

Chapter 19

Advances in Image and Video Compression Using Wavelet Transforms and Fovea Centralis



Juan C. Galan-Hernandez, Vicente Alarcon-Aquino, Oleg Starostenko,
Juan Manuel Ramirez-Cortes, and Pilar Gomez-Gil

Acronyms

ABAC	Adaptive binary arithmetic coding
AFV-SPECK	Adaptive fovea centralis set partitioned embedded block codec
AVC	Advance video coding
AWFV-Codec	Adaptive wavelet/fovea centralis-based codec
BGR	Blue green red color space
bpp	Bits per pixel
CDF9/7	Cohen–Daubechies–Feauveau wavelet
CIF	Common intermediate format
CMYK	Cyan magenta yellow black color space
CWT	Continuous wavelet transform
dB	Decibel
DCT	Discrete cosine transform
DWT	Discrete wavelet transform
FWT	Fast wavelet transform
FVHT	Fovea centralis hierarchical trees
GIF	Graphics interchange format
HEVC	High efficiency video coding
HVS	Human visual system

J. C. Galan-Hernandez · V. Alarcon-Aquino (✉) · O. Starostenko
Department of Computing, Electronics and Mechatronics, Universidad de las Americas Puebla,
Cholula, Puebla, Mexico
e-mail: juan.galan@udlap.mx; vicente.alarcon@udlap.mx; oleg.starostenko@udlap.mx

J. M. Ramirez-Cortes · P. Gomez-Gil
Department of Electronics and Computer Science, Instituto Nacional de Astrofisica, Tonantzintla,
Puebla, Mexico
e-mail: jmram@inaoep.mx; pgomez@inaoep.mx

iDCT	Integer discrete cosine transform
iSPECK	Inverse SPECK
iLWT	Inverse LWT
JBIG	Joint bi-level image group
JPEG	Joint photographic experts group
JPEG2000	Joint photographic experts group 2000
LCL	Lossless compression limit
LIP	List of insignificant pixels
LIS	List of insignificant sets
LSP	List of significant pixels
LWT	Lifting wavelet transform
MPEG	Moving picture experts group
MSE	Mean squared error
PCX	Personal computer exchange
pixel	Picture element
PNG	Portable network graphics
ppi	Pixels per inch
PSNR	Peak signal to noise ratio
RAR	Roshal archive file format
RGB	Red green blue color space
RLE	Run length encoding
ROI	Region of interest
SPECK	Set partitioned embedded block codec
SPIHT	Set partitioning in hierarchical tree
sRGB	Standard red green blue color space
SSIM	Structural similarity index
WebP	WebP
WT	Wavelet transform
Y'CBCR	Luma chrominance color space
ZIP	.ZIP file format

19.1 Introduction

The problem of storing images appeared along with the devices that allowed to capture and represent data in the form of visual information. Devices like image scanners (1950) and graphic processing units (1984) along with graphic manipulation software made possible to capture, create, and display images as digital images on a computer. A digital image is a numeric representation of a captured or software created image. This numeric representation is a discretization value made from a digital scanner device. The digital image can be represented as a two-dimensional numeric matrix. Each element of the matrix represents a small picture element (pixel). Such images are also known as raster images. Computer software such

as Adobe Photoshop¹ or Gimp² allow to create raster images. Also, most capturing devices like cameras and image scanners capture the image as a raster image [38, 39].

Compression algorithms are encapsulated with its decompression counterpart on digital image formats. A digital image format or standard specifies the following: a compression algorithm, a decompression algorithm, which color space is used for representing the image, how the data is stored inside a binary file and headers with metadata for image software [28]. Digital image formats that use only one two-dimensional matrix are used for storing black and white images or gray scale images. Digital color images on the other hand require more than one matrix in order to represent color. Usually, the number of matrices used are three for color spaces such as Red Green Blue color space (RGB) [20], Luma Chrominance color space ($Y' C_B C_R$) [33], and derivatives or four matrix for spaces such as Cyan Magenta Yellow Black color space (CMYK) [47]. Each matrix is known as a color channel. A common practice when using integer representation is to use one matrix of elements of 32 bits. The bits of each 32-bit element are split into four sets of 8 bits. Each 8-bit set is related to one color channel. When using a three channel color space, usually the four most significant bits set is either discarded as in RGB (or Blue Green Red color space (BGR) representation [62]) or used as a *transparency* information as in the Standard Red Green Blue color space (sRGB) format [20]. There are other digital image representations of an image such as vector images [48, 57]. However, this chapter will be focused only on raster type images. From now on the term image will be used to refer to digital raster images unless otherwise stated. Usually, the quality of an image grows as the amount of pixels taken per inch grows. This is known as pixels per inch (ppi).

Lossless compression is the best way to reduce the space needed to store a high quality image. Examples of such lossless compression algorithms are the Personal Computer Exchange (PCX) file format and the Graphics Interchange Format (GIF) file format. Nevertheless, it has been shown that the upper limit for an ideal lossless compression algorithm is around 30% [28]. Therefore, image file formats based on lossless compression algorithms are less convenient as the image increases in size. In consequence, new file formats were designed that take advantage of lossy compression algorithms. Lossy compression algorithms take into account the sensibility of the human visual system (HVS) in order to drop some of the details of the image when compressing. As a result, the reconstructed image is not the original image but a close representation of it. The aim of lossy compression is to build an algorithm that when reconstructing the image using the compressed stream, the reconstructed image will look almost the same for the user. Several lossy compression algorithms for images have been proposed; however, most of them are based on mathematical transformations that take the image matrix of color intensities and map it into a different space. The most common space used is the frequency space also known as frequency domain. When using the frequency domain, a matrix

¹<http://www.adobe.com>.

²<http://www.gimp.org>.

representing the intensity of each pixel of an image is considered to be in a spatial domain.

The Joint Photographic Experts Group (JPEG) and the JPEG2000 standards are examples of image lossy compression using a transform function [1, 50, 59]. Hybrid codecs based on the discrete cosine transform (DCT) are designed to attain higher compression ratios by combining loss and lossless compression algorithms. A modified version of the DCT is used in H.264/AVC (Advanced Video Coding) and H.265/HEVC (High Efficiency Video Coding) standards [6, 60, 63]. Algorithms based on the DCT are the one used in the JPEG file format [59] and the one used in the lossy definition of the WebP file format.³ The DCT is widely used because of its low computational complexity and its high quality when used for lossy compression. However, the wavelet transform (WT) shows better image reconstruction quality when used for lossy compression [5, 30]. By using the HVS based on fovea centralis, coding the quality of the reconstruction may be improved [12, 24, 40, 41]. Nowadays, few image formats use the WT for image compression. An example of these formats is the JPEG2000 file format [1].

There are several proposals for improving classic algorithms for current wavelet-based image compression methods such as the ones proposed in [13, 25]. However, there is no ideal algorithm that produces the best image reconstruction quality for any kind of image in any given application [3]. The reason is that when doing lossy compression, the algorithm must choose which details must be dropped in order to reach a given compression ratio. The main problem lies in which details to drop. For video compression, the problem of storage increases because a digital video is a set of several images, called frames, that represent a state of a taken video at a specific instant. Also, video file formats must store the information of sound, increasing the need of efficient lossy compression algorithms for images even if the sound is also compressed. Because sound compression is a related but different problem to image compression, from now on the rest of this chapter the term video compression will be used to refer only to the compression of the visual information or frames.

19.2 Data Compression

Lossless compression algorithms exploit the statistical behavior of the data to be compressed. The original data can be recovered perfectly after decompression. Lossless compression is used for generic data compression regardless of what the data represents images, sound, text, and so on. Current formats for data file compression like .Zip file format (ZIP) [44] and Roshal Archive file format (RAR) [45] use lossless compression algorithms. There are two main classifications of lossless compression: dictionary and statistical methods. These methods require data information. Information is an intuitive concept that deals with the acquisition

³<https://developers.google.com/speed/webp/?csw=1>.

of knowledge. This acquisition can be done in several ways such as through study, experience of past events, or in the form of a collation of data [3]. Thereof, an important aspect to take into account is how to measure data. Quantifying information is based on the observation of the content of a given message and evaluating how much it is learned from a previous pool of knowledge. The value of how much information is gained depends on the context. Claude Shannon, the precursor of information theory [18], proposed a way to measure how much information is transmitted or gain after transmitting a string of characters. Given an alphabet Σ , the amount of information H (entropy) from a string s is expressed in terms of the probability of each symbol where each symbol can be seen as a value of a random variable. The amount of information indicates how easily a set of data, in this case the given string, can be compressed. The entropy is expressed as

$$H(\mathbf{s}) = - \sum_{i=1}^n P_i \log_2 P_i \quad (19.1)$$

where n is the amount of symbols in the alphabet of s calculated as $n = |\Sigma|$ and P_i is the probability of the i -th symbol.

