Universidad Miguel Hernández

Departamento de Ingeniería
de Sistemas Industriales

Fast and Efficient Coding Tools for Digital Image and Video Signals

Tesis Doctoral

Memoria presentada para optar
al grado de Doctor por:
Otoniel Mario López Granado

dirigida por:
Manuel Pérez Malumbres
José S. Oliver Gil

Universidad Miguel Hernández

Departamento de Ingeniería
de Sistemas Industriales

Fast and Efficient Coding Tools for Digital Image and Video Signals

PhD thesis

A dissertation for the degree of
Doctor of Philosophy by:
Otoniel Mario López Granado

Advisors:
Manuel Pérez Malumbres
José S. Oliver Gil

D. MANUEL PÉREZ MALUMBRES, Catedrático de Universidad de la Universidad Miguel Hernández y D. JOSÉ SALVADOR OLIVER GIL, Profesor Titular de Universidad de la Universidad Politécnica de Valencia,

CERTIFICAN:

Que la presente memoria *Fast and Efficient Coding Tools for Digital Image and Video Signals*, ha sido realizada bajo su dirección, en el Departamento de Ingeniería de Sistemas Industriales de la Universidad Miguel Hernández, por el Ingeniero D. OTONIEL MARIO LÓPEZ GRANADO, y constituye su tesis para optar al grado de Doctor.

Para que conste, en cumplimiento de la legislación vigente, autorizan la presentación de la referida tesis doctoral ante la Comisión de Doctorado de la Universidad Miguel Hernández, firmando el presente certificado.

Elche, 2 de Marzo de 2010

Fdo. Manuel Pérez Malumbres

José Salvador Oliver Gil

D. OSCAR REINOSO GARCÍA, Profesor Titular de Universidad y director del Departamento de Ingeniería de Sistemas Industriales de la Universidad Miguel Hernández,

CERTIFICAN:

Que la presente memoria *Fast and Efficient Coding Tools for Digital Image and Video Signals*, realizada bajo la dirección de D. MANUEL PÉREZ MALUMBRES y D. JOSÉ SALVADOR OLIVER GIL, en el Departamento de Ingeniería de Sistemas Industriales de la Universidad Miguel Hernández, por el Ingeniero D. OTONIEL MARIO LÓPEZ GRANADO, constituye su tesis para optar al grado de Doctor.

Para que conste, en cumplimiento de la legislación vigente, autoriza la presentación de la referida tesis doctoral ante la Comisión de Doctorado de la Universidad Miguel Hernández, firmando el presente certificado.

Elche, 2 de Marzo de 2010

Fdo. Oscar Reinoso García

Acknowledgements

A mi esposa Maria José,
a la memoria de mi padre Mario,
a mi madre Rosa y a mis hermanos Nemuel y Jemimah.

Quiero dedicarles este trabajo, por el apoyo y cariño que siempre me han dado y por la confianza que han depositado en mí. En especial quiero agradecer a mi difunto padre sus palabras y enseñanzas que durante aquellos largos paseos nos inculcaba a mis hermanos y a mi, y que en gran medida han hecho que llegue hasta aquí. Únicamente deseo no haberos defraudado en ningún momento.

Measure not the work until the day's out
and the labor done.
(Elisabeth Barrett Browning)

Hace ya más de seis años que comenzó mi andadura en el ámbito universitario como docente y unos cuatro años que inicié mi trayectoria investigadora. Es por todo sabidos, que la realización de una tesis doctoral implica muchos sacrificios personales ya que ésta conlleva un intenso trabajo y dedicación de horas. Lo que no se suele advertir es que ese sacrificio también lo sufren muchos de nuestros seres queridos, como la familia y los amigos, pues dejas de compartir muchos momentos con ellos. Quiero agrader a mis amigos su comprension por no poderles dedicar todo el tiempo que desearía y que sin embargo siempre han estado ahí cuando los he necesitado. En especial, Cristina, Qk, Alvaro y Rosario, Manoli y Carlos, Julian y Mª José, Josep Enric, Manolo y Loli (mis socios), Vicente y Mavi, Pablo y Natalia y algún otro que seguro me dejo en el tintero, quiero que sepais que siempre os llevo en mis pensamientos.

Quiero también agradecer a mis directores de tesis y amigos, Manuel P. Malumbres y José Oliver, su paciencia e intensa implicación en la consecución de este trabajo. Sinceramente Mels, creo que tienes una paciencia infinita. A tí Jose quiero también agradecerte los buenos momentos que hemos pasado en Colchester, y que han hecho más llevadera mi estancia allí. Gracias a vuestros consejos y conocimientos, he podido llegar a este momento.

Agradecer también a Mohammed Ghanbari y al "Video networking Laboratory" de la Universidad de Essex, la posibilidad de realizar una estancia de investigación en sus instalaciones.

Quisiera también agradecer a mis compañeros y amigos del Departamento de Física y Arquitectura de Computadores, Miguel, Pablo, Hector, Vicente, Manuel, Jesus y Adrián los buenos momentos que pasamos juntos, como bien dice Vicente al estilo 'Camera Café' y que gracias a ellos hacen que la estancia en la universidad sea mucho más agradable.

En general quiero agradecer a todos los que de una forma u otra habéis hecho posible que este trabajo llegara a buen término.

Mis mayores agradecimientos son para las personas a las que les he dedicado el trabajo. A mis padres, Mario y Rosa, quiero agradecerles el esfuerzo que tanto economica como personalmente hicieron para poder darme unos estudios y un futuro profesional. Durante toda vuestra vida habeis antepuesto

vuestros hijos a todo lo demás. A mi hermana Jemimah y mi hermano Nemuel, que siempre han estado ahí cuando los he necesitado, quiero daros las gracias porque sois bondadosos y desinteresados y siempre me habeis ayudado sin esperar nada a cambio.

Finalmente, cómo expresar con palabras todo lo que tú haces por mí, Maria José. Desde que nos conocimos siempre has sido la persona que ha aportado serenidad, bondad y dulzura a mi vida. Tú me animaste a que emprendiera este nuevo camino en la docencia y la investigación. Principalmente tú has podido vivir dia a dia el esfuerzo personal que hemos realizado para la consecución de este trabajo, y que sólo gracias a tu comprensión y ánimo he podido sobrellevar. Jamás encontraré suficientes palabras para agradecer el que un día aparecieses en mi vida y lo único que deseo es que nunca faltes de ella.

A todos, Gracias.

IV

Abstract

Image and video compression is of utmost importance in multimedia systems and applications because it drastically reduces bandwidth requirements for transmission and memory requirements for storage. Great efforts have been made to improve coding efficiency of wavelet-based image encoders, achieving in this way a reduction in the bandwidth or amount of memory needed to transmit or store a compressed image. Unfortunately, many of these coding optimizations involve higher complexity, requiring faster and more expensive processors. For example, the JPEG 2000 standard uses a large number of contexts and an iterative time-consuming optimization algorithm called Post-Compression Rate Distortion (PCRD) to improve coding efficiency. Other encoders like Embedded Conditional Entropy Coding of Wavelet Coefficients (ECECOW) achieve very good coding efficiency with the introduction of high-order context modeling, being the model formation a really slow process. Even bit-plane coding employed in many encoders like Set Partitioning In Hierarchical Trees (SPIHT) or Subband-Block Hierarchical Partitioning (SBHP) results in a slow coding process since an image is scanned several times, focusing on a different bit-plane in each pass, which in addition causes a high cache miss rate. The aforementioned encoders are designed to obtain the maximum performance in rate-distortion terms, but unfortunately other design parameters like complexity or memory resources are not considered as critical as the former ones.

Recently, there has been increasing interest in the design of very fast wavelet image encoders focused on applications (interactive real-time image & video applications, Geographic Information System (GIS), etc.) and devices (digital cameras, mobile phones, Personal Digital Assistant (PDA), etc) where coding delay and/or available computing resources (working memory and power processing) are critical for proper operation. In that scenario, the data must be encoded as soon as possible to fit the application time restrictions using the scarce available resources in the system (memory and processing power). Basically, these encoders do not present any type of iterative method and each coefficient is encoded as soon as it is visited. This

process results in the loss of Signal to Noise Ratio (SNR) scalability and precise rate control capabilities. They simply apply a constant quantization to all the wavelet coefficients, encoding the image at a constant and uniform quality, as it happened in the former JPEG standard, where only a quality parameter was available (and no rate control was performed).

In this thesis, we propose several rate control algorithms for non-embedded encoders. These algorithms will predict the proper quantization values that lead to a final bit rate close to the target one. In particular, we propose several bit rate prediction methods with increasing complexity and accuracy. The proposed algorithms do not introduce great overhead in the coding process. Also, in this thesis, a sign coding stage is proposed in order to improve the coding efficiency of the non-embedded encoders.

Regarding video coding, a wide variety of video compression schemes have been reported in the literature. Most of them are based on the Discrete Cosine Transform (DCT) and motion estimation/compensation techniques. However, a lot of research interest was focused on developing video wavelet coders due to the great properties of wavelet transform. Most wavelet-based video encoding proposals are strongly based on inter-coding approaches, which require high-complexity encoder designs as counterpart to the excellent R/D performance benefits. Even so, some applications like professional video editing, digital cinema, video surveillance applications, multispectral satellite imaging, High Quality (HQ) video delivery, etc. would rather use an intra-coding system that is able to reconstruct a specific frame of a video sequence as soon as possible and with high visual quality. So, the strength of an intra video coding system relies on the ability to efficiently exploit the spatial redundancy of each video sequence frame avoiding complexity in the design of the encoding/decoding engines.

In this thesis, we propose a new lightweight and efficient intra video coder, called Motion Lower Tree Wavelet (M-LTW), based on the Lower Tree Wavelet (LTW) algorithm with accurate rate control capability and scene changes detection. This intra video encoder is able to encode an International Telecommunication Union (ITU) D1 size sequence in real time with good quality.

Finally, in this thesis, we will introduce a 3D-Wavelet based video encoder in order to improve the coding efficiency (more compression rate at the same quality than the INTRA video encoder), by exploiting the intrinsic video temporal redundancy.

Resumen

La compresion de imagen y vídeo es de suma importancia en los sistemas y aplicaciones multimedia porque reducen las necesidades de ancho de banda para su transmision y las necesidades de memoria para su almacenamiento. Se han realizado grandes esfuerzos para mejorar la eficiencia en la codificacion de los codificadores de imagen basados en la transformada wavelet, consiguiendo de esta forma una reducción tanto en el ancho de banda como en la memoria requerida para transmitir o almacenar una imagen comprimida. Desafortunadamente, muchas de estas optimizaciones implican un aumento en la complejidad de la codificación, siendo necesarios procesadores más rápidos y por consiguiente más caros. Por ejemplo, el actual estandar de codificación de imagen, JPEG2000, usa un gran numero de contextos y un algoritmo iterativo de optimización con un gran coste computacional llamado Post-Compression Rate Distortion (PCRD) con el fin de mejorar la eficiencia en la codificación. Otros codificadores como Embedded Conditional Entropy Coding of Wavelet Coefficients (ECECOW), consiguen una gran eficiencia en la codificación mediante la inclusión de un modelado basado en contextos de alto orden, siendo la formación del modelo un proceso muy lento. Otros codificadores, como Set Partitioning In Hierarchical Trees (SPIHT) o Subband-Block Hierarchical Partitioning (SBHP), realizan un procesamiento por planos de bits, lo que implica un proceso de codificación lento pues la imagen se procesa varias veces, centrándose cada vez en un plano de bit, lo que además produce muchos fallos de memoria caché. Estos codificadores estan diseñados para obtener el máximo rendimiento en terminos de calidad. Los codificadores mencionados anteriormente son escalables en calidad y son empotrados por naturaleza, es decir, son capaces de ajustarse a la tasa de bits deseada, dando la mejor calidad posible para dicha tasa de bits. Sin embargo, en estos codificadores no se han tenido en cuenta otros parametros de diseño como puedan ser los requisitos de memoria o la complejidad computacional.

En los ultimos años ha aumentado el interés por diseñar codificadores de imagen rápidos basados en la transformada wavelet para aplicaciones (ima-

gen y vídeo en tiempo real, sistemas de información geografica (GIS), etc) y dispositivos (cámaras digitales, telefonos moviles, Personal Digital Assistant (PDA), etc) donde los tiempos de codificación o los recursos disponibles son críticos. En un escenario como este, los datos deben codificarse tan pronto como sea posible para ajustarse a las restricciones de tiempo de la aplicación en cuestión y a los escasos recursos disponibles en el sistema. Básicamente, estos codificadores no presentan ningun tipo de proceso iterativo y cada coeficiente se codifica tan pronto como es visitado. Como contrapartida, esto implica la pérdida de escalabilidad en calidad Signal to Noise Ratio (SNR) y la posibilidad de ajustarse a una tasa de compresión en concreto. En estos codificadores únicamente se aplica una cuantización constante a todos los coeficientes wavelet, codificando de esta manera la imagen con una calidad uniforme, tal y como ocurría en el anterior estandar de compresión de imagen JPEG, donde sólo habia un único parametro de cuantización y no existía la posibilidad de ajustarse a una determinada tasa de compresión.

