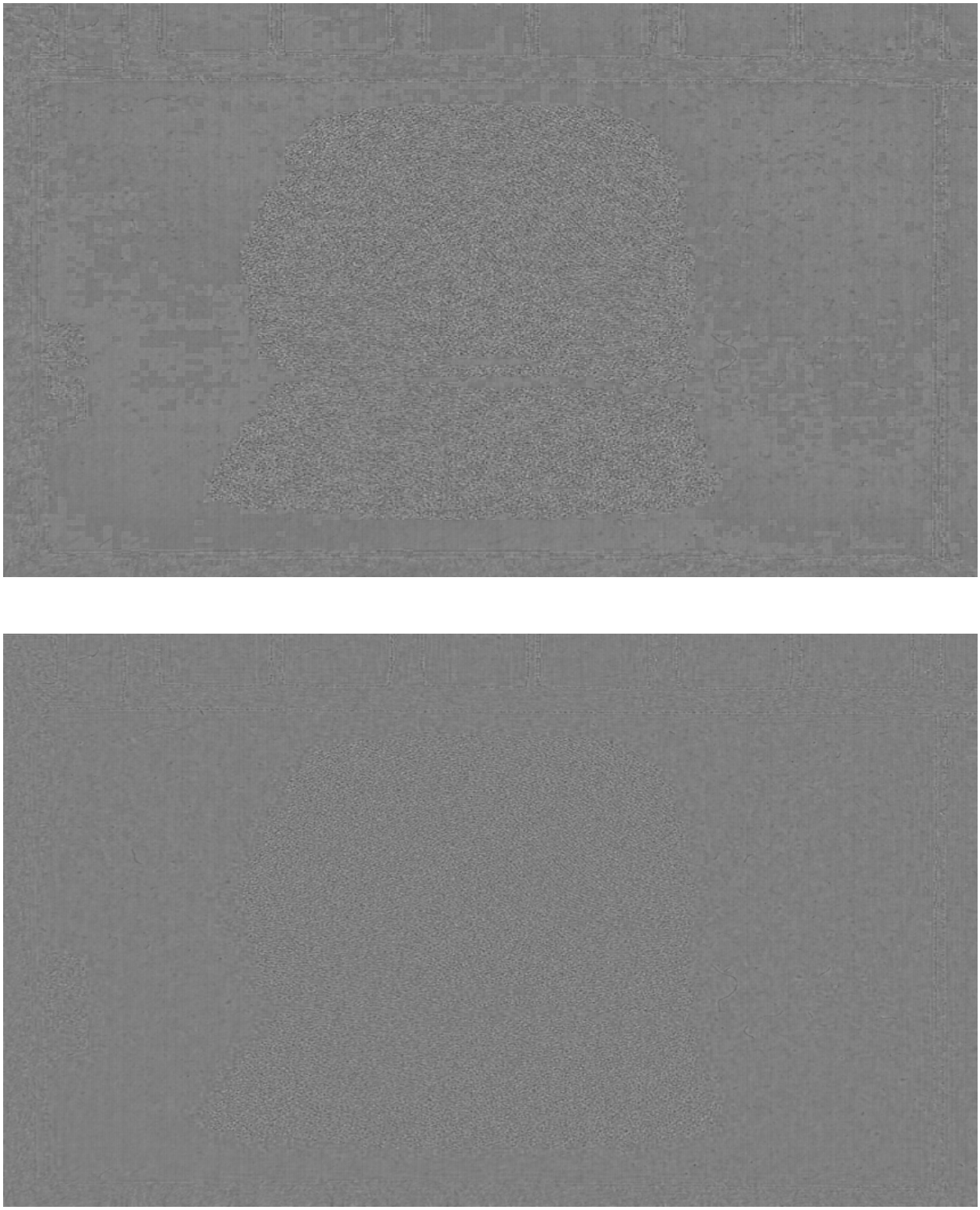


Figure 17: **The Error Images of JPEG at 30:1 (above) and WSQ at 30:1 (below)**

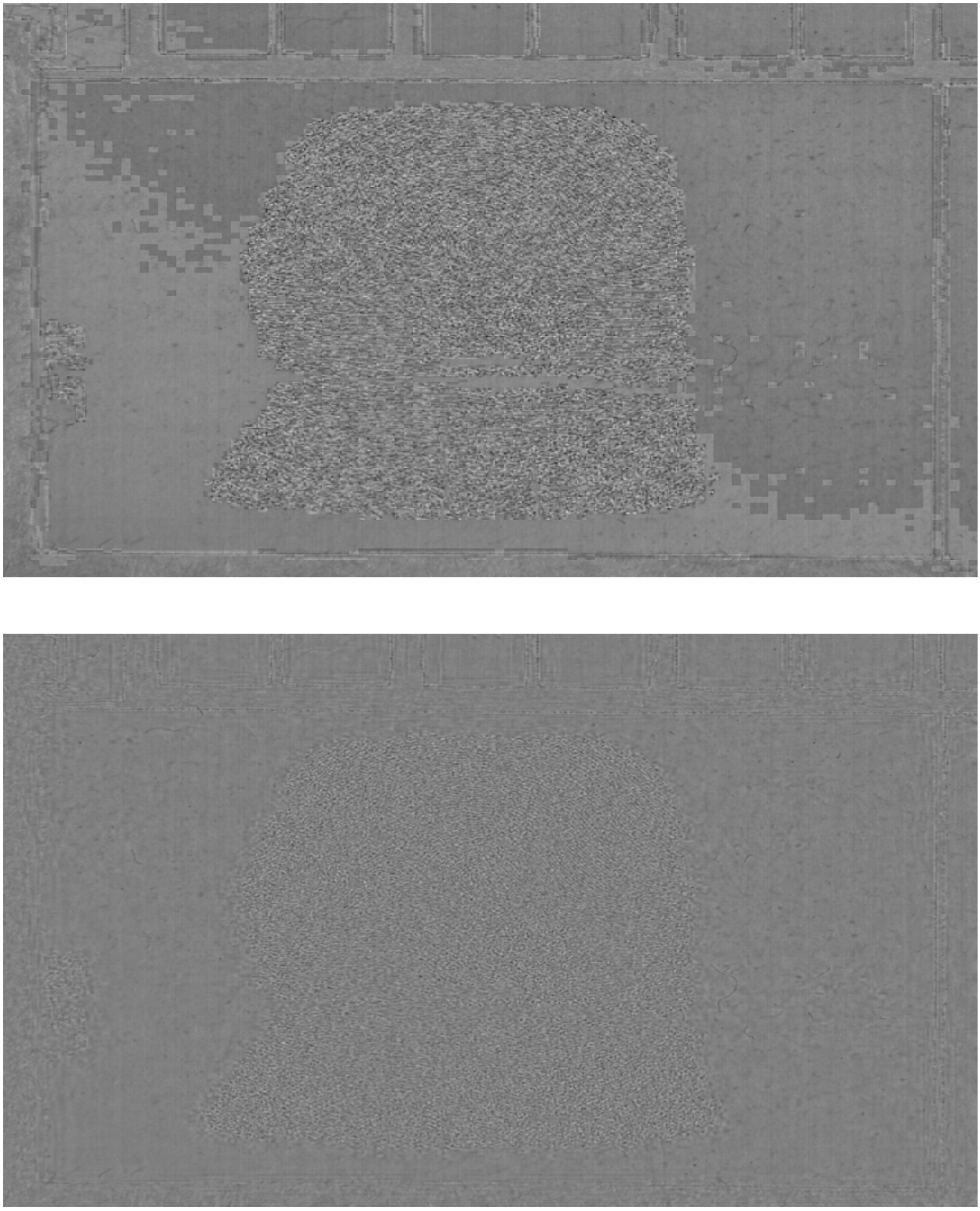


Figure 18: **The Error Images of JPEG at 45:1 (above) and WSQ at 45:1 (below)**

vendor for that implementation. The reference WSQ encoder/decoder software is used at NIST to review a vendor's submissions for WSQ implementation certification.

6 Summary

This chapter has addressed wavelet compression of images in general, and the FBI Wavelet Scalar Quantization Specification in particular. Choices for the wavelet filters, the decomposition trees, quantization, and entropy coding were discussed, and comparisons and tradeoffs between those choices were examined. In light of those considerations and the underlying theory of wavelet compression, the chapter shed some light on the careful choices made in WSQ. In addition, practical implementation aspects of WSQ were presented, and performance findings were briefly reported.

Much has developed in wavelet compression since the release of the WSQ Specification. Integer wavelet, lifting schemes, and quantization and bit-packing methods have been devised and shown to yield improved lossy compression