Equation (19.1) can be interpreted as the amount of information gained from a string. This is known as the data entropy. The term entropy was coined by Claude Shannon [45]. The name entropy was chosen because the same term is used in thermodynamics to indicate the amount of disorder in a physical system. The meaning in the thermodynamics field can be related to the information theory field by expressing the information gained from a string s as the different frequencies each symbol of the alphabet appears on the string s . Using Eq. (19.1), the redundancy of R in the data is defined by the difference between the largest entropy of a symbol set and its actual entropy [44] defined by

$$R(\mathbf{s}) = \log_2 n + H(\mathbf{s}). \quad (19.2)$$

How much a data stream can be compressed is defined in terms of its redundancy R . If the stream has a redundancy $R = 0$, the data cannot be further compressed. Thus, the aim of a lossless compression algorithm is, from a given data stream with $R > 0$, to create a compressed data stream where its redundancy $R = 0$. The main theorem of Shannon of source coding states that [45] a stream of data cannot be compressed further to a limit without lossless. Such limit defined in this chapter by lossless compression limit (LCL) denoted by ρ is defined using Eq. (19.1) as follows.

$$\rho(\mathbf{s}) = mH(\mathbf{s}) \quad (19.3)$$

where m is the amount of different symbols that appear on string s .

Dictionary and statistical coding algorithms use different approach to reduce the redundancy of a data stream. Dictionary methods encode the data by choosing strings and encoding them with a *token*. Each token is stored in a dictionary and is

associated with a particular string. An example of this is to use a numerical index for each word on a dictionary. For a dictionary of length of N , it will be needed an index with a size close to $\lceil \log_2 N \rceil$ bits. Dictionary methods perform better as the size of the data stream to be compressed tends to infinity [46]. There are popular methods of dictionary source coding such as LZ77, LZ78, and LZW [43]. A common implementation of LZ77 is the DEFLATE algorithm used by the Unix operating system. The statistical methods for compression use a statistical model of the data to be compressed. It assigns a code to each of the possible symbols of the stream. The performance of the algorithms is determined by how such codes are assigned. These codes are variable in size, and usually the shortest one is assigned to the symbol with the higher frequency on the data stream. There are different variable size codes that allow to assign codes to each symbol without ambiguity. One of the most popular methods is the Huffman code. The Huffman code uses the statistical model of the data in order to assign a unique variable size code to each symbol. Huffman code is used in current standards such as JPEG and Portable Network Graphics (PNG). However, Huffman code only produces ideal size codes when the probabilities of each symbol are a negative power of two [45]. Arithmetic encoding on the other hand is known for its better performance against the Huffman codes [44].

The main disadvantage of lossless coding is that it is bounded by Shannon's theorem (see Eq. (19.3)). However, a consequence of the Shannon's theorem is that if a data stream is compressed beyond the LCL, the new data stream begins to lose information and a reconstruction of the original data cannot be made [46]. As a result, lossy algorithms must be designed in order to select which data will be lost in the compression and how to get a close representation of the original data using the compressed data stream. The data selected to be discarded is usually the one that contains the fewer information possible about the data stream. Thereof, lossy coding algorithms are designed for specific data sets in order to be able to select which data is significant and which data will be discarded. There are several ways to design lossy compression algorithms. There are lossy algorithms that operate over the original mathematical domain of the given stream such as the run-length encoding (RLE) for images. However, the best algorithms known are those that its output is calculated when using a mathematical transform.

A mathematical transform is a function that maps a set into either another set or itself. Mathematical transforms used in lossy compression, specifically on sound and image compression, are projections from one space to another. The inverse of the chosen transform must be invertible in order to reconstruct a close approximation of the original data. The use of mathematical transform for compression is also known as transform coding. Transform coding is widely used in multimedia compression and is known as perceptual coding. The preferred transformations for perceptual coding are the ones that present graceful degradation [3]. This property allows to discard some of the data on the projected space while the inverse of the transform can reconstruct a close approximation of the original data. The most common functions for perceptual coding are the ones related to the Fourier transform. When using a Fourier-related transform, it is said that the transform translates the original data from the spatial

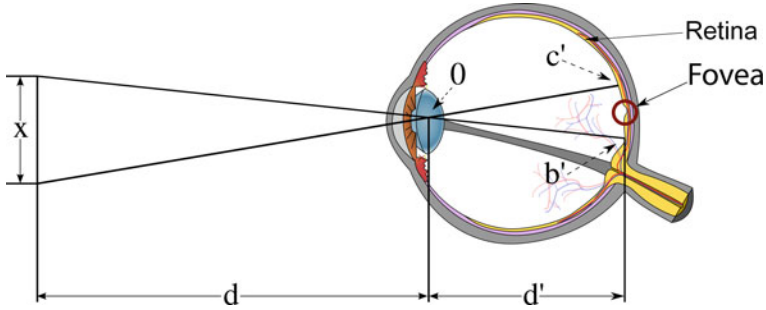


Fig. 19.1 Structure of the human eye

or time domain into the frequency domain. Being the data on the frequency domain which allows to discard some of the frequencies that are imperceptible to human perception. Hence the name of perceptual coding. Also, Fourier-related transforms degrade gracefully when decimal precision of some coefficients is lost. This allows to reduce the arithmetic precision of certain coefficients, thus reducing the number of bits required for its representation while retaining the most of the information of the original data. This process is known as *quantization*. The quantization method depends on how the data is represented by the transform on the frequency domain. There are several quantization algorithms for a given transform. The performance of the lossy compression algorithm depends on its transform and the quantization method. Several transforms have been proposed for multimedia coding such as the previously discussed DCT or the discrete WT (DWT).

Another common method used in lossy compression is the selection of regions of interest (ROIs) at different compression ratios. This feature mitigates such loss by conserving the details over a specific area. The ratio between the size of the compressed stream and the uncompressed stream is known as compression ratio [44]. In current standards such as MPEG4 and JPEG2000, ROIs can be defined [1, 19]. ROI-based algorithms are commonly used on image and video compression, and their main purpose is to assign more screen resources to a specific area [10]. ROIs are areas defined over an image selected by a given characteristic. ROI compression is when areas of the image are isolated using different desired final quality [19].

19.2.1 Fovea Centralis

The structure of the human eye (see Fig. 19.1) can also be exploited for compression. In applications wherever ROI is isolated, a selected part of the human eye called fovea centralis is utilized to increase the image quality for the human eye around ROI areas [49]. There are two main bodies on the tissue layer, particularly cones and rods. The number of cones in every eye varies between half a dozen and seven million. They are placed primarily within the central portion of the tissue layer, referred to as the fovea centralis, and are highly sensitive to color. The number of rods is

much larger, some seventy five to one hundred fifty million are distributed over the retinal surface. In Fig. 19.1, the circle between the points b' and c' marks wherever the cones reside, such area is termed fovea centralis. The larger area of distribution and the fact that several rods are connected to a single nerve reduce the amount of detail discernible by these receptors. The distance x in Fig. 19.1 is the area where the perception of an user would be the most acute, where the size of x is determined by the distance d between the observer and the image, and the distance d' between the retina and the back of the eye where the rods and cones reside. Anything outside of such area will be perceived with fewer details. This aliasing is exploited in fovea centralis compression. Fovea centralis compression can be applied over images with ROI; the use of fovea centralis around defined ROI improves the image quality for the human eye [15, 16, 24].

19.3 Wavelet Transforms

Fourier analysis is an useful tool for signal analysis. Fourier analysis is the study of general functions represented by using the Fourier transform [34]. The analysis is done by representing any periodic function as series of harmonically related sinusoids. It is useful in numerous fields, however it has some limitations [14]. Many of these limitations come from the fact that the Fourier basis elements are not localized in space. It is said that the basis of a transform is localized in space when its energy is concentrated around a given point. Accordingly, elements of the basis beyond certain radius will be 0 valued or close to 0. A basis that is not localized does not give information about how the frequency changes in relation to its position in time or space. There are refined tools that extend the capabilities of the Fourier transform in order to cover its weakness such as the windowed Fourier transform [27]. One mathematical tool that is able to analyze a signal and the structure of a signal at different sizes, thus yielding into information about the changes of frequency related to its position in time or space is the wavelet transform [2]. Time-frequency atoms are mathematical constructions that help to analyze a signal over multiple sizes. Time-frequency atoms are waveforms that are concentrated in time and frequency. The set of time-frequency atoms used for analyzing a signal is known as dictionary of atoms denoted by \mathfrak{D} . The wavelet transform builds this dictionary from a function $\psi(t) \in \mathbf{L}^2(\mathbb{R})$, where \mathbf{L}^2 is the Lebesgue space at power of 2, \mathbb{R} is the set of real numbers, and $\psi(t)$ denotes a wavelet function. ψ has several properties, it has zero average [27]

$$\int_{-\infty}^{\infty} \psi(t)dt = 0. \quad (19.4)$$

It is normalized $\|\psi\| = 1$ and centered in the neighborhood of $t = 0$. ψ is known as the *mother wavelet*. In order to create a dictionary \mathfrak{D} , ψ is scaled by ℓ and translated by u , namely [27]

$$\mathfrak{D} = \left\{ \psi_{\ell,u}(t) = \frac{1}{\sqrt{\ell}} \psi \left(\frac{t-u}{\ell} \right) \right\}_{u \in \mathbb{R}, \ell > 0} \tag{19.5}$$

The atoms remain normalized $\|\psi_{\ell,u}\| = 1$. The constant $\frac{1}{\sqrt{\ell}}$ is for energy normalization. The continuous wavelet transform (CWT) ω of $f(t) \in \mathbf{L}^2(\mathbb{R})$ at time u and scale ℓ is

$$\omega_{\ell,u}(f) = \langle f, \psi_{\ell,u} \rangle = \int_{-\infty}^{\infty} f(t) \frac{1}{\sqrt{\ell}} \psi^* \left(\frac{t-u}{\ell} \right) dt \tag{19.6}$$

where ψ^* is the complex conjugate of the mother wavelet ψ and $\langle \cdot, \cdot \rangle$ denotes an inner product.