En esta tesis, se proponen varios algoritmos de control de tasa de bits especialmente diseñados para codificadores de imagen no empotrados basados en la transformada wavelet. Estos algoritmos determinarán los valores de los parámetros de cuantización para obtener la tasa de bits deseada. En concreto, se proponen varios metodos de ajuste de tasa de compresión con una complejidad y precisión incremental. Los algoritmos propuestos no introducen una gran complejidad al proceso de codificación. Además, en esta tesis se propone una nueva aproximación a la compactación del signo de los coeficientes con el fin de mejorar la eficiencia de los compresores no empotrados.

En cuanto a codificación de vídeo se refiere, se han propuesto numerosos esquemas de compresión de vídeo, muchos de ellos basados en la transformada discreta del coseno (DCT) y en técnicas de estimación y compensación de movimiento. Si embargo, en los últimos años, se ha puesto especial énfasis en desarrollar codificadores de vídeo basados en la transformada wavelet. Muchos de estos codificadores están basados en una codificación de tipo "inter", lo que requiere unos diseños de codificación más complejos, aunque por contrapartida obtienen muy buenos resultados en cuanto a R/D se refiere. Sin embargo, algunas aplicaciones tales como la edición de vídeo profesional, el cine digital, las aplicaciones de vídeo vigilancia, las imágenes de satélite multiespectrales, etc, usan una codificación de tipo "intra", de manera que un frame es reconstruido tan pronto como es posible y con una buena calidad visual. De esta forma, la parte fundamental de los sistemas de codificación de vídeo de tipo "intra" radica en la capacidad de explotar de forma eficiente la redundancia espacial de cada frame de una secuencia de vídeo, evitando diseños complejos de codificadores.

En esta tesis, se propone un nuevo codificador de vídeo intra muy rápido, llamado M-LTW (Motion Lower Tree wavelet), que está basado en el codificador de imagen no empotrado LTW (Lower Tree Wavelet), al cual se le ha incluido un algoritmo de control de tasa de bits, también propuesto en esta tesis, que además es capaz de detectar cambios bruscos de escena.

Finalmente, en esta tesis, introduciremos un codificador basado en la transformada wavelet 3D con el fin de mejorar la compresión con respecto al codificador de tipo INTRA, explotando la redundancia temporal inherente a las secuencias de video.

Preface

Motivation

During the last decade, several image compression schemes have emerged in order to overcome the known limitations of block-based algorithms that use the Discrete Cosine Transform (DCT). These limitations include blocking artifacts and poor coding efficiency, mainly at moderate to low bit rates. Some of these alternative proposals were based on more complex techniques, like vector quantization and fractal image coding, whereas others successfully proposed the use of a different and more suitable mathematical transform, the Discrete Wavelet Transform (DWT). Thus, while the popular JPEG standard for image compression uses the DCT, the new JPEG 2000 standard proposes the use of the wavelet transform, with better rate/distortion (R/D) performance, and avoiding blocking artifacts because an image is not separately transformed and quantized block-by-block, but the wavelet transform is applied to the entire image.

Unfortunately, despite the benefits that wavelet-based image coding involves, other problems are introduced in these encoders; basically they are typically implemented with memory intensive and time-consuming algorithms and thereby system requirements are significantly higher than in other earlier image encoders like JPEG. These higher requirements represent a serious limitation when implementing multimedia applications like image compression in memory-constrained devices with relatively little computational power, such as digital cameras, mobile phones, PDAs and embedded devices. Presently, in many applications like videoconferencing (implemented with image coding if only intraframe redundancy is removed) features like low complexity and high symmetry are more important than R/D performance.

All the wavelet image coders, and in general all the transform-based encoders, consist of two main stages. During the first one, an image is transformed from a spatial domain to another domain, and in the case of wavelet transform a combined spatial-frequency domain called wavelet domain. In the second pass, the wavelet coefficients resulting from the transform domain

are quantized and encoded in an efficient way to achieve high compression efficiency and other features.

Great efforts have been made in the coding stage to improve compression efficiency, achieving in this way a reduction in the bandwidth or amount of memory needed to transmit or store a compressed image. Unhappily, many of these coding optimizations involve high complexity, requiring faster and more expensive processors. For example, the JPEG 2000 standard uses a large number of contexts and an iterative time-consuming optimization algorithm to improve coding efficiency. Furthermore, many times, wavelet image coders have features that are not always needed, but which make them CPU and memory intensive. For example, bit-plane coding employed in many encoders (like EZW and SPIHT) allows SNR scalability with an embedded bitstream, but it results in slow coding since an image is scanned several times, focusing on a different bit-plane in each pass, which in addition causes a high cache miss rate.

Recently, there has been increasing interest in the design of very fast wavelet image encoders focused on applications and devices where coding delay and/or available computing resources are critical. Essentially, these encoders do not present any type of iterative method and each coefficient is encoded as soon as it is visited. This process results in the loss of SNR scalability and precise rate control capabilities. They simply apply a constant quantization to all the wavelet coefficients, encoding the image at a constant and uniform quality, as it happened in the former JPEG standard.

In this thesis, we tackle the problem of rate control capability loss in these fast wavelet image encoders, designing fast rate control tools that compensate the embedded bitstream loss, but preserving the coding efficiency and the speed of the coding process. Also, in this thesis we add a sign coding stage to these fast non-embedded encoders in order to improve the coding efficiency.

Objectives

The specific objectives of this thesis can be detailed as follows:

- Develop fast and accurate rate control algorithms for non-embedded image encoders.
- Study the state-of-the-art sign coding techniques in order to design a new sign coding stage suitable for non-embedded image encoders.
- Develop a new fast and low-memory demanding non-embedded intra video encoder, based on wavelet transform.

  - Design an accurate rate control algorithm for intra video coding which includes scene change detection.
  - Develop a fast, low memory demanding intra video encoder with fixed point arithmetic.

- Develop a fast coding stage for a low memory demanding video encoder based on 3D wavelet transform.

Thesis overview

This thesis is organized in six chapters, which are introduced here:

Chapter 1 presents the fundamentals of image and video compression. In Section 1.1.2, we present several state-of-the-art wavelet image encoders, emphasizing non-embedded encoders. The present image coding standard JPEG 2000 is also presented in Section 1.1.3. On the other hand, in Section 1.2.2 and 1.2.3, both the video coding standard H.264 (also referred to as MPEG-4, part-10) and other wavelet video encoders are presented.

In Chapter 2 we present several rate control algorithms specially suited for non-embedded image encoders with increasing complexity and accuracy. These algorithms will predict the proper quantization values that lead to a final bit rate close to the target one. In order to evaluate the proposed rate control methods, we have selected the LTW encoder, which will briefly be described in Section 1.1.2.

In Chapter 3 an in-depth sign coding study is presented. Also, a practical solution for sign coding in the non-embedded LTW encoder is presented.

Chapter 4 introduces a new enhanced non-embedded codec called E-LTW, which includes a sign coding stage and an accurate rate control algorithm.

In Chapter 5 an intra video encoder (M-LTW) is presented. This new intra video codec introduces a modified version of the Model-based image rate control algorithm presented in Section 2.2, taking into account possible scene changes, camera panning or camera zooming.

An overview of 3D wavelet video coding is presented in Chapter 6. In Section 6.3 a recursive implementation of the frame-by-frame 3D wavelet transform is presented. Also, in Section 6.4 a fast coding stage for this frame-by-frame 3D-DWT scheme is presented.

Finally, Chapter 7 concludes the thesis and discusses future research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

1.1 Image Coding

Compression of digital images plays a key role in image storage and transmission. In this chapter a brief introduction to general image compression will be given.

1.1.1 Fundamentals

The usefulness of digital images in information transmission is not questionable, but the cost of storing and transmitting images is much larger compared to storage and transmission of text, so that for example image databases require more storage than document archives.

The amount of data transmitted via the Internet doubles every year, and a large portion of that data are images and video sequences. Reducing the bandwidth needs of any given device will result in significant cost reductions and will make the device more affordable. Magnetic hard discs (HD)s, CDs, DVDs, and Solid State Drives (SSD)s of larger capacity are released every year, in response to greater demand for storage of digital data. Image compression offers ways to represent an image in a more compact way, so that one can store more images and transmit images faster. The advantages of image compression come at the expense of a computational cost. Before storing or transmitting an image it is processed in such a way that will require fewer bits to represent it.

A compression algorithm tries to offer the best trade-off between the bandwidth, memory, computation factors and quality for a given application. For example, if we are limited in terms of memory we can spend more computational time to compress the image and make sure it fits into the given memory size. If we are computation limited we can store the image as it is with no compression or with limited compression with a simple compression algorithm.

Image compression algorithms have been the subject of research both in academia and industry for many years, but there is still room for new technologies. The first widely adopted international image compression standard was JPEG [37, 76] which was introduced in the late eighties. JPEG is based on DCT followed by entropy coding based on either Huffman coding [36, 87, 21] or binary arithmetic coding [53, 83, 21, 111]. It has been widely used from the printing industry to Internet applications. For example, all high-end printers compress the image to be printed before they actually send it to the print engine, and most images transmitted via the internet are JPEG compressed. JPEG is intended for continuous tone images of more than one bit depth. Algorithms for binary images work in a different way,

and JBIG-1 and JBIG-2 are the standards covering this area. There are other standards, such as facsimile transmission standards [43], the FlashPix file format [32], the TIFF file format [5], and page description languages like Portable Document Format (PDF).

There are two major classes of image compression algorithms, namely lossy and lossless algorithms. Lossless algorithms preserve the image data, i.e. original and reconstructed images are exactly the same. In lossy image compression, original and reconstructed images may or may not be identical in a strict mathematical sense, but to a human observer they may look the same, so the goal is to achieve compression that is visually lossless. Both lossy and lossless compression algorithms are used today in a broad range of applications, from transmitting satellite images, to web browsing to image printing and scanning. With lossy compression algorithms we can achieve significantly larger compression ratios compared to lossless algorithms.

Generic compression system

Most image coders consist of transform, quantization and entropy coding, as seen in Figure 1.1. The transform block is in general a reversible operation, i.e. a cascade of forward and inverse transform block is the identity operation. $T.T^{-1}(arg) = T^{-1}.T(arg) = arg$. Quantization, on the other hand, introduces some loss. The quantizer usually maps an interval of real numbers to a single index, constituting the only lossy part of the coding system i.e., $Q^{-1}.Q(arg) \neq arg$. It is lossy because the knowledge of an index is only enough to give us the corresponding interval in the real line but not the exact number in the real line. The entropy coder is the building block responsible for compression, it maps more frequent indexes to small codewords and less frequent indexes to larger codewords. It is also a reversible operation. A large portion of the computational complexity of a compression system is due to the entropy coding part of the system. More compression usually translates to higher computational complexity. In general, arithmetic [53] and Huffman coding [36] are the most common choices. Arithmetic coding is intended for high-end applications where complexity is not a concern, but compression performance is, while Huffman coding is intended for low-end applications where simplicity is more important. Typically the most memory intensive element is the transform. Quantization, on the other hand, is a much simpler process than the transform or the entropy coder.

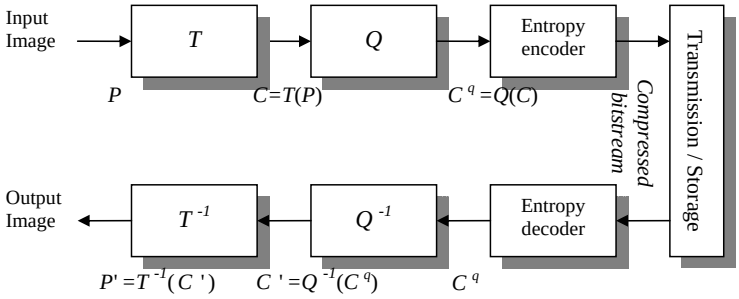


Figure 1.1: Overview of an image coder and decoder based on transform coding. T and T^{-1} are the forward and inverse transform functions, respectively. Q and Q^{-1} are the quantizer and dequantizer functions, respectively. The original set of pixels is represented by P

1.1.2 Wavelet based encoders

The wavelet transform is able to spatially decorrelate the image pixels in a linear way. However, more complex dependencies exist in natural images. Therefore, we still need good processing techniques, beyond simple entropy coding, in order to reduce these high-order statistical dependencies and so improve compression efficiency. The way in which wavelet coefficients are encoded establishes the coding model and it is the main difference among different encoders. In this section, we survey some of the most important wavelet-based image encoders that have been reported in the literature. In the performance analysis of each proposal, we not only focus on their coding efficiency but also on their complexity, since reduced complexity is one of the objectives of this thesis.