performance and to allow for lossless and lossy-to-lossless modes of compression. JPEG 2000, which incorporates those ideas and more, has been finalized and released. It will be interesting to test those new methods and especially JPEG 2000 on fingerprint compression, and determine their performance improvement while preserving the ability to do fingerprint identification.

References

[1] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image Coding Using Wavelet Transform," *IEEE Trans. on Image Processing,* Vol. 1, No. 2, pp. 205–220, April 1992.

[2] M. F. Barnsley and L. P. Hurd, *Fractal Image Compression.* AK Peters, Wellesley, MA, USA, 1993.

[3] J. Bradley, C. Brislawn, and H. Topper, "The FBI Wavelet/Scalar Quatization Standard for Fingerprint Image Compression," *Proc. SPIE*, V. 1961, pp. 293–304, Orlando, Florida, Arp. 1993.

[4] C. Brislawn, "Fingerprints Go Digital," *Notices American Mathematical Society,* v.42, pp. 1278–1283, No. 1995

[5] C. Brislawn, J. Bradley, R. Onyshczak, and H. Topper, "The FBI Compression Standard for Digitized Fingerprint Images," *Proc. SPIE*, V. 2847, pp. 344–355, Denver, Colorado, Aug. 1996.

[6] C. K. Chui, *An Introduction to Wavelets*, Academic Press, 1992.