Because images are two-dimensional signals, a two-dimensional wavelet transform is needed. Let $\bar{\psi}_{\ell,u}$ be

$$\bar{\psi}_{\ell,u}(t) = \frac{1}{\sqrt{\ell}} \psi^* \left(\frac{t-u}{\ell} \right), \tag{19.7}$$

extending Eq. (19.6) to two dimensions, the wavelet transform at parameters u_v, ℓ_v, u_h, ℓ_h of $f(t, x) \in \mathbf{L}^2(\mathbb{R}^2)$ yields into

$$\begin{aligned} \omega_{\ell_v, u_v, \ell_h, u_h}^2(f) &= \langle \langle f, \bar{\psi}_{\ell_v, u_v} \rangle, \bar{\psi}_{\ell_h, u_h} \rangle \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t, x) \bar{\psi}_{\ell_v, u_v} \bar{\psi}_{\ell_h, u_h} dt dx. \end{aligned} \tag{19.8}$$

where ω is the wavelet operator. Also, because digital images are stored as a discrete finite signal, a discrete version of the CWT is needed. Let $f[n]$ be a discrete signal obtained from a continuous function f defined on the interval $[0, 1]$ by a low-pass filtering and uniform sampling at intervals N^{-1} . The DWT can only be calculated at scales $N^{-1} < \ell < 1$. Also, let $\psi(n)$ be a wavelet with a support included in $[-K/2, K/2]$. For $1 \leq \ell = a^j \leq NK^{-1}$, a discrete wavelet scaled by a^j is defined by [27]

$$\psi_j[n] = \frac{1}{\sqrt{a^j}} \psi \left(\frac{n}{a^j} \right). \tag{19.9}$$

The DWT is defined by a circular convolution with $\bar{\psi}_j[n]$ defined as $\bar{\psi}_j[n] = \psi_j^*[-n]$ with DWT described as

$$\omega_{a^j} f[n] = \sum_{m=0}^{N-1} f[m] \psi_j^*[m-n] = f * \bar{\psi}_j[n] \tag{19.10}$$

where $*$ is the convolution operator. Also, signal f is assumed to be periodic of length N in order to avoid border problems.

In order to speed up the computation of the wavelet coefficients, a second approach that simplifies the DWT is referred to as the *lifting scheme*. The lifting scheme [32, 51] is another way of looking at the DWT, where all the operations are performed in the time domain [1]. Computing the wavelet transform using lifting steps consists of several stages. The idea is to compute a trivial wavelet transform (the lazy wavelet) and then improve its properties by alternating the dual lifting or prediction step and the primal lifting or updating step [44]. The lazy wavelet only splits the signal into its even and odd indexed samples, namely

$$(even[n - 1], odd[n - 1]) = Split(f[n]). \quad (19.11)$$

where $f[n]$ is a given discrete signal, *even* and *odd* are the even and odd signals of the lazy wavelet, and *Split* is the split function. A dual lifting step consists of applying a filter to the even samples and subtracting the result from the odd ones. This is based on the fact that each value $f[n]_{2\ell+1}$ of the next decomposition level in the odd set is adjacent to the corresponding value $f[n]_{2\ell}$ in the even set, where ℓ is the decomposition level. Thus, the two values are correlated and any can be used to predict the other. The prediction step is given by

$$d[n - 1] = odd[n - 1] - P(even[n - 1]). \quad (19.12)$$

where d is the difference signal of the *odd* part of the lifting wavelet and the result of the prediction P operator applied to the *even* part of the lazy wavelet. A primal lifting step does the opposite: applying a filter to the odd samples and adding the result to the even samples. The update operation U follows the prediction step. It calculates the $2[n - 1]$ averages $s[n - 1]_\ell$ as the sum

$$s[n - 1] = even[n - 1] + U(d[n - 1]). \quad (19.13)$$

where U is defined by

$$s[n - 1]_\ell = f[n]_{2\ell} + \frac{d[n - 1]_\ell}{2}. \quad (19.14)$$

The process of applying the prediction and update operators is repeated as many times as needed. Each wavelet filter bank is categorized by its own U operator and the amount of rounds of the process. The calculation process of U is described in [26]. This scheme often requires far fewer computations compared to the convolution-based DWT, and its computational complexity can be reduced up to 50% [1, 11, 53]. As a result, this lifting approach has been recommended for implementation of the DWT in the JPEG2000 standard.⁴

⁴JPEG2000 draft at <http://www.jpeg.org/public/fcd15444-1.pdf>.

19.4 Image Compression

Image data compression is concerned with coding of data to minimize the number of bits used to represent an image. Current image compression standards use a combination of lossless and lossy algorithms. These can be used over the same data set because both algorithms exploit different properties of the image. On the one hand, lossless-based compression exploits data redundancy, but on the other hand, lossy-based compression exploits its transform properties and quantization. The simplest quantization equation used in image coding is defined as [4, 17]

$$C^q = \left\lfloor \frac{1}{\Delta q} C \right\rfloor \tag{19.15}$$

where $\lfloor \cdot \rfloor$ is the floor operation, $\Delta q > 1$ is known as the *quantization delta*, C is the matrix of coefficients obtained from applying a transform to the given image, and C^q is the matrix of quantized coefficients. Spatial redundancy takes a variety of different forms in an image. For example, it includes strongly correlated repeated patterns in the background of the image and correlated repeated base shapes, colors, and patterns across an image. The combination of lossy and lossless compression allows achieving lower compression ratios. Figure 19.2 shows a block diagram of the classic lossy/lossless image coding scheme [4, 17].

In Fig. 19.2, the image is interpreted as a matrix I , then the coefficients matrix C of the chosen transform is calculated. Subsequently, the quantized coefficient matrix C^q is calculated and the final lossless compressed stream S is calculated on the entropy coding block. The color space used for image compression is often the $Y' C_B C_R$ color space. This color space is chosen because it has been found that the human eye is more sensitive to changes on the luma channel (Y') than on the color channels ($C_B C_R$) [33]. This allows compressing at lower ratios the color channels than the luma channel. As a result, compression algorithms are evaluated over the luma channel only. Thereof, all the analyses of the algorithms presented in this chapter are evaluated on the luma channel. The equation used for calculating $Y' C_B C_R$ from other common color space RGB suggested in [17] is the following:

$$\begin{bmatrix} Y' \\ C_R \\ C_B \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & 0.515 & 0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{19.16}$$

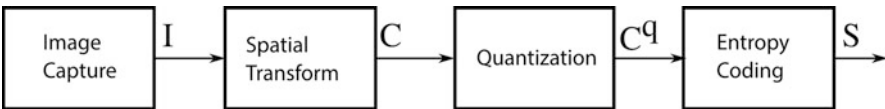


Fig. 19.2 Block diagram of the classic image coding scheme [17]

where R , G , B are the values for each channel on the RGB color space of a given pixel. From Eq. (19.16), the luma is calculated as

$$Y' = 0.299R + 0.587G + 0.114B. \quad (19.17)$$

One of the foremost image compression algorithms is the JPEG image coding standard (see Fig. 19.3). Outlined in [59], JPEG framework defines a lossy compression and a lossless compression algorithm used in tandem. The lossy compression algorithm of JPEG uses the DCT. However, in order to reduce the complexity of the algorithm [4], the image is split into non overlapping blocks of 8×8 pixels. Each block is referred to as *macroblock*. Processing macroblocks requires less computation and allows the algorithm to optimize transmission by sending the data of processed macroblocks while processing the rest of the images [19].