Overview

The wavelet transform computation represents only the first step in transform coding, and it is employed to decorrelate the input samples (pixels in the case of image coding), achieving a less redundant smaller area of coefficients, which concentrates most energy, whereas the remaining coefficients are reduced and, in many cases, become zero or very close to zero. Therefore, the Discrete Wavelet Transform (DWT) is a common point in wavelet coding, and there is almost no difference in this part from one wavelet-based encoder to another one. In this step, almost the only degree of freedom for an encoder is the wavelet family and the type of wavelet decomposition. Although most schemes are based on the B9/7 transform [7] with a dyadic decomposition, other wavelet families and wavelet decompositions (such as wavelet packets

[20, 81]) have been employed [116, 63, 100, 80].

Following the scheme depicted in Figure 1.1, after the DWT computation an encoder must define the way in which rate/distortion is modified through quantization, and how to encode the quantized coefficients. The way quantization and coding is applied defines a specific model for each wavelet encoder, and it is probably the main difference among different encoders.

Some wavelet encoders apply in combination the quantization and entropy coding steps, so as to improve coding performance by means of optimization algorithms (such as the Lagrange multiplier method [96, 81]), or to allow other features, like SNR scalability (e.g., by applying quantization through successive approximation [93, 84]). Actually, the model employed not only establishes the compression performance but also other additional features of the output bit-stream. E.g., generally speaking, depending on the order in which coefficients are encoded, an image can be decoded with resolution or quality scalability.

A wide variety of wavelet-based image compression schemes have been reported in the literature, ranging from simple entropy coding to more complex techniques such as vector quantization [98, 66], tree-based coding [93, 84], block-based coding [104, 75], edge-based coding [56], joint space-frequency quantization schemes [115, 116], trellis coding [44], etc.

The early wavelet-based image coders [113, 7] were designed in order to exploit the ability of the wavelet transform to compact the energy of an image in a simple way. They employed scalar or vector quantizers and variable-length entropy coding, showing little improvement with respect to popular DCT-based algorithms like JPEG. In fact, in [34], some early wavelet encoders were compared with JPEG, concluding that these encoders obtained better results than JPEG only when very low bit rates were used (below 0.25 bits per pixel (bpp) for an original grey-scale 8 bpp image). However, despite a not very brilliant beginning, DWT has been successfully employed later in the field of image coding.

In this chapter, some of the most relevant and efficient wavelet tree-based coding techniques that have been proposed recently are surveyed. Among the wide variety of efficient encoders available in the literature, we highlight the non-embedded proposals and the fastest coding/decoding schemes. The reason why we focus on this type of encoder is that we are interested in models with low computational requirements.

Embedded zero-tree wavelet (EZW) coding

In the early 90s, there was the general idea that more efficient image coding would only be achieved by means of sophisticated techniques with high

complexity. The embedded zero-tree wavelet encoder (Embedded Zero-tree Wavelet (EZW)) [93] can be considered the first wavelet image coder that broke that trend. This encoder exploits the properties of the wavelet coefficients more efficiently than the rest of early techniques and thereby, it considerably outperforms their coding performance.

The EZW algorithm is mainly based on two basic ideas: (a) the similarity between the same type of wavelet subband, with higher energy as the subband level increases, and (b) a type of quantization based on a successive-approximation scheme that can be adjusted in order to get a specific bit rate in an embedded way. The former idea is exploited by means of coefficient trees, whereas the latter is usually implemented with bit-plane coding. In addition, the encoder includes an adaptive arithmetic encoder to encode the generated symbols. Although the EZW technique never became a standard, it is of great historical importance in the field of wavelet-based image coding because the aforementioned two principles were later used and refined by many other coding methods.

Let us define the coefficient trees employed in EZW. In a dyadic wavelet decomposition there are coefficients from different subbands representing the same spatial location in the sense that one coefficient in a scale corresponds spatially with four coefficients in the correspondent previous subband. This connection can be extended recursively with these four coefficients and the corresponding direct descendants (sometimes called offspring) at the previous levels, so that coefficient trees can be defined as shown in Figure 1.2. Since each node in a tree has four direct descendants (except the coefficients at the first level, corresponding with the leaf nodes), this type of tree is sometimes called quadtree. Note that a quadtree (or subquadtree) can be built from each coefficient by considering it as the root node of a tree.

The key idea employed by EZW to perform tree-based coding is that, in natural images, most energy tends to concentrate at coarser scales (i.e., higher decomposition levels). Then, it can be expected that the closer to the root node a coefficient is, the larger magnitude it has. Therefore, if a node of a coefficient tree is lower than a threshold, its descendant coefficients are likely to be lower as well. In other words, the probability for all four children to be lower than a threshold is much higher if the parent is also lower than that threshold. We can take advantage of this fact by coding the subband coefficients by means of trees and successive approximation, so that when a node and all its descendant coefficients are lower than a threshold, just a symbol is used to encode that entire branch.

The EZW algorithm is performed in several steps, with two stages per step: the dominant pass and the subordinate pass. Successive-approximation can be implemented as a bit-plane encoder so that the method can be outlined

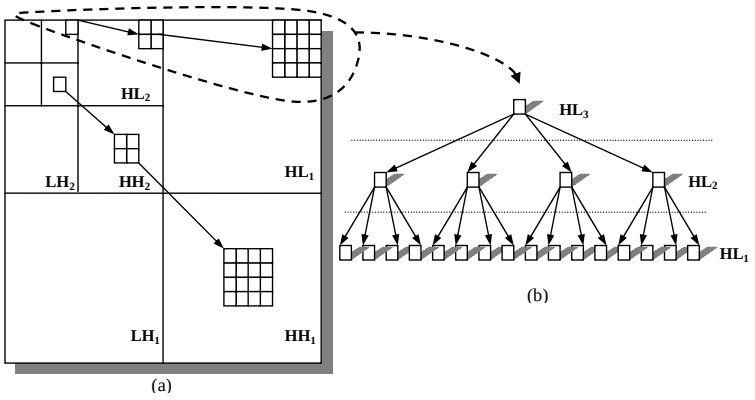


Figure 1.2: Definition of wavelet coefficient trees. In (a), it is shown that coefficients of the same type of subband (HL, LH or HH) representing the same image area through different levels can be logically arranged as a quadtree, in which each node is a wavelet coefficient. The parent/child relation between each pair of nodes in the quadtree is presented in (b)

as follows: Consider that we need n bits to represent the highest coefficient of the image (in absolute value). Then, the first step will be focused on all those coefficients that need exactly n bits to be coded (ranging from 2^{n-1} to 2^n-1), which are considered to be significant with respect to n. In the dominant pass, each coefficient falling in this range (in absolute value) is labeled and encoded as significant positive/negative (sp/sn), depending on its sign. These coefficients will no longer be processed in further dominant passes, but in subordinate passes. On the other hand, the remaining coefficients (those in the range $[0, 2^{n-1}[$) are encoded as zero-tree root (zr) if all its descendants also belong to this range, or as isolated zero (iz) if any descendant is significant. Note that no descendant of a zero-tree root needs to be encoded in this step, because they are already represented by the zero-tree root symbol. In the subordinate pass, the bit n of coefficients labeled as sp/sn in any prior step is coded. In the next step, the n value is decreased by one, so that we focus now on the following bit (from Most Significant Bit (MSB) to Least Significant Bit (LSB)). This compression process finishes when the desired bit rate is reached, and the decoder can partially use the incoming bit-stream to reconstruct a progressively improved version of the original image. That is why this coder is called embedded.

In the dominant pass, four types of symbols need to be coded: sp, sn, zr, and iz, whereas in the subordinate pass only two are needed (bit zero and bit one). In order to get higher compression, an adaptive arithmetic encoder is used to encode the symbols computed during the dominant pass.

Due to its successive-approximation nature, EZW is SNR scalable, although at the expense of sacrificing spatial scalability. In addition, line-based wavelet transforms [18] are not suitable for this encoder, because the whole image is needed in memory to perform several image scans focusing on different bit planes and searching for zero-trees. Moreover, EZW needs to compute coefficient trees and performs multiple scans on the transform coefficients, which involves high computational time, most of all in cache-based architectures due to the higher cache miss rate.

Set partitioning in hierarchical trees (SPIHT)

Said and Pearlman [84] proposed a variation of EZW, called SPIHT, which is able to achieve better results than EZW even without arithmetic coding. SPIHT is based on the same principles as EZW. However, improvements are mainly due to the way it searches for significant coefficients in the quadtrees, by splitting them with a novel partitioning algorithm.

Like in EZW, SPIHT encodes the wavelet subbands in successive steps, focusing on a different bit plane in each step. For a certain bit plane (n), the set partitioning sorting algorithm included in SPIHT identifies the insignificant coefficients in the transformed image. This algorithm encodes the coefficient significance by means of significance tests, which query each set to know if it has at least one significant coefficient. If so, it divides that set into more subsets and it then repeats the same question, otherwise we have identified a group of insignificant coefficients with respect to the current bit plane. The result of each query is encoded with a simple binary symbol, so that the decoder can reconstruct the same groups of insignificant sets. The subsets with significant coefficients are successively divided until each single significant coefficient is identified. When all the subsets are found to be insignificant with respect to the current bit plane, all the significant coefficients have been located, and the sorting pass is over for this step. The algorithm then encodes the corresponding bit (n) of those coefficients found significant in previous steps, which is called the refinement pass. Afterward, it focuses on the following bit plane ($n-1$) and repeats the same process until the desired bit rate is reached. Note that the sorting and refinement passes of SPIHT are equivalent in concept to the dominant and subordinate passes of EZW, respectively.

SPIHT uses spatial orientation trees (which are basically the quadtrees of Figure 1.2) to construct the initial set of coefficients and to establish the rules to divide them in the sorting algorithm. The notation employed in the algorithm is shown in Figure 1.3(b). For a given coefficient $c_{i,j}$, $D(c_{i,j})$ is the set of all the descendant coefficients of $c_{i,j}$. This set can be split into direct

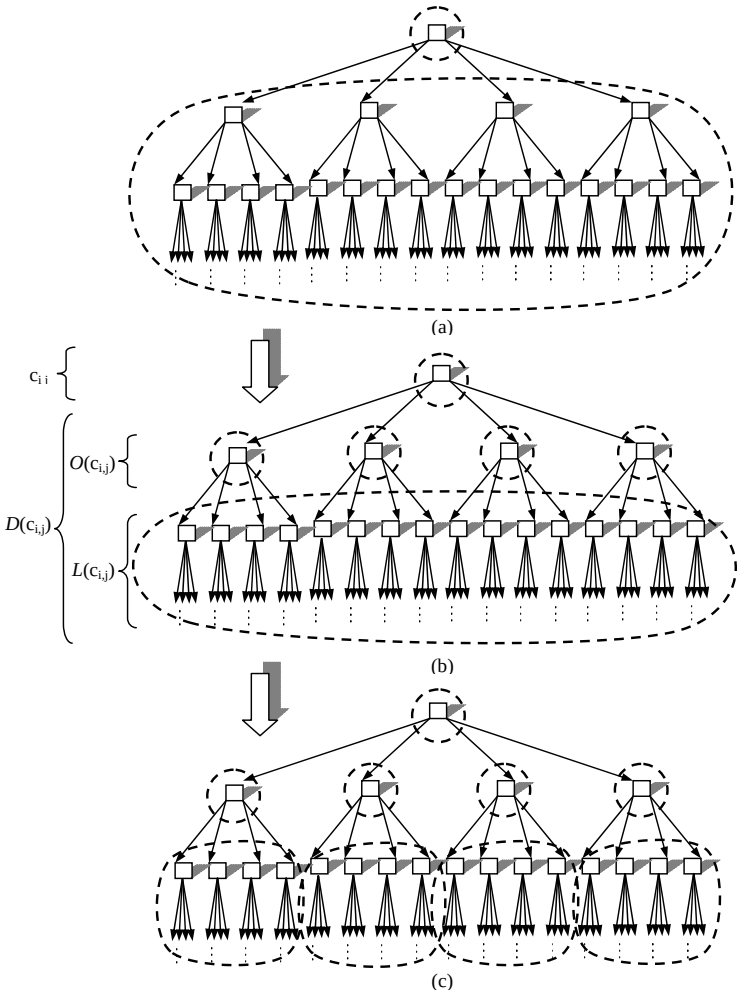


Figure 1.3: Example of division of coefficient sets arranged in spatial orientation trees. This division is carried out by the set partitioning sorting algorithm executed in the sorting pass of SPIHT. The descendants of $c_{i,j}$ presented in (a) are partitioned as shown in (b); if needed, the subset of (b) is divided as shown in (c), and so on

descendants (or offspring) $O(c_{i,j})$ and non-direct descendants $L(c_{i,j})$.