[7] R. J. Clarke, *Transform Coding of Images,* Academic Press, London, 1985.

[8] "MPEG-4:Coding of Moving Pictures and Audio," ISO/IEC 14496, 1999.

[9] R. R. Coifman and M. V. Wickerhauser, "Entropy-Based Algorithms for Best Basis Selection," *IEEE Trans. Info. Theory,* Vol. 38, No. 2, pp.713–718, March 1992.

[10] I. Daubechies, *Ten Lectures on Wavelets*, Society for Inductrial and Applied Mathematics, Philadelphia, 1992.

[11] I. Ersoy, F. Ercal and M. Gokmen, "A Model-Based Approach for Compression of Fingerprint Images", *Proc. IEEE Int'l Conf. on Image Processing, ICIP'99,* Kobe, Oct.1999 Japan, Vol.2, pp.973-977.

[12] *WSQ Gray-Scale Fingerprint Image Compression Specification,* Standard IAFIS-IC-0110v2, Criminal Justice Information Services, FBI, 1993.

[13] A. Gersho, "Quantization," *IEEE Communications Magazine,* 15, September 1977.

[14] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression.* Norwell, MA: Kluwer Academic Publishers, 1991.

[15] M. Gokmen, I. Ersoy and A.K. Jain, "Compression Of Fingerprint Images Using Hybrid Image Model", *Proc. IEEE Int'l Conf. on Image Processing, ICIP'96,* Lausanne, 1996, Vol.III, pp. 395-398.

[16] D. A. Huffman, "A Method for the Reconstruction of Minimum Redundancy Codes," *Proceedings of the IRE,* 40:1098-1101, 1951.

[17] S. P. Lloyd, "Least Squares Quantization in PCM," *IEEE Transactions on Information Theory,* IT-28, pp. 127–135, March 1982.

[18] Richard G. Lyons, *Understanding Digital Signal Processing.* Addison-Wesley, 1996.

[19] S.G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* pp. 674–693, July 1989.

[20] B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard,* Van Nostrand Reinhold, New York 1993.

[21] K. R. Rao and P. Yip, *Discrete Cosine Transform - Algorithms, Advantages and Applications*, Academic Press, New York, 1990.

[22] W. Sweldens, "The lifting scheme: construction of second generation wavelets," *SIAM J. Math. Anal.*, No. 2, vol. 29, pp. 511–546, 1997.

[23] H. Tanaka and A. Leon-Garcia, "Efficient Run-Length Encoding," *IEEE Trans. Info. Theory,* IT-28, No. 6, pp. 880–890, 1987.

[24] David S. Taubman, Michael W. Marcellin, *Jpeg2000 : Image Compression Fundamentals, Standards, and Practice.* Kluwer International Series in Engineering and Computer Science, Nov., 2001.

[25] A. H. Tewfik, D. Sinha, and P. Jorgensen, "On the Optimal Choice of a Wavelet for Signal Representation," *IEEE Trans. Info. Theory,* Vol. 38, No. 2, pp. 747-765, March 1992.

[26] J. Rissanen and G. Langdon, "Arithmetic Coding," *IBM J. Res. Develop.* 23, pp. 149–162, March 1979. Also in *IEEE Trans. Comm.* COM-29, No. 6, pp. 858–867, June 1981.

[27] K. Sayood, *Introduction to Data Compression,* Morgan Kauffmann Publishers, San Fransisco, California, 1996

[28] J. M Shapiro, "Embedded Image Coding Using Zerotrees of Wavelet Coefficients," *IEEE Transactions on Signal Processing,* vol. 41, No. 12, pp. 3445–3462, Dec. 1993.

[29] S. Srikanth and N.N. Murthy, "Adapting JPEG To Fingerprint Images", *Criminal Justice Information Services Technology Symposium (CJISTS'93),* sponsored by Federal Bureau Of Investigation and National Institute Of Standards And Technology, Gaithersburg, Maryland, USA.

[30] P. P. Vaidayanathan, *Multirate Systems and Filter Banks*, Prentice Hall, 1993.

[31] M. Vetterli and J. Kovacevic, *Wavelets and Subband Coding*, Prentice Hall, 1995.

[32] J. Villasenor, B. Belzer, and J. Liao, "Wavelet filter evaluation for efficient image compression," *IEEE Transactions on Image Processing,* vol. 4, pp. 1053–1060, 1995.

[33] J. W. Woods and S. D. O'Neal, "Subband Coding of Images," *IEEE Trans. Acous. Speech Signal processing,* ASSP-34, No. 5, pp. 1278–1288, 1986.

[34] A. Youssef, "Selection of Good Biorthogonal Wavelets for Data Compression," *International Conference on Imaging, Science, Systems, and Technology CISST '97,* Las Vegas, pp. 323-330, June 1997.

[35] J. Ziv and A. Lempel, "Compression of Individual Sequences via Variable Rate Coding," *IEEE Trans. Info. Theory,* IT-24, pp. 530–536, 1978.