In Fig. 19.3, the RGB image is transformed into the $Y' C_B C_R$ color space. Then, the image is split into macroblocks. Then, the DCT macroblock applies the transform to each macroblock individually. After the transformation of a macroblock is calculated, the coefficients are quantized by a fixed ratio. JPEG standard defines a quantization matrix. Because each coefficient has a different significance on the reconstruction of the image, the quantization matrix stores a quantization ratio for each of the coefficient of a macroblock. The standard provides with values for the quantization matrix. However, some manufacturers defined its own quantization matrices in order to improve the quality of the algorithm. After quantization, the next step is resorting each macroblock into zigzag order. This allows to exploit the entropy of the lower diagonal of the macroblocks [1]. The last block of the algorithm applies lossless compression to the quantized sorted coefficients. Early versions of the algorithm define RLE and Huffman coding as the lossless algorithms for JPEG. However, the last version of JPEG [45] also includes arithmetic coding in order to reduce the compression ratio. The final overall quality of JPEG is mostly given by the

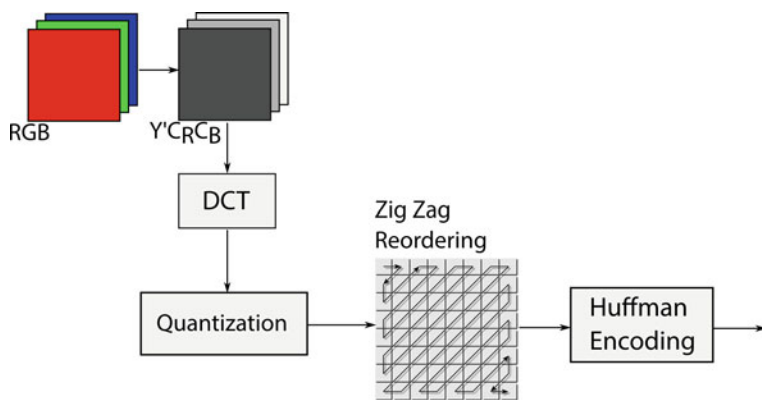


Fig. 19.3 Block diagram of the JPEG2000 standard [59]

quantization matrix, however it is not possible to precalculate the final compression ratio [43].

It is well-known that in compression applications, wavelet-based approaches outperform the block-DCT methods [22, 35]. This is due to the fact that wavelet-based approaches can reduce the blocking artifacts, provide better energy compaction because of the multi-resolution feature of wavelet basis, and have better correspondence with the HVS system [58]. Therefore, wavelet-based compression algorithms have been recommended for the JPEG2000 standard [17, 45].

19.4.1 Foveated Images

Images with a non-uniform resolution that have been used in image and video compression are known as foveated images. Equation (19.18) shows a representation of a foveated image [9].

$$I_x^0 = \int I_x c^{-1}(x) s \left(\frac{t-x}{\omega(x)} \right) dt \tag{19.18}$$

where $c(x) = \left\| s \left(\frac{-x}{\omega(x)} \right) \right\|$, I_x is the pixel at position x of a given image, $\omega(x)$ is a weight function, and I_x^0 is the foveated image. The function s is known as the weighted translation of s by x [24]. A variation of the fast wavelet transform (FWT) is reported in [7] that operates over the wavelet transform. For an image I , its foveation is given by

$$I^0 = \langle I, \Phi_{\ell_0,0,0} \rangle + \sum_{u_v, \ell_v, u_h, \ell_h} c_j^k[\ell_v, u_h] \langle I, \Psi_{\ell_v, u_h, \ell_h}^{u_v} \rangle \Psi_{\ell_v, u_h, \ell_h}^{u_v} \tag{19.19}$$

where $\Phi_{\ell_0,0,0}$ is the father wavelet, and $\Psi_{\ell_v, u_h, \ell_h}^{u_v}$ is the mother wavelet scaled and translated with $u_v = \{h, v, d\}$ and the operator $\langle \cdot, \cdot \rangle$ is the convolution operator. $c_j^k[\ell_v, u_h]$ is defined as

$$\begin{aligned} c_j^k[\ell_v, u_h] &= \langle T \Psi_{0, \ell_v, u_h}^{u_v}, \Psi_{0, \ell_v, u_h}^{u_v} \rangle \\ &= \int_{-\infty}^{\infty} dy \int_{-\infty}^{\infty} dx \Psi_{0, \ell_v, u_h}^{u_v}(x, y) \int_{-\infty}^{\infty} dt \int_{-\infty}^{\infty} ds \Psi_{0, \ell_v, u_h}^{u_v}(s, t) g_{\omega(x,y)}(s, t) \end{aligned} \tag{19.20}$$

where T is the fovea centralization operator and $g_{\omega(x,y)}(s, t)$ is the smoothing function defined as

$$g_{\omega(x,y)}(s, t) := \frac{1}{\omega(x, y)^2} g \left(\frac{s-x}{\omega(x, y)}, \frac{t-y}{\omega(x, y)} \right). \tag{19.21}$$

where the weight function $\omega(x, y)$ is defined by

$$\omega(x, y) = \alpha \| (x, y) - (\gamma_1, \gamma_2) \|_2 + \beta \tag{19.22}$$

where α is the rate, $\gamma = (\gamma_1, \gamma_2)$ is the fovea centralis, and β is the fovea centralis resolution [7].

19.5 Video Compression

Because video is just a sequence of several images called frames, video coding algorithms or video codecs use image compression extensively. To achieve high compression ratios is suitable to combine lossy and lossless compression algorithms. Classic video coding frameworks have three main algorithms (see Fig. 19.4), namely intra-frame coding (spatial transform and inverse spatial transform), inter-frame coding (motion estimation and compensation), and variable length coding (variable length coder).

In intra-frame coding, which uses the information of previous or future frames, a frame of a video stream is normally compressed using lossy algorithms. The encoder should work out the variations (prediction error) between the expected frame and the original frame. The first step in the motion compensated video coder is to create a motion compensated prediction error of the macroblocks. This calculation requires only a single frame to be stored in the receiver. Notice that for color images, motion compensation is performed only for the luma component of the image. The decimated motion vectors obtained for luma are then exploited to form motion compensated chroma components. The resulting error signal for each of the components is transformed using DCT, quantized by an adaptive quantizer, entropy

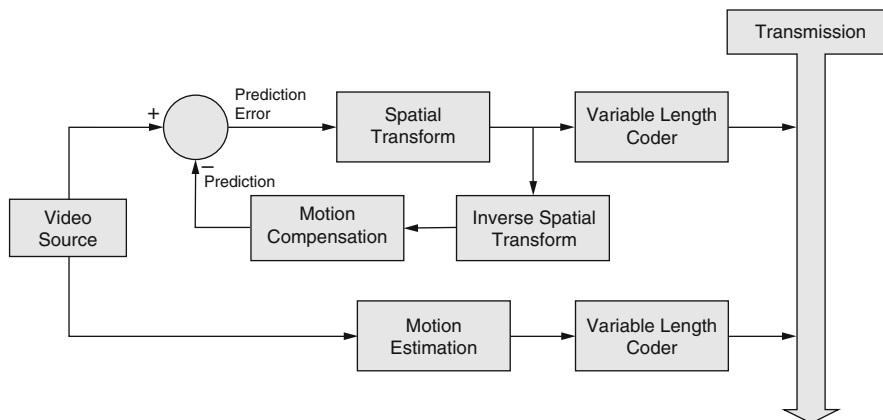


Fig. 19.4 Block diagram of the classic video coding framework [6]

encoded using a variable length coder, and buffered for transmission over a fixed-rate channel. The main problem of the block matching motion compensation is its high computational complexity.

Most video coding standards such as the H.264 [36] or the newest proposed standard H.265/HEVC codec [52] rely on the DCT for lossy intra-frame coding applied to macroblocks of a dimension of 4×4 . The smaller macroblock allows reducing artifacts on the reconstructed image [37]. However, in order to improve the speed of the algorithm, the transform used is the integer discrete cosine transform (IDCT) [8]. The IDCT is an approximation of the DCT used in JPEG standard. Instead of calculating a convolution, two different matrices are defined that are an approximation of the base of the DCT.

19.6 An Approach to Image Compression Based on ROI and Fovea Centralis

Image compression within the frequency domain based on real-valued coefficients is carried out through coefficient quantization. In this process of quantization, these coefficients become integer-valued for further compression employing either a RLE or an arithmetic encoding algorithms, which are known as variable quantization algorithms. The variable quantization algorithm exploits the fovea centralis result of the HVS based on a fovea centralis window, which is focused at a given fixation point to see a way to quantize each wavelet coefficient [15]. A modified version of the set partitioning in hierarchical tree (SPIHT) algorithm is utilized to quantize and compress these coefficients.