In the SPIHT algorithm, the initial sets of coefficients are defined as $D(c_{i,j})$ $\forall c_{i,j} \in LL_N$. The way a set $D(c_{i,j})$ is partitioned in a sorting pass is shown in Figure 1.3. Each set $D(c_{i,j})$, such as the one shown in Figure 1.3(a), is partitioned into its four direct descendants $d_1, d_2, d_3, d_4 \in O(c_{i,j})$ as four single coefficients, and its non-direct descendants $L(c_{i,j})$ as a new subset (see Figure 1.3(b)). Later, if the $L(c_{i,j})$ subset has to be partitioned, it

is divided into four subsets formed by $D(d_1)$, $D(d_2)$, $D(d_3)$ and $D(d_4)$, as shown in Figure 1.3(c). Each of these subsets can be further partitioned as we have just described. The detailed coding and decoding algorithms are described in [84]. In these algorithms, the sorting pass includes two lists to identify single coefficients: a list for the significant coefficients (called List of Significant Pixels (LSP)) and another for the insignificant ones (List of Insignificant Pixels (LIP)). On the other hand, the insignificant subsets are identified with another list (called List of Insignificant Sets (LIS)), in which each subset can be of type $D(c_{i,j})$ or $L(c_{i,j})$ (an extra tag is needed to specify it). Note that there is no list of significant subsets because when a subset is found to have a significant coefficient, it is successively partitioned until the significant coefficient or coefficients are refined to the granularity of a single coefficient.

The coding efficiency of SPIHT can be improved by using adaptive arithmetic coding to encode as a single symbol the significance values resulting from the significance tests (queries). The SPIHT algorithm has been considered a reference benchmark for wavelet image coding in a large number of papers. In addition, many papers have been published based on the tree-based SPIHT algorithm, including video coding [59, 51], hyperspectral image coding [103] and a generalization of the set partitioning algorithm [35]. Due to its similarities to EZW, the features of SPIHT are the same as those mentioned for EZW, except for the improvements in coding performance.

Lower tree wavelet (LTW) encoder

Not all the tree-based algorithms in the literature are based on successive quantization implemented with bit-plane coding, leading to an embedded bit-stream. LTW is a tree-based wavelet image encoder, with state-of-the-art coding efficiency, but less resource demanding than other encoders in the literature. The basic idea of this encoder is very simple: after computing a dyadic wavelet transform of an image, the wavelet coefficients are first quantized (using uniform scalar quantization by a factor Q) and then encoded with arithmetic coding.

In LTW [71], the quantization process is performed by two strategies: one coarser and another finer. The finer one consists in applying a scalar uniform quantization, Q, to wavelet coefficients. The coarser one is based on removing the least significant bit planes, *rplanes*, from wavelet coefficients. A tree structure (similar to that of [84]) is used not only to reduce data redundancy among subbands, but also as a simple and fast way of grouping coefficients. As a consequence, the total number of symbols needed to encode the image is reduced, decreasing the overall execution time. This structure

is called lower tree, and it is a coefficient tree in which all its coefficients are lower than $2^{rplanes}$.

The LTW algorithm consists of two stages. In the first one, the significance map is built after quantizing the wavelet coefficients (by means of both Q and *rplanes* parameters). In Figure 1.6 (right) we can see the significance map built from wavelet decomposition shown in Figure 1.6 (left). The symbol set employed in this proposal is the following one: a *LOWER* symbol represents an insignificant coefficient that is the root of a lower-tree, the rest of coefficients in a lower-tree are labeled as *LOWER_COMPONENT*, but they are never encoded because they are already represented by the root coefficient. If a coefficient is insignificant but it does not belong to a lower-tree because it has at least one significant descendant, it is labeled as an *ISOLATED_LOWER* symbol. For a significant coefficient, two types of 'numeric symbols' are used according to the coefficient offspring. (a) A 'regular numeric symbol' ($nbits_{i,j}$) shows the number of bits needed to encode a coefficient, (b) and a special 'LOWER numeric symbol' ($nbits_{i,j}^{LOWER}$) not only indicates the number of bits of the coefficient, but also the fact that all its descendants are labeled as LOWER_COMPONENT, and thus they belong to a lower-tree (i.e, 4^L in Figure 1.6 (right)).

Let us describe the coding algorithm. In the first stage (symbol computation), all wavelet subbands are scanned in 2x2 blocks of coefficients, from the first decomposition level to the N^{th} (to be able to build the lower-trees from leaves to root). In the first level subband, if the four coefficients in each 2x2 block are insignificant (i.e., lower than $2^{rplanes}$), they are considered to be part of the same lower-tree, labeled as *LOWER_COMPONENT*. Then, when scanning upper level subbands, if a 2x2 block has four insignificant coefficients, and all their direct descendants are *LOWER_COMPONENT*, the coefficients in that block are labeled as *LOWER_COMPONENT*, increasing the lower-tree size. However, when at least one coefficient in the block is significant, the lower-tree cannot continue growing. In that case, a symbol for each coefficient is computed one by one. Each insignificant coefficient in the block is assigned a *LOWER* symbol if all its descendants are *LOWER_COMPONENT*, otherwise it is assigned an *ISOLATED_LOWER* symbol. On the other hand, for each significant coefficient, a symbol indicating the number of bits needed to represent that coefficient is employed (see algorithm in Figure 1.4).

In order to reduce memory overhead, labels are applied by overwriting the coefficient value by an integer value associated to the corresponding label, which must be outside the possible range of significant coefficients (typically, by reusing the values in the quantized range $\left[0 \dots 2^{rplanes}\right]$).

Function: LTWCalculateSymbols( )
Scan the first level subbands (HH_1, LH_1 and HL_1) in $2x2$ blocks.
For each block B_n
if $|c_{i,j}| < 2^{rplanes}\ \forall c_{i,j} \in B_n$
Set $c_{i,j} = LOWER\_COMPONENT$
else
For each $c_{i,j} \in B_n$)
if $|c_{i,j}| < 2^{rplanes}$
Set $c_{i,j} = LOWER$
Scan the remaining subbands (from level 2 to N) in $2x2$ blocks.
For each block B_n
if $\Big(|c_{i,j}| < 2^{rplanes} \wedge descendant(c_{i,j}) = LOWER\_COMPONENT\Big)$
$\forall c_{i,j} \in B_n$
Set $c_{i,j} = LOWER\_COMPONENT\ \forall c_{i,j} \in B_n$
else
For each $c_{i,j} \in B_n$)
if $|c_{i,j}| < 2^{rplanes} \wedge$ descendant$(c_{i,j})$=$LOWER\_COMPONENT$
Set $C_{i,j} = LOWER$
if $|c_{i,j}| < 2^{rplanes} \wedge$ descendant$(c_{i,j})\neq LOWER_COMPONENT$
Set $c_{i,j} = ISOLATED\_LOWER$
End

Figure 1.4: Lower tree coding. Symbol computation

Finally, in the second stage (see algorithm in Figure 1.5), subbands are encoded from the LL_N subband to the first-level wavelet subbands, as shown in Figure 1.7. Observe that this is the order in which the decoder needs to know the symbols, so that lower-tree roots are decoded before its leaves. In addition, this order provides resolution scalability, because LL_N is a low-resolution scaled version of the original image, and as more subbands are being received, the low-resolution image can be doubled in size. In each subband, for each 2x2 block, the symbols computed in the first stage are entropy coded by means of an arithmetic encoder. Recall that no *LOWER-COMPONENT* is encoded. In addition, significant bits and its sign are needed for each significant coefficient and therefore binary encoded.

Function: LTWOutputCoefficients( )
  Scan subbands (from N to 1, in $2x2$ blocks)
  For each $c_{i,j}$ in a subband
    if $c_{i,j} \neq LOWER\_COMPONENT$
      if $c_{i,j} = LOWER$
        arithmetic_output $LOWER$
      else if $c_{i,j} = ISOLATED\_LOWER$
        arithmetic_output $ISOLATED\_LOWER$
      else
        $nbits_{i,j} = \lceil log_2(|c_{i,j}|) \rceil$
        if $descendant(c_{i,j}) \neq LOWER\_COMPONENT$
          arithmetic_output $nbits_{i,j}$
        else
          arithmetic_output $nbits_{i,j}^{LOWER}$
        output $bit_{nbits_{i,j}-1}(|c_{i,j}|) \ldots bit_{rplane+1}(|c_{i,j}|)$
        output $\text{sign}(c_{i,j})$
End
Note: $bit_n(c)$ is a function that returns the n^{th} bit of c.

Figure 1.5: Lower tree coding. Output the wavelet coefficients

Space-frequency quantization (SFQ)

Space-Frequency Quantization (SFQ) encoder presented in [115] is a non-embedded tree-based image encoder. In order to minimize distortion for a target bit rate, this algorithm relies on: (1) the construction of trees of zero-coefficients (which is considered a space quantization) and, (2) a single common uniform scalar quantization applied to the wavelet subbands (this is the frequency quantization). The joint application of (1) and (2) is performed in an optimal manner, with the Lagrange multiplier method [26]. To this end, the algorithm tries to identify the optimal subset of coefficients to be discarded by encoding them as a quadtree, and the optimal step-size to quantize the remaining coefficients by applying a uniform scalar quantizer. In order to determine the best option for the space quantization, the algorithm considers not only entire quad-trees, like the one shown in Figure 1.2, but also different shapes of trees, by pruning tree branches. Information about tree pruning and the rest of quantized coefficients, along with the employed step-size, are encoded with entropy coding and sent to the decoder as part of the compressed bit-stream.

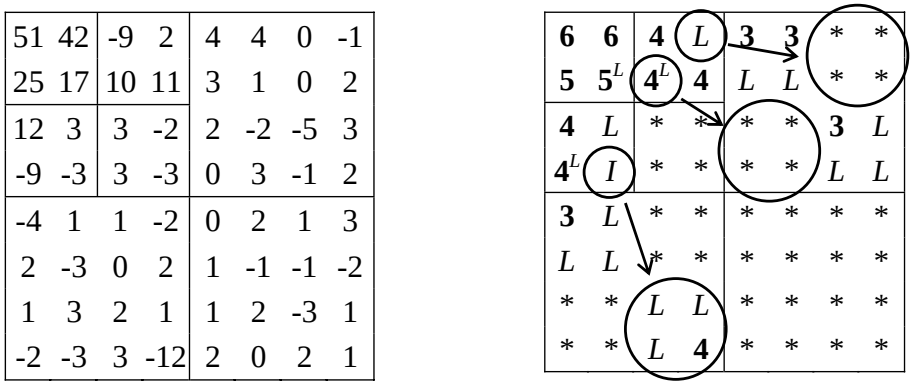


Figure 1.6: left: 2-level wavelet transform of an 8x8 example image, right: Symbol Map using *rplanes*=2

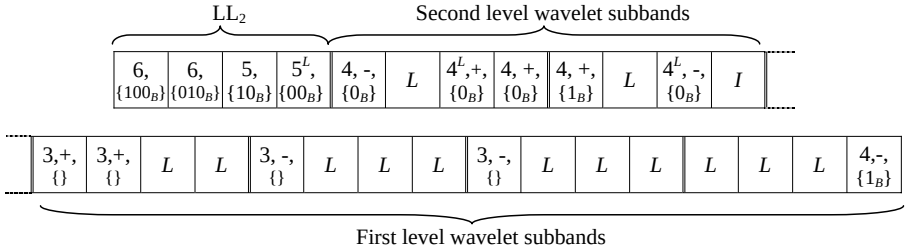


Figure 1.7: Example image encoded using LTW

Despite not being embedded, SFQ achieves precise rate control due to the use of an iterative rate/distortion optimization algorithm for a given bit rate. As a result of this algorithm, the coding performance of SFQ is slightly better than SPIHT. However, this iterative optimization algorithm is time-consuming and causes the SFQ encoder to be about five times slower than SPIHT.

Non-embedded SPIHT

In [74], Pearlman introduces the discussion about the general necessity of embedding in image coding. As we have mentioned in subsection 1.1.2, bit plane coding slows the execution of both the encoder and decoder, and sometimes it provides no benefit to the application, or even worse, it is not feasible. In particular, a line-based wavelet transform cannot be employed along with bit plane coding unless further rearrangement of the bit-stream is performed, needing at least the entire bit-stream in memory. On the other hand, we may just want to encode an image at a constant quality. In this case, successive

approximation is not strictly required, except eventually to improve coding efficiency.

The variation of SPIHT introduced in [74] is to send all the bits down to a given bit plane (r) once a single coefficient has been found significant, so as to avoid the refinement passes. In this version, the coding process finishes when that bit plane (r) is reached in a sorting pass. Another option is to pre-quantize all the coefficients with a uniform scalar quantizer, and then encode all the bit planes (again without refinement passes). The desired distortion level (or compression level) is controlled by modifying the r parameter in the first variation, or the quantization step in the second one. Note that in both approaches, the LSP list of SPIHT is no longer needed.

Although this version is faster than the original one, neither multiple image scans nor bit plane processing of the sorting passes is avoided. Hence, the problems addressed in subsection 1.1.2 still remain.

Progressive resolution decomposition (PROGRESS)

The modification of SPIHT described in the previous subsection is neither SNR nor resolution scalable. Recently, the authors of SPIHT have proposed a new version of SPIHT [16] for very fast resolution scalable encoding, based on the principles of decreasing energy of the wavelet coefficients along the subband levels, and the fact that the energy is quite similar for coefficients at the same level. Since it supports resolution scalability with great speed, the authors consider that it is an excellent choice for remote sensing and GIS applications, where rapid browsing of various scales of large images is necessary.

PROGRESS uses a pre-defined constant quality factor, just like the non-embedded SPIHT algorithm. In order to reduce complexity, bit plane coding is avoided and each coefficient is visited only once. Entropy coding is also avoided.

For each coefficient, the goal is to encode the sign and the bits below the most significant non-zero bit. To this end, the number of bits required for each coefficient must be known in advance. Basically, at a subband level, for each coefficient $c_{i,j}$ in that subband, the PROGRESS algorithm identifies the number of bits needed to encode the highest coefficient in a SPIHT-like subset $D(c_{i,j})$ (let us call this value r), and then it encodes each coefficient contained in $O(c_{i,j})$ with that number of bits. In order that the decoder can reconstruct the original coefficients, r is also encoded. In the next subband level, PROGRESS repeats the same operation for each $D(d_{m,n})\forall d_{m,n} \in O(c_{i,j})$. This algorithm is repeated through the successive subband levels, from the LL_N subband down to the first subband level. However, when the number

of bits needed to encode a subset is found to be zero, a group of insignificant coefficients has been identified, and then this subset is no longer partitioned and encoded.

In order to improve coding efficiency, each r for a given subset is not encoded as a single value, but rather as the difference between that value in this subset and in its parent subset (i.e., the direct subset from which a subset stems). Since this difference is always positive (or zero), and its probability distribution is higher as it approaches zero, unary coding[1] is employed. Some other implementation details and the complete encoding algorithm are given in [16].

Experimental results show that PROGRESS is up to two times faster in coding and four times faster in decoding than the binary version of SPIHT (i.e., SPIHT without entropy coding). However, its coding efficiency is relatively poor, being slightly worse than binary SPIHT. The low coding performance is not only due to its lack of entropy coding, but also because it always employs the number of bits required by the highest coefficient in a subset. This problem especially affects highly detailed images. These images are more likely to have high descendant coefficients, which could cause their parents to use more bits than actually needed.

1.1.3 Image coding standard: JPEG 2000

Embedded Block Coding with Optimized Truncation (EBCOT)

The EBCOT [104] encoder is certainly the most important block-based wavelet encoder reported in the literature. This encoder is a refined version of the Layered Zero Coding (LZC) technique proposed by Taubman and Zakhor in [105]. The importance of EBCOT lies in the fact that it was selected to be included as the coding subsystem of the JPEG 2000 standard [38]. EBCOT achieves most requirements of JPEG 2000, such as a rich embedded bit-stream with advanced scalability, random access, robustness, etc., by means of block-based coding for the reasons given above. Furthermore, the decrease in coding efficiency caused by the lack of inter-band redundancy removal is compensated by the use of more contexts in the arithmetic encoder, a finer-granularity coding algorithm (with three passes per bit plane instead of two), and a PCRD optimization algorithm based on the Lagrange multiplier method.

Due to the importance of EBCOT in the JPEG 2000 standard, we will describe it in some detail. For a more complete and general description, there are many other references such as [106, 3, 79] or even the standard document

[1] In unary coding, a number n is represented with n ones followed by a zero.

[38]. Note that the EBCOT algorithm originally published by Taubman in [104] was slightly changed for the JPEG 2000 standard in order to reduce complexity and other issues. We will focus on this adapted version.

After applying the DWT to the image, the EBCOT algorithm encodes the wavelet coefficients in fixed-size code blocks. In this first step, called tier 1 coding, each code block is completely and independently encoded, getting in this manner an independent bit-stream for each code block. Then, in the second step, tier 2 coding, fragments of bit-stream of each codeblock are selected to achieve the desired target bit rate (rate control) in an optimal way (i.e., minimizing distortion), and it is arranged in such a way so that the selected scalability is accomplished.

Prior to EBCOT, a uniform scalar quantization with deadzone is applied to the wavelet coefficients. All the code blocks in the same subband are quantized with the same step-size so that blocking artifacts are avoided. Therefore, in general, this quantization has little rate control meaning, which is performed later in tier 2 coding. Rather, it is used to balance the importance of the coefficient values (recall that the DWT employed in JPEG 2000 avoids dynamic range expansion but is not energy preserving), and in a practical way, to convert the floating point coefficients resulting from most wavelet transforms into integer data. Another way to select the quantizer step size is depending on the perceptual importance of each subband to improve visual quality based on the human visual system [6, 117, 57].

Regarding the code block size, the total number of coefficients in a block should not exceed 4096, and both width and height must be an integer power of two. Thereby, the typical code block size is 64x64, although other smaller sizes can be used (e.g., for memory saving or complexity issues). Of course, once a block size is determined, smaller code blocks can appear on the subband boundary or in subbands smaller than a regular block.

Block coding: tier 1 coding. Once the wavelet subbands are divided into blocks, an independent bit-stream is generated from each code block in the tier 1 coding stage. Each bit-stream is created with a special adaptive binary arithmetic encoder with several contexts called MQ-coder [99]. The MQ-coder is a reduced-complexity version of the usual arithmetic encoder [111], limited to coding binary symbols. The JPEG 2000 standard document [38] gives a detailed flowchart description of this encoder.

In this stage, each code block is encoded bit plane by bit plane, starting from the most significant non-zero bit plane. For each bit plane, several passes are given in order to identify the coefficients which become significant in this bit plane, and to encode the significant bits of those coefficients

found significant in previous bit planes. This working philosophy is shared by many other well known encoders like EZW and SPIHT. However, unlike these encoders, three passes (instead of two) are given for each bit plane[2]. In the first pass, called significance propagation pass, the significance of the coefficients that were insignificant in previous bit planes but are likely to become significant in this bit plane are encoded. Then, in the second pass, called magnitude refinement pass, a refinement bit is encoded for each coefficient found significant in a previous bit plane. Finally, the significance of the rest of coefficients (i.e., those that were insignificant and are likely to remain insignificant in this bit plane) are encoded in the third pass, called clean-up pass.

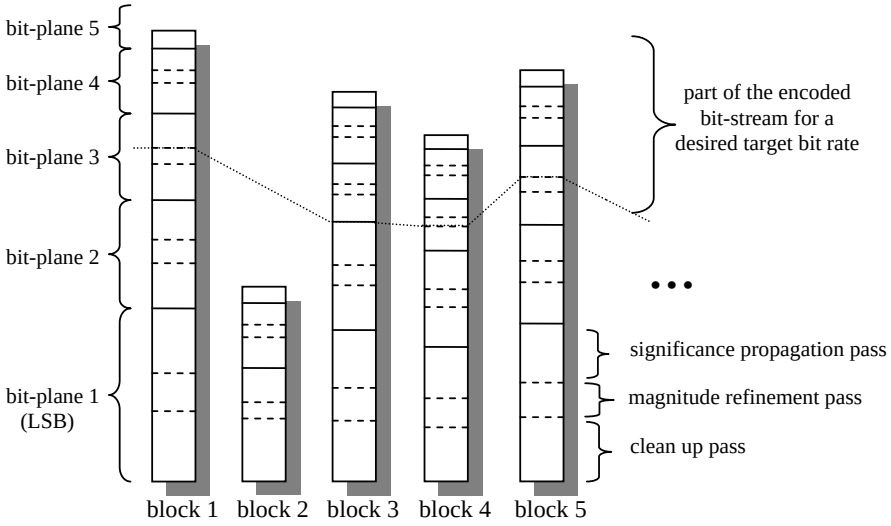


Figure 1.8: Example of block coding in JPEG 2000. In tier 1 coding, each code block is completely encoded bit plane by bit plane, with three passes per bit plane (namely signification propagation, magnitude refinement and clean-up passes). Only part of each code block is included in the final bit-stream. In this figure, the truncation point for each code block is pointed out with a dotted line. These truncation points are computed with an optimization algorithm in tier 2 coding, in order to match the desired bit rate with the lowest distortion

In tier 2 coding, the bit-stream resulting from several contiguous full passes are selected from each code block to build the final bit-stream. Therefore, the bit-stream generated from each pass is the lowest granularity for the final bit-stream formation. In each code block, the point in which its bit-stream is truncated to contribute to the final bit-stream for a given bit rate

[2]The original EBCOT algorithm [104] had four passes instead of three.

is called the optimal truncation point. Figure 1.8 illustrates the encoding process and gives an example of truncation points.

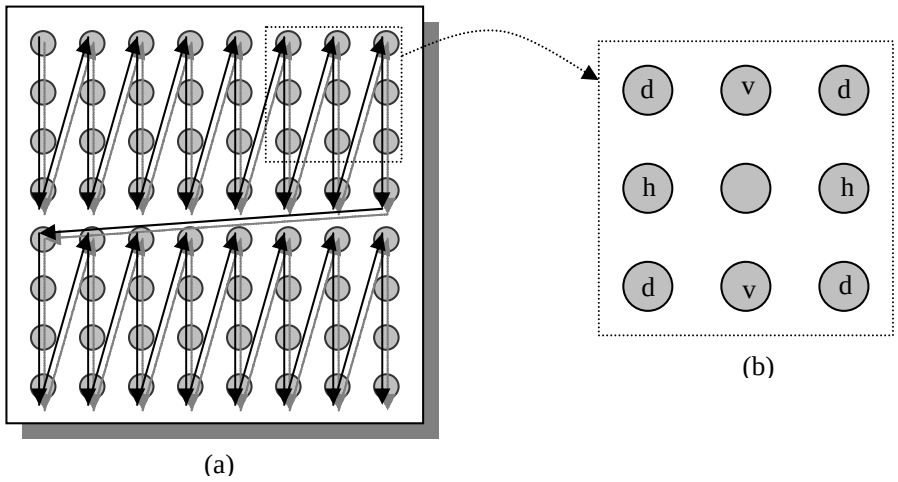


Figure 1.9: (a) Scan order within an 8x8 code block in JPEG 2000, and (b) context employed for a coefficient, formed by its eight neighbor coefficients (two horizontal, two vertical, and four diagonal)

The order of the passes has been decided according to their contribution to rate/distortion improvements, so that a pass that is more likely to introduce more reduction of distortion with a lower rate increase is encoded in first place. Of course, after encoding the three passes, the same reduction of distortion and the same bit rate is reached independently of the order of the passes. However, the proposed order yields more benefits if the truncation point is not at the end of a bit-plane coding (i.e., it is not between a clean-up pass and a significance propagation pass), but in the middle of it.

If we compare this algorithm with EZW or SPIHT in broad terms, we see that the main difference (apart from the lack of trees) is that the pass employed to identify new significant coefficients (called dominant pass in EZW and sorting pass in SPIHT) has been split into two passes in order to have more passes from which to choose a truncation point.

For implementation convenience, the order in which coefficients are scanned in a codeblock is in stripes formed by columns of four coefficients, as shown in Figure 1.9(a).

Let us see more details of each coding pass. In the significance propagation pass, a coefficient is said to be likely to become significant if, at the beginning of that pass, it has at least one significant neighbor. Certainly, this condition does not guarantee that it will become significant in this bit

plane, and therefore its significance still has to be encoded. In order to improve coding efficiency, nine contexts are used according to the significance of its eight immediate neighbors (see Figure 1.9(b)). The exact context assignment, mapping from the $2^8 - 1$ possible contexts to nine contexts, can be found in [104]. In addition, when a coefficient eventually becomes significant, its sign is also arithmetically encoded with five different contexts.