Figure 19.5 shows the block diagram of the compression approach based on ROIs and fovea centralis called here fovea centralis hierarchical tree (FVHT) algorithm. Assuming a video stream with frames F_i , the applied blocks can be described as follows [15]. In the *Motion estimation* block, the fovea centralis points are estimated using video frames F_i and F_{i-1} . The *ROI estimation* block outputs an array of fovea centralis points as ROI_i , where each pixel different of 0 is taken as a fovea centralis. The fovea centralis cutoff window is described in [15]. The *Lifting Wavelet Transform (LWT)* block generates the coefficients denoted as $C(\cdot)_i$ (see Sect. 19.3). The *Quantization* block maps to integers the coefficients $C(\cdot)_i$ into $C(\cdot)_i^q$ using a fixed quantization for compression. Finally, the *FVHT* block outputs a compressed stream of the quantized coefficients $C(\cdot)_i^q$ using the information of the estimated fovea centralis points ROI_i .

Note that the fovea centralis points ROI_i are input parameters to the FVHT rather than using the motion estimation block. The window parameters and the cutoff window are calculated as long as there is a fixation point [15, 24]. The reported method permits defining ROIs of variable size around the fixation point that retains the best quality. Further details on the approach described here can be found in [15].

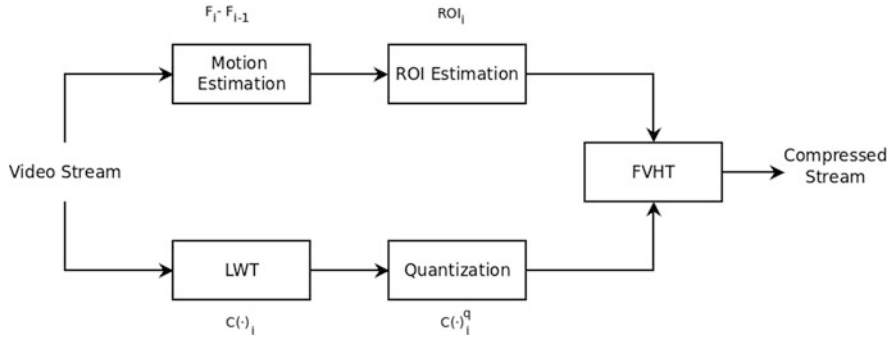


Fig. 19.5 Block diagram of compression approach based on ROI and fovea centralis [15]

19.6.1 FVHT Algorithm

The compression bit rate can be computed by assessing the decaying window function on each algorithm pass at each coefficient coordinate as it is proposed in the FVHT algorithm [15]. First, the coefficient is encoded whether the current bit rate is lower than wavelet subband, otherwise it is discarded. The sorting pass is modified in order to classify the coefficients according to its distance to the scaled fovea centralis and the cutoff window. Each time an attempt to add a coefficient to the list of significant pixels (*LSP*) is done, the assigned bits per pixel (bpp) is calculated, and the coefficient is classified. However, it should be noted that on the significance pass, the positions of the coefficients are discarded from the list of insignificant pixels (*LIP*) and on the refinement pass, they are discarded from the *LSP*. The list of insignificant sets (*LIS*) will remain the same as in the SPIHT algorithm [15, 42]. The execution time of the algorithm was analyzed using *Big O* notation, concluding that the complexity of the algorithm is linear ($O(n)$) [15]. The memory usage was also analyzed, yielding a size of $\frac{71}{64}n$. The FVHT is memory intensive when compared with classic methods based on the DCT transform that can be computed using no extra storage.

19.6.2 Simulation Results

The FVHT algorithm is assessed using standard non-compressed 512×512 images. The fovea centralis is defined at the center pixel (256,256) with two parameters, namely a radius of the ROI and the power law function (the ramp function), which are defined in [15]. As stated in the JPEG2000 standard and for a fair comparison, the biorthogonal Cohen–Daubechies–Feauveau (CDF) 9/7 is considered using four levels of decomposition [1]. The reported results are compared against the SPIHT algorithm. Figures 19.6 and 19.7 show the reconstructed wavelet coefficients of the

Fig. 19.6 Reconstructed image (“cameraman”) using SPIHT algorithm at 1 bpp compression ratio [15]

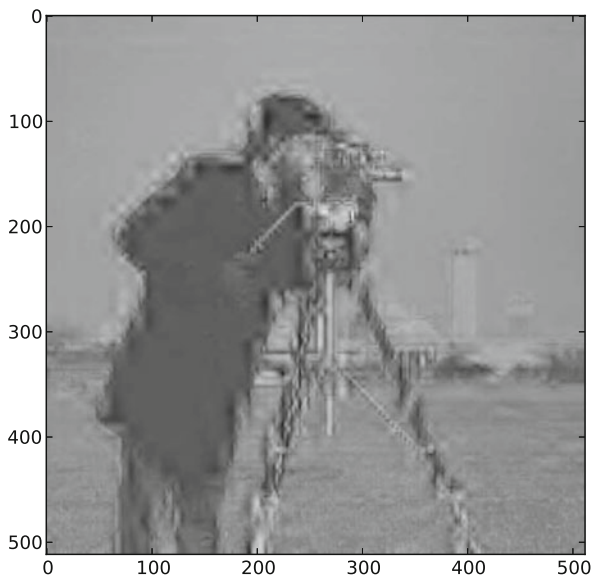
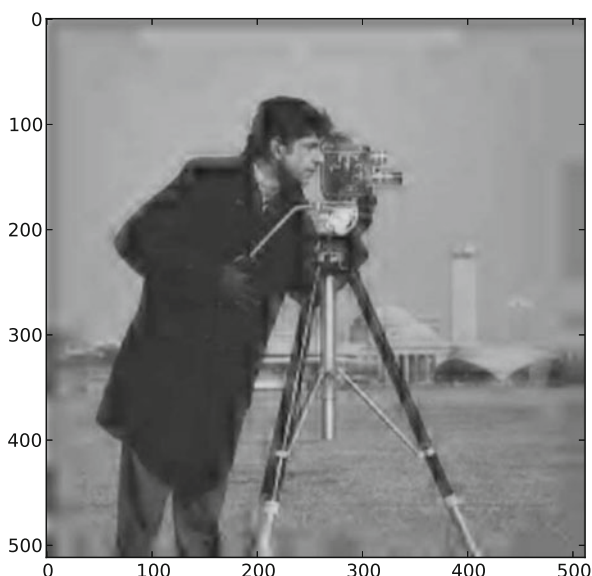


Fig. 19.7 Reconstructed image (“cameraman”) using FVHT compression algorithm at 0.06–1 bpp compression ratio [15]. Fovea centralis at (256,256)



cameraman image at 1 bit per pixel (bpp) with both SPIHT and FVHT, respectively. The same reconstructed wavelet coefficients at 1 bpp as its higher compression ratio and 0.06 bpp as its lower compression ratio are shown in Fig. 19.7. It is observed that the FVHT algorithm has better performance than SPIHT algorithm particularly over small areas around the fovea centralis or those closer to the fixation point. Further details on this approach can be found in [15].

19.7 Wavelet-Based Coding Approaches: SPECK-Based Codec and Adaptive Wavelet/Fovea Centralis-Based Codec

Two wavelet-based coding approaches based on the LWT [27] are described in this section [16]. The first called Set Partitioned Embedded bloCK (SPECK)-based codec (SP-Codec) is shown in Fig. 19.8 [31]. In the *Z-order block*, all coefficients position are organized and mapped from 2D to 1D using the Z-transform. The quantization step is carried out on *LWT and SPECK* blocks. The *adaptive binary arithmetic coding (ABAC)* block, which is a lossless compression algorithm, allows compressing a data stream while at the same time computes the statistical model (see Sect. 19.7.1) [31]. The *inverse LWT (iLWT) and inverse SPECK (iSPECK)* are applied to the compressed stream generated in the SPECK block, and finally the *motion compensation and estimation* blocks compute the motion vectors based on the block matching algorithm for each inter-frame.

The second proposal referred to as adaptive wavelet/fovea centralis-based codec (AWFV-Codec) reported in [16] aims to further increase the quality of the decoded frames (see Fig. 19.9). The reported adaptive fovea centralis-SPECK (AFV-SPECK) algorithm defines a center, a ROI area radius, and a decaying window [15, 16] and as a result various compression ratios may be considered. An external subsystem is assumed that computes the fovea centralis point of one observer, which is later provided to the AFV-SPECK coding algorithm.

19.7.1 Adaptive Binary Arithmetic Coding

The adaptive binary arithmetic coding (ABAC) is a version of the arithmetic coding algorithm applied to an alphabet with only two elements $\Sigma = 0, 1$ [64]. This application is commonly used for bi tonal images [23]. Also, it does not require a previously calculated statistical model. Each time a symbol is read, the statistical

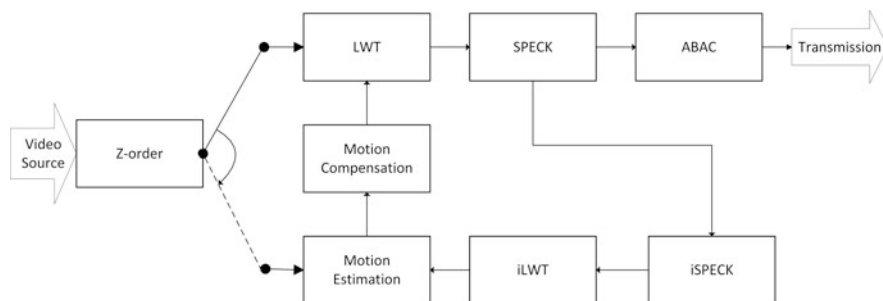


Fig. 19.8 Video coding framework SPECK-based codec (SP-Codec) [16]

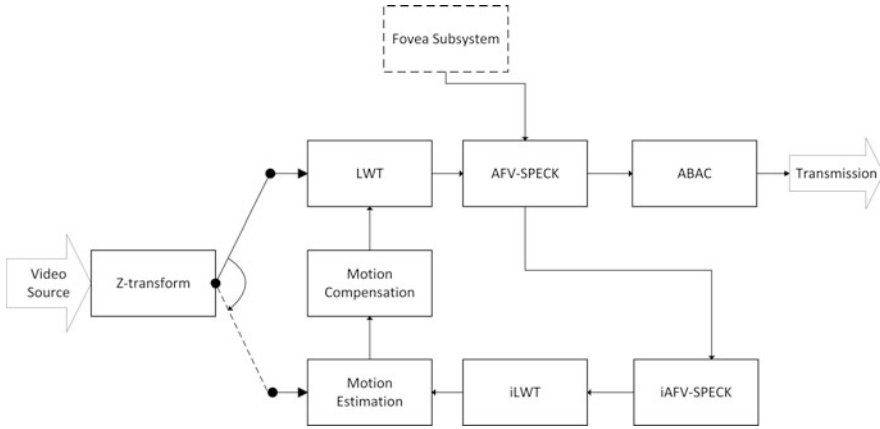


Fig. 19.9 Video coding framework AWFV-Codec [16]

model is updated. The adaptive part of the algorithm decreases its performance when compared against a static approach. However, the main advantage is that the input data is not preprocessed. As a result, the efficiency of the transmission of the compressed stream increases because there is no wait time for the calculation of the statistical model. There are several applications for ABAC as in JPEG and Joint Bi-level Image Group (JBIG)⁵ when dealing with black and white images. However, because SPECK encodes per bit, it makes ABAC suitable to compress the output of SPECK. In order to increase the computing time performance of the proposed framework, ABAC is included as its variable length encoder. Listing 19.1 shows the pseudocode for adaptive binary arithmetic coding.

Listing 19.1 ABAC algorithm

```

function ArithmeticCoding ( s )
    fq ← 1
    r ← 0
    l ← 0
    u ← 1
    for all s ∈ s do
        r ← r + 1
        if s = 0 then
            l' ← 0
            u' ← P/r
            fq ← fq + 1
        else
            l' ← fq/r
            u' ← 1
    
```

⁵<http://jpeg.org/jbig/index.html>.

```

        end if
        d = u - l
        u = l + d · u'
        l = l + d · l'
    end for
    return l
end function

```

In classic arithmetic coding, the interval used for arithmetic compression is $[0, 1)$. The function receives a string s to be compressed. The variable fq will store the frequency of the symbol 0. Because there are only two symbols on the alphabet, it is only needed to store one of the frequencies and compute the other by

$$P_1 = 1 - P_0 \quad (19.23)$$

where P_i is the probability of the symbol i . The probability of the symbol 0 is given by

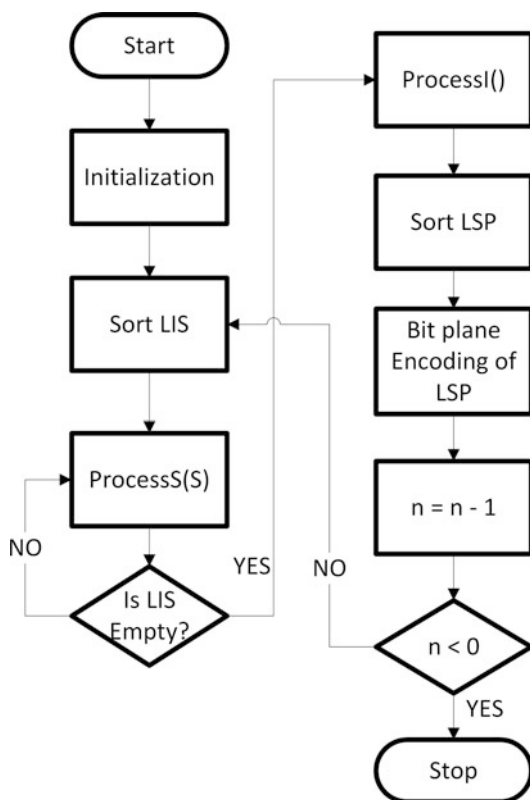
$$P_0 = \frac{fq}{r} \quad (19.24)$$

where r is the amount of read symbols. The algorithm stores the lower bound of the main interval on l and the upper bound on u . Each time a symbol is read, the counter r is increased on 1 and the interval for the input symbol is updated by using the frequency of the symbol 0 stored on fq . If a 0 is read, the frequency fq is increased in 1. After updating the statistical model, the new main interval is computed and the next symbol is read. The process stops when there are no more symbols to read on s and the statistical model P . P is a set that contains the probabilities of all different symbols s of the alphabet of s .

19.7.2 AFV-SPECK Algorithm

In the AFV-SPECK algorithm, every time a new coefficient is categorized as significant it will also be tested for its individual compression ratio using the cutoff window for each wavelet decomposition subband [16] (see Fig. 19.10). Note that the main loop remains the same as with SPECK. The input is the set of quantized coefficients, while the output is stored on S (assessed for significance by the function *ProcessS*), and the sorting of the LSP set is also added. If S is significant and only has one element (x, y) , the sign of quantized coefficient is stored on S and the set is removed from LIS. The function *ProcessI* evaluates I for its significance. As with FVHT, the computational complexity of AFV-SPECK will be expressed in terms of the *Big O* notation. The AFV-SPECK algorithm has a computational complexity of $\mathcal{O}(n)$. The analysis of the memory usage yielded that AFV-SPECK uses more memory when implemented as proposed in [31]. Further details can be found in [16].

Fig. 19.10 Flowchart of the main AFV-SPECK algorithm loop [16]



19.7.3 Simulation Results

To assess the reviewed video coding frameworks, SP-Codec and AWFV-Codec standard test images and video sequences were used⁶ [16]. For intra-frame coding, H.265 standard based on the IDCT using a 4×4 pixel block size is compared against SPECK and AFV-SPECK algorithms. Both binary streams were further compressed using the ABAC algorithm. The delta used for quantization was set to $\Delta = 40$, see e.g., [52]. Note that the chosen quantization delta and other parameters were used as input for SPECK and AFV-SPECK algorithms [16]. This is due to the fact that the compression ratio of the H.265 cannot be specified beforehand.

It is well-known that there is no analytic method to represent the exact perception of the HVS [56]. As a result, there are different metrics for image quality metrics [55]. In this work, the peak signal-to-noise ratio (PSNR) is used as performance metric [37]. The PSNR is defined in terms of the mean squared error (MSE) given by the equation

⁶<https://media.xiph.org/video/derf/>.

$$\text{MSE}(I, K) = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I_{i,j} - K_{i,j}]^2 \quad (19.25)$$

where m denotes the rows and n the columns of original image matrix, I is the matrix of the original image, and K represents the reconstructed image matrix. Using Eq. (19.25), the PSNR is given by

$$\text{PSNR}(I, K) = 10 \log_{10} \left[\frac{\text{MAX}_I^2}{\text{MSE}(I, K)} \right] \quad (19.26)$$

where MAX_I^2 is the square of the maximum value that a pixel of the image I can take. Such value depends on the amount of bits used per channel. Commonly, an image of 8 bits per channel has $\text{MAX}_I^2 = 255^2$. PSNR is measured in decibels (dB). Usually, it is considered that a reconstructed image with a PSNR of 40 dB or higher is of good quality for an average user [44]. However, trained users should require higher PSNR values. The 40 dB threshold is only a convention and has not been proved. Expected values of good reconstructions are between 20 dB and 50 dB [44].

As stated in the standard JPEG2000 and for a fair comparison, we use the biorthogonal CDF9/7 with four levels of decomposition [1]. Two metrics are used to assess the performance of the reported algorithms, namely PSNR and structural similarity index (SSIM) [54, 61]. This metric indicates that a reconstructed image with high quality should give a SSIM index closer to 1. Table 19.1 depicts comparisons in images for various video sequences using H.265, SPECK and AFV-SPECK algorithms, where CIF stands for common intermediate format. This table shows that the SPECK algorithm has a high PSNR (see e.g., [29]). It also observed that since the reported AFV-SPECK algorithm is based on ROIs and fovea centralis, it is expected that the result of these metrics to be equal or lower than the SPECK algorithm. Further details on these comparisons and other sequences are reported in [16].