In the case of the magnitude refinement pass, a refinement bit is arithmetically encoded with two contexts if it has just become significant in the previous bit plane (i.e., it is the first bit encoded for this coefficient). For the rest of bits, they are considered to have even distribution and thereby another single context is used without dependence of the neighboring values.

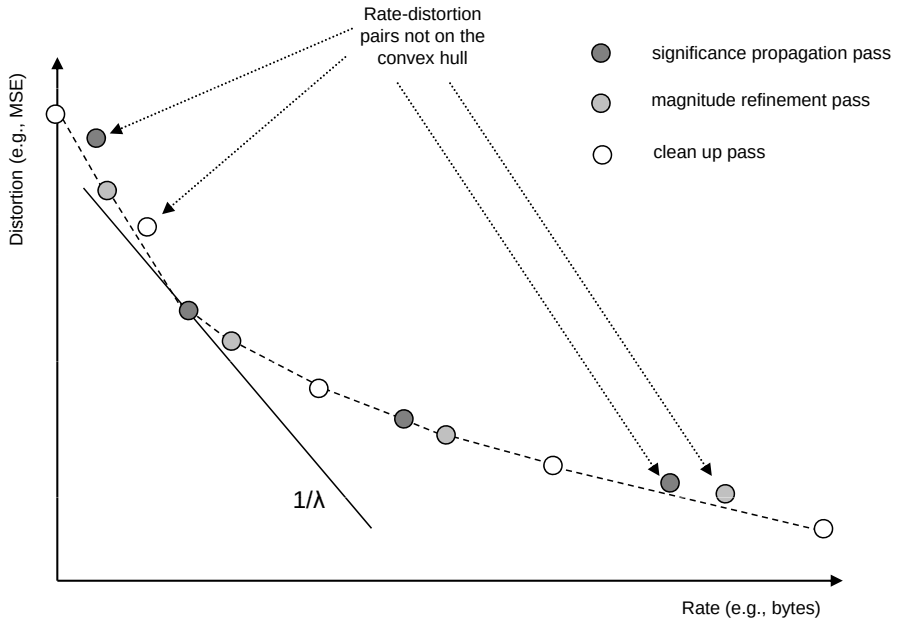


Figure 1.10: Example of convex hull formed by distortion-rate pairs from block 1 of Figure 1.8. In a convex hull, the slopes must be strictly decreasing. Four rate-distortion pairs are not on the convex hull, and therefore they are not eligible for the set of possible truncation points. A line with a slope of $1 \div \lambda$ determines the optimal truncation point for a given value of λ

The clean-up pass is implemented in a similar manner to the signification propagation pass, with the same nine contexts employed to encode the significance of a single coefficient. However, the clean-up pass includes a novel run mode, which serves to reduce complexity, rather than improve coding efficiency. Observe that most coefficients are insignificant in this pass, and therefore the same binary symbol is encoded many times. We can reduce

complexity if we take advantage of this fact and reduce the number of encoded symbols. To this end, when four coefficients forming a column have insignificant neighbors, a run mode is entered. In this mode, we do not encode single coefficients but a binary symbol that specifies if any of the four coefficients in a column is significant. This binary symbol is encoded with a single context.

Note that, for the most significant non-zero bit plane (i.e., the first bit plane that is encoded), neither a significance propagation pass nor a magnitude refinement pass is performed, because there is no previous significant coefficient (see example in Figure 1.8). Finally, it is also worth mentioning that, from the above description, we can deduce that the MQ-coder must be able to support (at least) eighteen contexts.

Bit-stream organization: tier 2 coding. In tier 2 coding, the bit-streams generated from each code block are multiplexed using a specific file format to accomplish the desired scalability. Rate control tasks are also performed in this second stage.

In order to determine the optimal truncation point in each code block for a desired bit rate, EBCOT proposes a post-compression rate distortion (PCRD) optimization algorithm, which is basically a variation of the Lagrange multiplier method [26]. This algorithm computes a convex hull (where slopes must be strictly decreasing) for each code block from a set of distortion-rate pairs (see Figure 1.10 for an example of a convex hull). Each pair defines the contribution of a coding pass to reduce image distortion (e.g., measured as Mean Squared Error (MSE) reduction) and the cost of that pass (e.g., the number of bytes required to encode that pass). For an optimal bit-stream formation, the rate-distortion pairs in the interior of the convex hull cannot be selected as truncation points.

Given the set of convex hulls for each code block, an optimal bit-stream can be achieved as follows. Consider a factor λ that defines a straight line with $1 \div \lambda$ slope. The optimal truncation point for each convex hull is given by the point to which that line is "tangent-like"[3]. In other words, it is the point at which the rate/distortion slope changes from being greater than $1 \div \lambda$ to less than it (see example in Figure 1.10). In this way, we can compute an optimal bit-stream by calculating a truncation point for each code block with a given λ. However, no rate control is performed. In order to achieve a target bit rate, the value of λ is iteratively changed and the optimal set of truncation points are recomputed with each value of λ. From all the sets of truncation points iteratively computed that do not exceed the desired bit rate, the one that yields the highest distortion reduction is selected. In other

words, we select the largest bit-stream that does not exceed the target bit rate.

Quality (SNR) scalability can be achieved if this rate control algorithm is executed several times, once for each partial target bit rate (R_1, R_2,..., R_n). Therefore, the selected coding passes that optimally lead to a bit rate R_1 are said to form the quality layer 1; then, the added coding passes that lead to a bit rate R_2 form the quality layer 2, and so on. In this way, EBCOT produces an embedded bit-stream, but with a coarser granularity than the one of EZW and SPIHT. On the other hand, for resolution scalability, we just have to arrange the selected code blocks depending on the subband level, from the LL_N to the first-level wavelet subbands. A wide variety of types of scalability is accomplished by combining various quality layers and the suitable code block arrangement in the final bit-stream.

Performance and complexity analysis. Although EBCOT only exploits intra-block redundancy, it generally performs as well as SPIHT, or even better than it, in terms of coding efficiency, mainly due to (1) the use of more contexts, (2) the introduction of a third pass to encode the most important information in first place, and (3) the PCRD optimization algorithm. In addition, if we consider artificial images or highly detailed natural images, EBCOT clearly outperforms SPIHT, because in this type of image, SPIHT can establish fewer coefficient trees, and also due to the use of more contexts in EBCOT, enabling a better and more precise adaptation of its probability model.

Let us perform a complexity analysis of EBCOT. Recall that the main complexity problem in SPIHT is introduced by bit-plane coding. Nonetheless, although both EBCOT and SPIHT use bit-plane coding, EBCOT avoids the locality problems that increase the cache miss rate by encoding an image block-by-block. Moreover, the set of code block bit-streams is more likely to fit into the cache, and therefore further post-processing does not cause so many cache misses. In spite of this, the EBCOT algorithm can be considered more complex than SPIHT (except for very large images in cache-based systems). There are several reasons for this. First, bit plane coding is still present, and for each bit plane, it must be performed for all the coefficients in a block. Compare it with EZW and SPIHT, where the coefficients in a tree are neither encoded nor scanned. Second, the significance analysis is

[3]Formally speaking, the given convex hulls are not curves and then we cannot consider a line tangent to it. Here, we actually mean a line that touches a convex hull and does not intersect it. Note that in the case of a curve, there is only a line tangent to each point, whereas in our convex hulls, there are many "tangent-like" lines for each possible truncation point.

more complex in EBCOT, since more contexts are used. Third, in a regular implementation of EBCOT, each coefficient is fully encoded, bit plane by bit plane, despite the fact that some bit planes will not be included in the final bit-stream due to rate control restrictions although some advanced implementations of JPEG 2000 perform a conservative heuristic for incrementally estimating the number of coding passes that will be included in the final bit-stream, and determine those bit planes that do not need to be computed. Finally, the PCRD optimization algorithm is performed and it is an iterative process.

1.2 Video Coding

Television won't last. It's a flash in the pan.
(Mary Somerville, radio presenter, in 1948)

1.2.1 Fundamentals

Compression is an almost mandatory step in storage and transmission of video, since, as simple computation can show, one hour of color video at CCIR 601 resolution (576x704 pixels per frame) requires about 110 GB for storing or 240 Mbps for real time transmission.

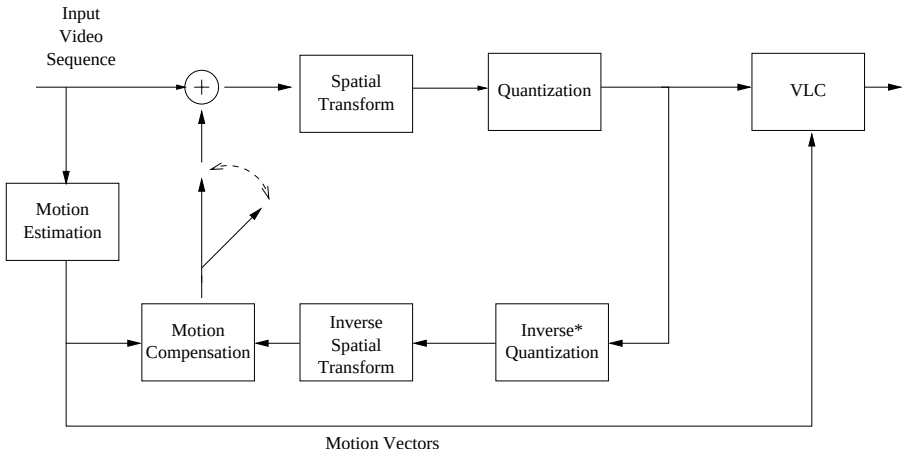


Figure 1.11: General Scheme of a Hybrid Video Encoder

On the other hand, video is a highly redundant signal, as it is made up of still images (called frames) which are usually very similar to one another, and moreover are composed by homogeneous regions. The similarity among different frames is also known as temporal redundancy, while the homogeneity of single frames is called spatial redundancy. Most video encoders perform their job by exploiting both kinds of redundancy and thus using a spatial analysis (or spatial compression) stage and a temporal analysis (or temporal compression) stage.

Hybrid video coding

The most successful video compression schemes to date are those based on Hybrid video coding. This definition refers to two different techniques used in order to exploit spatial redundancy and temporal redundancy. Spatial

compression is indeed obtained by means of a transform based approach, which makes use of the DCT, or its variations. Temporal compression is achieved by computing a Motion-Compensated (MC-ed) prediction of the current frame and then encoding the corresponding prediction error. Of course, such an encoding scheme needs a Motion Estimation (ME) stage in order to find Motion information necessary for prediction.

A general scheme of a hybrid encoder is given in Figure 1.11. Its main characteristics are briefly recalled here.

The hybrid encoder works in two possible modes: Intraframe and Interframe. In the intraframe mode, the current frame is encoded without any reference to other frames, so it can be decoded independently from the others. Intra-coded frames (also called anchor frames) have worse compression performances than inter-coded frames, as the latter benefits from Motion-compensated prediction. Nevertheless they are very important as they assure random access, error propagation control and fast-forward decoding capabilities. The intra frames are usually encoded with a JPEG-like algorithm, as they undergo DCT, Quantization and Variable Length Coding (VLC). The spatial transform stage concentrates signal energy in a few significative coefficients, which can be quantized differently according to their visual importance. The quantization step here is usually tuned in order to match the output bit rate to the channel characteristics.

In the interframe mode, the current frame is predicted by Motion compensation from previously encoded frames. Usually, Motion-Compensated prediction of the current frame is generated by composing blocks taken at displaced positions in the reference frame(s). The position at which blocks should be considered is obtained by adding to the current position a displacement vector, also known as Motion Vector (MV). Once current frame prediction is obtained, the prediction error is computed, and it is encoded with the same scheme as intra frames, that is, it undergoes a spatial transform, quantization and entropy coding.

In order to obtain Motion vectors, a ME stage is needed. This stage has to find which vector better describe current block motion with respect to one (or several) reference frame. Motion Vectors have to be encoded and transmitted as well. A VLC stage is used at this end.

All existing video coding standards share this basic structure, except for some MPEG-4 features. The simple scheme described so far does not integrate any scalability support. A scalable compressed bit-stream can be defined as one made up of multiple embedded subsets, each of them representing the original video sequence at a particular resolution, frame rate or quality. Moreover, each subset should be an efficient compression of the data it represents. Scalability is a very important feature in network delivery

of multimedia (and of video in particular), as it allows encoding the video just once, while it can be decoded at different rates and quality parameters, according to the requirements of different users.

The importance of scalability was gradually recognized in video coding standards. The earliest algorithms (as ITU H.261 norm [42, 54]) did not provide scalability features, but as soon as MPEG-1 was released [39], the standardization boards had already begun to address this issue. In fact, MPEG-1 scalability is very limited (it allows a sort of temporal scalability thanks to the subdivision in Group of Pictures (GOP). The following ISO standards, MPEG-2 and MPEG-4 [40, 41, 97] increasingly recognized scalability importance, allowing more sophisticated features. MPEG-2 compressed bit-stream can be separated in subsets corresponding to multiple spatial resolutions and quantization precisions. This is achieved by introducing multiple motion compensation loops, which, on the other hand, involves a remarkable reduction in compression efficiency. For this reason, it is not convenient to use more than two or three scales.

Scalability issues were even more deeply addressed in MPEG-4, whose Fine Grain Scalability (FGS) allows a large number of scales. It is possible to avoid further Motion Compensation (MC) loops, but this comes at the cost of a drift phenomenon in motion compensation at different scales. In any case, introducing scalability affects significantly performances. The fundamental reason is the predictive MC loop, which is based on the assumption that at any moment the decoder is completely aware of all information already encoded. This means that for each embedded subset to be consistently decodable, multiple motion compensation loops must be employed, and they inherently degrade performances. An alternative approach (always within a hybrid scheme) could provide the possibility, for the local decoding loop at the encoder side, to lose synchronization with the decoder at certain scales; otherwise, the enhancement information at certain scales should ignore motion redundancy. However, both solutions degrade performances at those scales.

The conclusion is that hybrid schemes, characterized with a feedback loop at the encoder, are inherently limited in scalability.

1.2.2 Video coding standard: H.264

The encoder (shown in Figure 1.12) has two paths known as the *forward path* (left to right) and the *reconstruction path* (right to left). In the *forward path* an input frame or field F_n is processed in MBs (16x16 pixels), and can be coded in *Intra* or in *Inter* mode. The encoder creates a reconstructed frame (P), based on reconstructed pictures samples. In *Intra* mode, P is formed

from samples in the current slice that have been previously encoded, decoded and reconstructed (uF_n in the Figure 1.12). In the Inter mode, P is created by *Motion Compensation* (MC) prediction from the reference pictures. These reference pictures may be chosen from a selection of past or future pictures that have already been encoded, reconstructed and filtered. This prediction image (P) is subtracted from the current image to produce a residual image, which will be transformed and quantized to obtain X, a set of quantized transform coefficients which are reordered and entropy encoded. The encoder also decodes the frame to provide a reference for future predictions. The X image is scaled (Q^{-1}) and inverse transformed (T^{-1}) to produce Dn. The P image is added to D_n to create the reconstructed image uF_n. However, this image is unfiltered. In the last step, a filter is used to reduce the effects of blocking distortion.

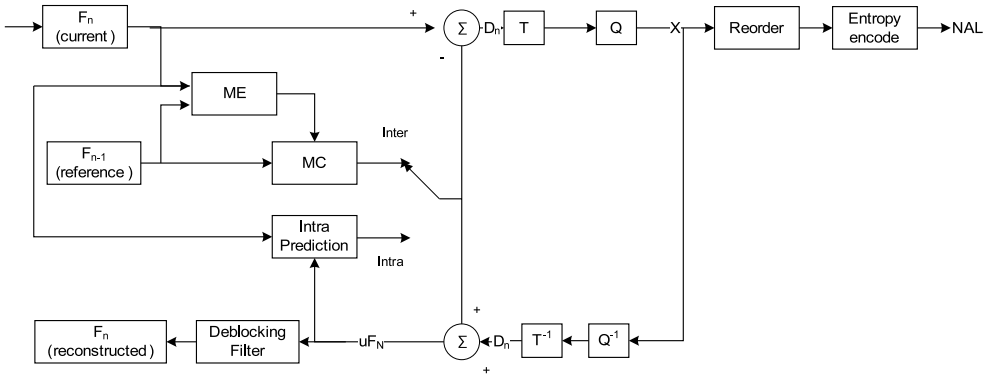


Figure 1.12: Block Diagram for an H.264 encoder

A Deblocking Filter is used to reduce blocking distortion and is applied to each decoded macroblock. This module may improve the compression performance, because the filtered image is often a more reliable reproduction of the original frame than a block and unfiltered image. In the encoder (see Figure 1.12) this filter processes the macroblock after the inverse transform T^{-1}, prior to the stage of reconstruction and storing for future predictions. In the decoder (Figure 1.13), it is the last operation of the process. The function of this module is to smooth block edges, improving the appearance of the decoded frames. The filtered image is used for motion compensation in future frames. The filter is applied to vertical and horizontal edges of 4x4 blocks in a macroblock but the edges on slices boundaries.

The Transform, used in the H.264 standard, T and T^{-1}, depends on the type of residual data to be coded. There are three kinds of transforms available: a *Hadamard Transform* (HT) for the 4x4 array of luminance *Dominant*

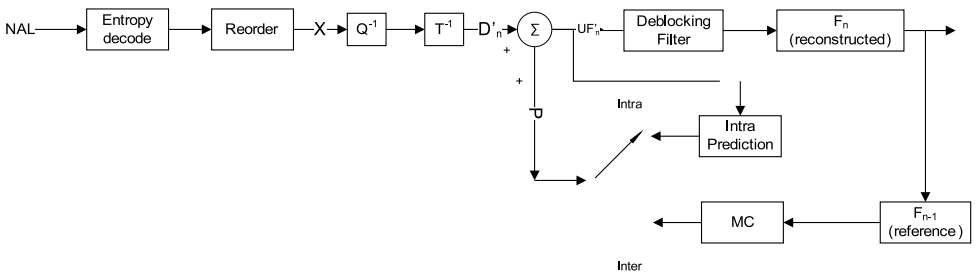


Figure 1.13: Block Diagram for an H.264 decoder

Component (DC) coefficients in Intra MBs predicted in 16x16 mode, a HT for the 2x2 array of chrominance DC coefficients in any macroblock and a DCT-based transform for all other 4x4 blocks in the residual data. The H.264 transform [31] is based on the DCT but with some fundamental differences:

- It is an integer transform, which implies no floating point operations are needed. The mismatch between the encoder and the decoder is zero without loss of accuracy.
- It can be implemented using only additions and shifts.
- The number of operations can be reduced by integrating part of the operations involved in the transform into the quantizer.

As depicted in the H.264 reference standard [1] the two dimensional DCT transform is implemented applying a one-dimensional DCT transform twice, one to the horizontal dimension and another to the vertical one [82]. In the first step, the horizontal correlation within the nxn samples block is exploited and in the second step the one-dimensional DCT transform is applied to exploit the vertical correlation.

The transformation matrix H is a 4x4 matrix defined as [1] in 1.1:

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{pmatrix} \tag{1.1}$$

The inverse transformation matrix H_{inv} is a 4x4 matrix defined as [1] in 1.2:

$$H_{inv} = \begin{pmatrix} 1 & 1 & 1 & \frac{1}{2} \\ 2 & \frac{1}{2} & -1 & -1 \\ 1 & -\frac{1}{2} & -1 & 1 \\ 1 & -1 & 1 & -\frac{1}{2} \end{pmatrix} \tag{1.2}$$

The relationship between the matrices H_{inv} and H is given by equation 1.3), where I is the Identity matrix:

$$H_{inv} \begin{pmatrix} \frac{1}{4} & 0 & 0 & 0 \\ 0 & \frac{1}{5} & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & \frac{1}{5} \end{pmatrix} H = I \tag{1.3}$$

The Quantizer, Q and Q^{-1} (Figure 1.12 and Figure 1.13), adopted by the H.264 standard, is a scalar quantizer. A total of 52 values for the *Quantification Parameter* (QP) are supported by the standard. The quantification step is doubled in size for every increment of 6 in QP. The wide range of quantizer step sizes makes it possible for an encoder to control the trade-off accurately and flexibly between bit rate and quality. Besides, the H.264 standard allows different values for the QP for luminance and chrominance. The quantization step-sizes are not linearly related to the quantization parameter (as in all prior standards). A default relationship is specified between the quantization step sizes used for luminance and chrominance, and the encoder can adjust this relationship at the slice level to balance the desired fidelity of the color components.

The Entropy encoding (Figure 1.12) or the Entropy decoding (Figure 1.13) are the modules where the elements of the sequence are encoded/-decoded, using fixed or variable length binary codes. As shown later, this operation depends of the profile being used to encode/decode the video sequence.

The entropy-encoded coefficients, together with side information required to decode each macroblock from the compressed bit-stream pass to the *Network Abstraction Layer* (NAL) where the picture will be prepared for transmission or storage. The H.264 standard does not specify the mechanism of transmitting NAL units, but a distinction is made between transmission over packet-based transport mechanisms (packet networks) and transmission in a continuous data stream (circuit-switched channels). Each NAL unit contains a *Raw Byte Sequence Payload* (RBSP), a set of data corresponding to coded video data or header information. The reason to use variable code lengths and NAL is to discriminate between coding and transport features.

On the other hand, the decoder (Figure 1.13) only has the forward path

(left to right). The data flow path in the decoder shows the similarities between encoder and decoder.

The input for the decoder is a compressed bit-stream from the NAL, and the entropy module decodes the data to generate a set of quantized coefficients, denoted by X in Figure 1.13. These are scaled and inverse transformed to give $D'n$, exactly the same $D'n$ created in the encoder (Figure 1.12), in case that there were no errors during the process. Using the information stored in the video sequence, the decoder generates the P image. The decoder adds these two images to produce $uF'n$, which will be filtered to obtain $F'n$.

In the decoder (Figure 1.13), each block coming from the quantizer is mapped into a sixteen element array in a zig-zag order. This is the function made by the reorder. This module has the function to prepare the data (reordering the coefficients for optimization) for the next module, where the entropy coding is performed. The inverse process is made by the decoder (Figure 1.13). The MB coefficients are reordered before the inverse quantification.

H.264 Inter Prediction

There are some concepts redefined in the H.264 standard which will be used in the next sections. They are summarized in the following paragraphs:

The *fields* and the *frames* are used in a different way. Both can be encoded to produce a coded picture of interlaced video, however only a frame can be coded using progressive video. The decoding order is not necessarily related to the number of frames of each encoded frame. Each coded field or frame has an associated picture order count, which defines the decoding order. Previously coded pictures, reference pictures, may be used for Inter prediction of further coded pictures.

A coded picture consists of a number of MBs, each containing 16x16 luminance samples and associated 8x8 chrominance samples (Cb and Cr in the H.264 standard) if any, depending on the sampling format. Within each picture, MBs are ordered in slices, where a slice is a set of MBs in raster scan order, but not necessarily contiguous. An I slice may contain only I MBs types, a P slice may contain P and I MB types and a B slice may contain B and I MB types.

The MB prediction (Intra or Inter) is performed in the H.264 standard using previously encoded data. In the case of the Intra prediction, an Intra MB is predicted from the current slice after having been encoded, decoded and reconstructed. For the Inter prediction, the MB is predicted using samples previously encoded. The MB prediction and the current MB are subtracted,

and the result is compressed and transmitted to the decoder, together with the information required for the decoder to repeat the prediction process (motion vectors, prediction mode, etc.). The decoder needs this information to create the prediction and adds the residual to it. The encoder must encode and decode the sequence to make sure that the decoder will have the same reconstructed information.

H.264 allows 4:2:0 progressive or interlaced video. In the default sampling format (4:2:0), chrominance samples (Cb and Cr) are aligned horizontally with every 2nd luminance sample and are located vertically between two luminance samples. Chrominance components have half the horizontal and vertical resolution of the luminance component.

The basic mechanism used to encode the residual is the *Context Based Adaptive Variable Length Coding* (CAVLC) [10]. CAVLC uses run-level coding to represent strings of zeros compactly. The number of coefficients is encoded using a look-up table, and the choice depends on the number of nonzero coefficients in neighboring blocks. This mechanism can take advantage, just in case the magnitude of nonzero coefficients tends to be larger at the start of the reordered array, and smaller towards the higher frequencies. CAVLC chooses the entry of *Variable Length Code* (VLC) look-up table for the level parameter, depending on recently coded level magnitudes.

In the H.264 standard, the MB mode decision in Inter frames (those where the motion estimation is carried out) is the most computationally expensive process due to the use of the variable block-size, motion estimation, quarter-pixel motion compensation, etc. *Inter prediction* creates a prediction model from one or more previously encoded video frames or fields using *block based motion compensation* as depicted in Figure 1.14.