Table 19.1 Comparisons between SPECK, AFV-SPECK and H.265 (see [16])

Name	BPP	H.265		SPECK		AFV-SPECK	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Lena gray 512	1.32	29.37	0.83	37.07	0.96	35.00	0.95
Lake	1.33	29.24	0.83	32.93	0.93	31.56	0.91
Peppers gray	1.27	29.31	0.81	34.04	0.92	32.77	0.90
Cameraman	1.28	28.93	0.75	40.07	0.97	34.59	0.94
Akiyo cif	1.19	28.96	0.78	35.42	0.94	33.12	0.92
Paris cif	1.28	28.79	0.76	30.45	0.85	29.53	0.83
Soccer cif	1.37	29.33	0.74	34.89	0.93	32.23	0.92

19.8 Conclusions

In this chapter, two wavelet-based algorithms were reviewed, namely FVHT and AFV-SPECK. Such algorithms exploit the HVS in order to increase the quality of the reconstructed image for an observer. The algorithms were assessed against classic compression algorithms such as the JPEG base algorithm and the algorithm used on the H.265 standard. Simple wavelet compression shows better performance when compressing images allowing to reach compression ratios of 0.06 bpp while retaining a good visual quality. The reported algorithms show similar behavior while increasing the quality of the compressed image over designed areas. However, when evaluated for overall quality, the reported algorithms show less performance than its non-fovea-based counterparts. This makes necessary an external subsystem that calculates the fixation point of the observers. Additionally, two wavelet-based video coding frameworks were surveyed, namely SP-Codec and AWFV-Codec [16]. The revised video frameworks increase the key frame reconstruction using wavelet-based compression that is also applied to motion compensation reconstruction. Fovea centralis coding also increases the quality of the reconstructed video as in AWFV-Codec, and in some cases, increases the quality of the reconstructed frames against non-fovea-based frameworks like SP-Codec. The reported AWFV-Codec is a viable choice for fast video streaming but it also reduces the utility of the stream when recorded. This is due to the fact that the video would be recorded without possibility of recovering the information discarded outside the fovea centralis. However, when stream recording is needed SP-Codec yields into better reconstruction quality than classic methods such as the H.265/HEVC video coding frameworks [15, 16]. The reported image compression algorithms FVHT and spatial transform AFV-SPECK require extra storage besides the wavelet coefficients. Methods will be investigated for in place computation for quantization in order to decrease the memory usage of both reported algorithms and for automatic foveation such as in [21].

Acknowledgement The authors gratefully acknowledge the financial support from CONACYT, Mexico.

References

1. Acharya, T., & Tsai, P. S. (2004). *JPEG2000 standard for image compression*. Hoboken, NJ: Wiley.
2. Alarcon-Aquino, V., & Barria, J. A. (2006). Multiresolution FIR neural-network-based learning algorithm applied to network traffic prediction. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 36(2), 208–220. <https://doi.org/10.1109/TSMCC.2004.843217>
3. Bocharova, I. (2010). *Compression for multimedia*. Cambridge: Cambridge University Press. <http://books.google.com/books?id=9UXBxPT5vuUC&pgis=1>
4. Böck, A. (2009). *Video compression systems: From first principles to concatenated codecs*. IET telecommunication series. Stevenage: Institution of Engineering and Technology. <http://books.google.com.mx/books?id=zJyOx08p421C>

5. Boopathi, G., & Arockiasamy, S. (2012). Image compression: Wavelet transform using radial basis function (RBF) neural network. In: *2012 Annual IEEE India Conference (INDICON)* (pp. 340–344). Piscataway: IEEE. <https://doi.org/10.1109/INDICON.2012.6420640>
6. Bovik, A. C. (2009). *The essential guide to video processing* (1st ed.). London: Academic Press.
7. Chang, E., Mallat, S., & Yap, C. (2000). Wavelet foveation. *Applied and Computational Harmonic Analysis*, 9(3), 312–335.
8. Cintra, R., Bayer, F., & Tablada, C. (2014). Low-complexity 8-point DCT approximations based on integer functions. *Signal Processing*, 99, 201–214. <https://doi.org/10.1016/j.sigpro.2013.12.027>. <http://www.sciencedirect.com/science/article/pii/S0165168413005161>
9. Ciocoiu, I. B. (2009). ECG signal compression using 2D wavelet foveation. In *Proceedings of the 2009 International Conference on Hybrid Information Technology - ICHIT '09* (Vol. 13, pp. 576–580)
10. Ciubotaru, B., Ghinea, G., & Muntean, G. M. (2014). Subjective assessment of region of interest-aware adaptive multimedia streaming quality. *IEEE Transactions on Broadcasting*, 60(1), 50–60. <https://doi.org/10.1109/TBC.2013.2290238>. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6755558>
11. Daubechies, I., & Sweldens, W. (1998). Factoring wavelet transforms into lifting steps. *The Journal of Fourier Analysis and Applications*, 4(3), 247–269. <https://doi.org/10.1007/BF02476026>. <http://link.springer.com/10.1007/BF02476026>
12. Dempsey, P. (2016). The teardown: HTC vive VR headset. *Engineering Technology*, 11(7–8), 80–81. <https://doi.org/10.1049/et.2016.0731>
13. Ding, J. J., Chen, H. H., & Wei, W. Y. (2013). Adaptive Golomb code for joint geometrically distributed data and its application in image coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(4), 661–670. <https://doi.org/10.1109/TCSVT.2012.2211952>. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6261530>
14. Frazier, M. (1999). *An introduction to wavelets through linear algebra*. Berlin: Springer. <http://books.google.com/books?id=IIRdY9nUTZgC&pgis=1>
15. Galan-Hernandez, J., Alarcon-Aquino, V., Ramirez-Cortes, J., & Starostenko, O. (2013). Region-of-interest coding based on fovea and hierarchical tress. *Information Technology and Control*, 42, 127–352. <http://dx.doi.org/10.5755/j01.itc.42.4.3076>. <http://www.itc.ktu.lt/index.php/ITC/article/view/3076>
16. Galan-Hernandez, J., Alarcon-Aquino, V., Starostenko, O., Ramirez-Cortes, J., & Gomez-Gil, P. (2018). Wavelet-based frame video coding algorithms using fovea and speck. *Engineering Applications of Artificial Intelligence*, 69, 127–136. <https://doi.org/10.1016/j.engappai.2017.12.008>. <http://www.sciencedirect.com/science/article/pii/S0952197617303032>
17. Gonzalez, R. C., & Woods, R. E. (2006). *Digital image processing* (3rd ed.). Upper Saddle River, NJ: Prentice-Hall.
18. Gray, R. M. (2011). *Entropy and information theory* (Google eBook). Berlin: Springer. <http://books.google.com/books?id=wdSOqgVbdRcC&pgis=1>
19. Hanzo, L., Cherriman, P. J., & Streit, J. (2007). *Video compression and communications*. Chichester, UK: Wiley.
20. Homann, J. P. (2008). *Digital color management: Principles and strategies for the standardized print production* (Google eBook). Berlin: Springer. <http://books.google.com/books?id=LatEFg5VBZ4C&pgis=1>
21. Itti, L. (2004). Automatic foveation for video compression using a neurobiological model of visual attention. *IEEE Transactions on Image Processing*, 13(10), 1304–1318. <http://dx.doi.org/10.1109/TIP.2004.834657>
22. Kondo, H., & Oishi, Y. (2000). Digital image compression using directional sub-block DCT. In *WCC 2000 - ICCT 2000. 2000 International Conference on Communication Technology Proceedings (Cat. No.00EX420)* (Vol. 1, pp. 985–992). Piscataway: IEEE. <http://dx.doi.org/10.1109/ICCT.2000.889357>. <http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=889357>

23. Lakhani, G. (2013). Modifying JPEG binary arithmetic codec for exploiting inter/intra-block and DCT coefficient sign redundancies. *IEEE transactions on Image Processing: A Publication of the IEEE Signal Processing Society*, 22(4), 1326–39. <http://dx.doi.org/10.1109/TIP.2012.2228492>. <http://www.ncbi.nlm.nih.gov/pubmed/23192556>
24. Lee, S., & Bovik, A. C. (2003). Fast algorithms for foveated video processing. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(2), 149–162. <http://dx.doi.org/10.1109/TCSVT.2002.808441>
25. Li, J. (2013). An improved wavelet image lossless compression algorithm. *International Journal for Light and Electron Optics*, 124(11), 1041–1044. <http://dx.doi.org/10.1109/10.1016/j.ijleo.2013.01.012>. <http://www.sciencedirect.com/science/article/pii/S0030402613001447>
26. Liu, L. (2008). On filter bank and transform design with the lifting scheme. Baltimore, MD: Johns Hopkins University. <http://books.google.com/books?id=f0IxpHYF0pAC&pgis=1>
27. Mallat, S. (2008). *A wavelet tour of signal processing, third edition: The sparse way* (3rd ed.). New York: Academic Press.
28. Miano, J. (1999). *Compressed image file formats: JPEG, PNG, GIF, XBM, BMP* (Vol. 757). Reading, MA: Addison-Wesley. http://books.google.com/books?id=_nJL.vY757dQC&pgis=1
29. Mohanty, B., & Mohanty, M. N. (2013). A novel speck algorithm for faster image compression. In *2013 International Conference on Machine Intelligence and Research Advancement* (pp. 479–482). <http://dx.doi.org/10.1109/ICMIRA.2013.101>
30. Ozenli, D. (2016). Dirac video codec and its performance analysis in different wavelet bases. In *24th Signal Processing and Communication Application Conference (SIU)* (pp. 1565–1568). <http://dx.doi.org/10.1109/SIU.2016.7496052>
31. Pearlman, W., Islam, A., Nagaraj, N., & Said, A. (2004) Efficient, low-complexity image coding with a set-partitioning embedded block coder. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(11), 1219–1235. <http://dx.doi.org/10.1109/TCSVT.2004.835150>
32. Peter, S., & Win, S. (2000). *Wavelets in the geosciences*. Lecture Notes in Earth Sciences (Vol. 90). Berlin: Springer. <http://dx.doi.org/10.1007/BFb0011093>. <http://www.springerlink.com/index/10.1007/BFb0011091>, <http://link.springer.com/10.1007/BFb0011091>
33. Poynton, C. (2012). *Digital video and HD: Algorithms and interfaces* (Google eBook). Amsterdam: Elsevier. <http://books.google.com/books?id=dSCEGFt47NkC&pgis=1>
34. Rao, K. R., Kim, D. N., & Hwang, J. J. (2011). *Fast Fourier transform—algorithms and applications: Algorithms and applications* (Google eBook). Berlin: Springer. <http://books.google.com/books?id=48rQQ8v2rKEC&pgis=1>
35. Rehna, V. (2012). Wavelet based image coding schemes: A recent survey. *International Journal on Soft Computing*, 3(3), 101–118. <http://dx.doi.org/10.5121/ijsc.2012.3308>. <http://www.airccse.org/journal/ijsc/papers/3312ijsc08.pdf>
36. Richardson, I. E. (2004). *H.264 and MPEG-4 video compression: Video coding for next-generation multimedia* (Google eBook). London: Wiley. <http://books.google.com/books?id=n9YVhx2zgZ4C&pgis=1>
37. Richardson, I. E. G. (2002). Video codec design. Chichester, UK: Wiley. <http://dx.doi.org/10.1002/0470847832>, <http://doi.wiley.com/10.1002/0470847832>
38. Rivas-Lopez, M., Sergiyenko, O., & Tyrsa, V. (2008). Machine vision: Approaches and limitations. In: Zhihui, X. (ed.) *Chapter 22: Computer vision*. Rijeka: IntechOpen. <https://doi.org/10.5772/6156>
39. Rivas-Lopez, M., Sergiyenko, O., Flores-Fuentes, W., & Rodriguez-Quinonez, J. C. (2019). *Optoelectronics in machine vision-based theories and applications* (Vol. 4018). Hershey, PA: IGI Global. ISBN: 978-1-5225-5751-7.
40. Ross, D., & Lenton, D. (2016). The graphic: Oculus rift. *Engineering Technology*, 11(1), 16–16. <http://dx.doi.org/10.1049/et.2016.0119>
41. Sacha, D., Zhang, L., Sedlmair, M., Lee, J. A., Peltonen, J., Weiskopf, D., et al. (2017). Visual interaction with dimensionality reduction: A structured literature analysis. *IEEE Transactions on Visualization and Computer Graphics*, 23(1), 241–250. <http://dx.doi.org/10.1109/TVCG.2016.2598495>

42. Said, A., & Pearlman, W. (1996). A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3), 243–250.
43. Salomon, D. (2006). *Coding for data and computer communications (Google eBook)*. Berlin: Springer. <http://books.google.com/books?id=Zr9bjEpXKnIC&pgis=1>
44. Salomon, D. (2006). *Data compression: The complete reference*. New York, NY: Springer.
45. Salomon, D., Bryant, D., & Motta, G. (2010). *Handbook of data compression (Google eBook)*. Berlin: Springer. <http://books.google.com/books?id=LHCY4VbiFqAC&pgis=1>
46. Sayood, K. (2012). *Introduction to data compression*. Amsterdam: Elsevier. <http://dx.doi.org/10.1016/B978-0-12-415796-5.00003-X>. <http://www.sciencedirect.com/science/article/pii/B978012415796500003X>
47. Schanda, J. (2007). *Colorimetry: Understanding the CIE system (Google eBook)*. London: Wiley. <http://books.google.com/books?id=uZadszSGe9MC&pgis=1>
48. Sergiyenko, O., & Rodriguez-Quinonez, J. C. (2017). *Developing and applying optoelectronics in machine vision* (Vol. 4018). Hershey, PA: IGI Global. ISBN: 978-1-5225-0632-4.
49. Silverstein, L. D. (2008). Foundations of vision. *Color Research & Application*, 21(2), 142–144.
50. Song, E. C., Cuff, P., & Poor, H. V. (2016). The likelihood encoder for lossy compression. *IEEE Transactions on Information Theory*, 62(4), 1836–1849. <http://dx.doi.org/10.1109/TIT.2016.2529657>
51. Stollnitz, E., DeRose, A., & Salesin, D. (1995). Wavelets for computer graphics: A primer. I. *IEEE Computer Graphics and Applications*, 15(3), 76–84. <http://dx.doi.org/10.1109/38.376616>. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=376616>
52. Sullivan, G. J., Ohm, J. R., Han, W. J., & Wiegand, T. (2012). Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12), 1649–1668. <http://dx.doi.org/10.1109/TCSVT.2012.2221191>
53. Sweldens, W. (1996). The lifting scheme: A custom-design construction of biorthogonal wavelets. *Applied and Computational Harmonic Analysis*, 3(2), 186–200. <http://dx.doi.org/10.1006/acha.1996.0015>. <http://www.sciencedirect.com/science/article/pii/S1063520396900159>
54. Tan, T. K., Weerakkody, R., Mrak, M., Ramzan, N., Baroncini, V., Ohm, J. R., et al. (2016). Video quality evaluation methodology and verification testing of HEVC compression performance. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(1), 76–90. <http://dx.doi.org/10.1109/TCSVT.2015.2477916>
55. Tanchenko, A. (2014). Visual-PSNR measure of image quality. *Journal of Visual Communication and Image Representation*, 25(5), 874–878. <http://dx.doi.org/10.1016/j.jvcir.2014.01.008>. <http://www.sciencedirect.com/science/article/pii/S1047320314000091>
56. Theodoridis, S. (2013). *Academic press library in signal processing: Image, video processing and analysis, hardware, audio, acoustic and speech processing (Google eBook)*. London: Academic Press. <http://books.google.com/books?id=QJ3HqmLG8glC&pgis=1>
57. Viction Workshop L. (2011). *Vectorism: Vector graphics today*. Victionary. <http://books.google.com/books?id=dHaeZwEACAAJ&pgis=1>
58. Wallace, G. (1992). The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1), xviii–xxxiv. <http://dx.doi.org/10.1109/30.125072>. <http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=125072>
59. Wallace, G. K. (1991). The JPEG still picture compression standard. *Communications of the ACM*, 34(4), 30–44. <http://dx.doi.org/10.1145/103085.103089>. <http://dl.acm.org/citation.cfm?id=103085.103089>
60. Walls, F. G., & MacInnis, A. S. (2016). VESA display stream compression for television and cinema applications. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 6(4), 460–470. <http://dx.doi.org/10.1109/JETCAS.2016.2602009>
61. Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4), 600–612. <http://dx.doi.org/10.1109/TIP.2003.819861>

62. Werner, J. S., & Backhaus, W. G. K. (1998). *Color vision: Perspectives from different disciplines*. New York, NY: Walter de Gruyter. <http://books.google.com/books?id=gN0UaSUTbnUC&pgis=1>
63. Wien, M. (2015). *High efficiency video coding— coding tools and specification*. Berlin: Springer.
64. Zhang, L., Wang, D. &, Zheng, D. (2012). Segmentation of source symbols for adaptive arithmetic coding. *IEEE Transactions on Broadcasting*, 58(2), 228–235. <http://dx.doi.org/10.1109/TBC.2012.2186728>. <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6166502>