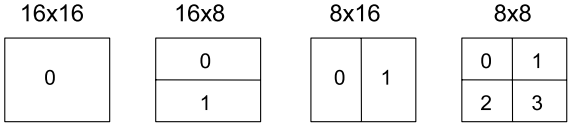


Figure 1.14: MB partitions: 16x16, 16x8, 8x16 and 8x8

H.264 uses *block-based motion compensation*, the same principle adopted by every major coding standard since H.261. Important differences from earlier standards include the support for a range of block sizes (down to 4x4) and fine sub-pixel motion vectors (1/4 pixel in the luminance component). H.264 supports motion compensation block sizes ranging from 16x16 to 4x4 luminance samples with many options between the two.

The luminance component of each MB (16x16 samples) may be divided

into four different ways (Figure 1.14): one 16x16 MB partition, two 16x8 partitions, two 8x16 partitions or four 8x8 partitions. Each of the sub-divided regions is a MB partition. If the 8x8 mode is chosen, each of the four 8x8 MB partitions within the MB may be further separated into four different ways (Figure 1.15): one 8x8 partition, two 8x4 partitions, two 4x8 partitions or four 4x4 partitions (known as *sub-macroblock partitions*). These partitions and sub-partitions give rise to a large number of possible combinations within each macroblock. This method of partitioning MBs into motion compensated sub-blocks of varying size is known as *tree structured motion compensation*.

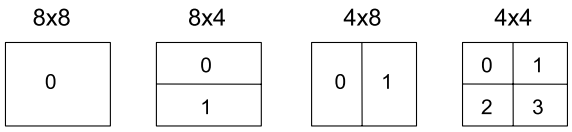


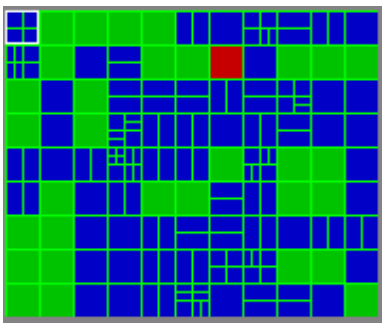
Figure 1.15: Sub-macroblock partitions: 8x8, 8x4, 4x8 and 4x4

The resolution of each chrominance component in a macroblock (Cr and Cb) is half that of the luminance component. Each chrominance block is partitioned in the same way as the luminance component, except that the partition sizes have exactly half the horizontal and vertical resolution (an 8x16 partition in luminance corresponds to a 4x8 partition in chrominance; an 8x4 partition in luminance corresponds to 4x2 in chrominance and so on). The horizontal and vertical components of each motion vector (one per partition) are halved when applied to the chrominance blocks.

Figure 1.16 shows the second frame of sequences *Foreman*, *Flower and Garden* and *Paris*, and their mode decisions made by the *Inter prediction*, in the *Baseline Profile* with all parameters as default. In the example, the best match for the present current block is given for the mode that has the smallest *Sum Absolute Differences* (SAE). See 1.17 for legend.

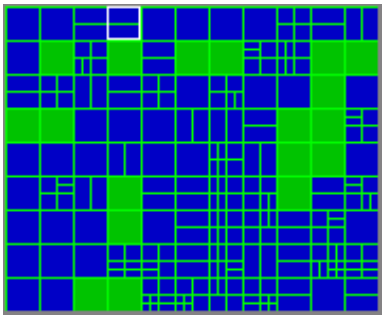
In order to evaluate the *motion vectors*, each partition in an inter-coded MB is predicted from an area of the same size in a reference picture. The offset between the two areas (the motion vector) has 1/4-pixel resolution (for the luminance component). If the video source sampling is 4:2:0, 1/8 pixel samples are required in the chrominance components (corresponding to 1/4-pixel samples in the luminance). The luminance and chrominance samples at sub-pixel positions do not exist in the reference picture and so it is necessary to create them using interpolation from nearby image samples. For example, in Figure 1.18, a 4x4 block in a frame is predicted from a region of the reference picture in the neighborhood of the current position. If the horizontal and vertical components of the motion vectors are

(a) Foreman second frame



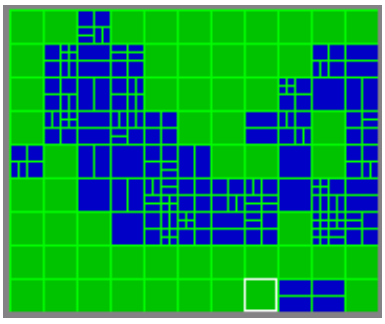
(b) Foreman second frame mode decision

(c) Flower second frame



(d) Flower second frame mode decision

(e) Paris second frame



(f) Paris second frame mode decision

Figure 1.16: Inter prediction in H.264

integers, the relevant samples in the reference block actually exist. If one or both vectors components are fractional values, the prediction samples are generated by interpolation between adjacent samples in the reference frame. Sub-pixel motion compensation can provide significantly better compression performance than integer-pixel compensation, at the expense of increased complexity. Quarter-pixel accuracy outperforms half-pixel accuracy.

Encoding a motion vector for each partition can take a significant num-

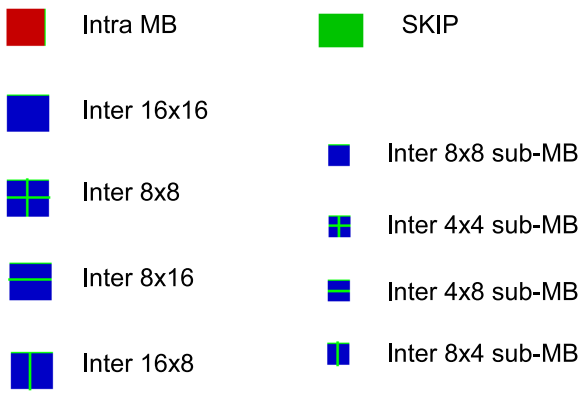
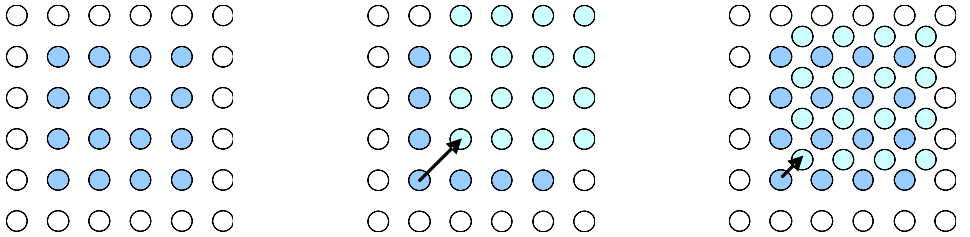


Figure 1.17: Different kinds of Inter MBs in Figure 1.16



(a) 4x4 block current frame (b) Reference vector (1,-1) (c) Reference vector (0.75,-0.5)

Figure 1.18: 4x4 example of integer and sub-sample prediction

ber of bits, especially if small partition sizes are chosen. Motion vectors for neighboring partitions are often highly correlated and therefore each motion vector is predicted from vectors of nearby, previously coded partitions. The method of forming a predicted motion vector depends on the motion compensation partition size and on the availability of nearby vectors.

H.264 Intra Prediction

H.264 incorporates an Intra picture prediction into its coding process (defined within the pixel domain) whose main aim is to improve the compression efficiency of the Intra coded pictures and Intra MBs. Intra prediction can result in significant savings when the motion present in the video sequence is minimal and the spatial correlations are significant. Throughout this section, the principle of operation of the Intra frame prediction modes as applied to the luminance and chrominance blocks will be illustrated.

While macro blocks of 16x16 pixels are still used, predicting an MB from the previously encoded MBs in the same picture is new in H.264. For luminance component, an MB may make use of 4x4 and 16x16 block prediction modes, referred to as Intra_4x4 and Intra_16x16, respectively. Recently, the Intra_8x8 block prediction mode has been added as part of the *Fidelity Range Extension* (FREext) of the standard. There are nine 4x4 and 8x8 possible block prediction directions and four 16x16 block prediction directions. For the chrominance component, an MB makes use of 8x8 block prediction mode only. There are four 8x8 possible block prediction directions. The prediction directions for the 8x8 prediction mode are similar to the ones used for the 16x16 prediction mode in the luminance component.

These intra prediction modes include a directional prediction, thus greatly improving the prediction in the presence of directional structures. With the Intra frame prediction, the I pictures can be more efficiently encoded than in other standards which do not use Intra frame prediction.

For each MB, and for each color component (Y,U,V), one prediction mode and one set of prediction directions is maintained. The H.264 encoder selects the best combination mode/direction by using the Sum of Absolute Errors (SAE). This implies that for each existing direction of each mode, the predictor within the pixel domain is created from the boundary pixels of the current partition and the SAE costs are evaluated. The best combination of mode/direction is determined corresponding to the one presenting the minimum SAE cost. The residual is encoded using a 4x4 integer based transform.

H.264 Profiles

H.264 defines a set of *Profiles*, each supporting a set of coding functions and each specifying the requirements of a decoder that satisfies the *Profile*. Table 1.1 summarizes the different options available in the three profiles defined in the H.264 standard.

In general, the *Baseline Profile* is designed for video telephony, video conferencing and wireless communications. The *Main Profile* may be useful for broadcasting media applications, such as digital television and video storage, while one potential application for the *Extended Profile* is multimedia streaming.

A video picture can be coded as such, if it has all the macroblocks of the video picture or more slices otherwise. The number of macroblocks per slice does not need to be constant within a picture. There is a minimum inter dependency between coded slices which can help to limit the propagation of errors. There are five types of coded slices shown in Table 1.2. A coded

Coding functions	Baseline Profile	Main Profile	Extended Profile
I slices	X	X	X
P slices	X	X	X
B slices		X	X
SP and SI slices			X
CAVLC	X	X	X
CABAC		X	
Slice Groups and ASO	X		X
Redundant Slices	X		X
Weighted Prediction		X	X
Data Partitioning			X
Interface		X	

Table 1.1: H.264 Baseline, Main and Extended Profiles

Slice Type	Description	Profiles
I(Intra)	Contains only I MBs	All
P (Predicted)	Contains P and/or I MBs	All
B (Bi predictive)	Contains B and/or I MBs	Extended and Main
SP (Switching P)	Facilitates switching between coded streams: contains P and/or I MBs	Extended
SI (Switching I)	Facilitates switching between coded streams: contains SI, a kind of I MBs	Extended

Table 1.2: H.264 slice mode

picture may be formed by different types of slices. The types of slices available depend on the profile selected.

In the following sub-sections the different profiles available in the H.264, the coding functions and the slice types are briefly described. Nevertheless, the interested reader can find more information related on this topic in [82].

The Baseline Profile

The *Baseline Profile* supports coded bit-streams containing I and P slices. P slices can contain Intra, Inter or skipped macroblocks. If one macroblock is encoded as skipped, no more data are sent to that macroblock. Inter MBs are predicted using previously coded pictures, using motion compensation with quarter sample motion vector accuracy (in the luminance component). The use of an H.264 encoder capable of inserting a picture delimiter RBSP

unit at the boundary between coded pictures is recommended. This shows the start of a new coded picture indicating which slice types are allowed in the following coded picture. If this mechanism is not used, the decoder will expect to detect the occurrence of a new picture based on the header of the first slice in the new picture.

Other options available in the *Baseline Profile* are resumed in the following lines:

- *Redundant slices.* The encoder can encode redundant pictures, within the full or with part of the coded picture. These pictures will be used in case the primary coded picture is damaged during transmission or storage.

- *Arbitrary Slice Order* (ASO). The slices in a coded frame may follow any decoding order.

- *Slice groups.* A slice group is a subset of the macroblocks in a coded picture and may contain one or more slices. Within each slice in a slice group, MBs are coded in raster order. If only one slice group is used per picture, then all the MBs in the picture are coded in raster order. In this case, ASO can not be used.

The Main Profile

In general, the *Main Profile* is a superset of the *Baseline Profile* where B slices (bi-predicted), weighted prediction for creating a motion-compensated prediction block, interlaced video (frames or fields) and *Context-base Adaptive Binary Arithmetic Coding* (CABAC) as entropy coding method, are mechanisms enhancing the Baseline Profile. These mechanisms are optional; they can be enabled or disabled in the H.264 standard. However, in this profile the redundant slices, ASO and multiple slice groups are not supported.

A B slice may be predicted from one or two reference pictures, before or after the current picture in temporal order. It depends on the reference pictures available in the encoder and decoder. In this way, there are more options to select the prediction reference for the macroblocks in a B slice. Macroblock partitions in this kind of slice can be done in direct mode, motion-compensated or motion-compensated bi-predictive. The different algorithms proposed in this dissertation only run with I and P slices, reason for which no more details will be provided on this kind of slices.

Weighted Prediction is a method of modifying the samples of motion-compensated prediction data in a P or B slice macroblock. The prediction samples may be scaled by a weighting factor, before obtaining the motion

compensated prediction. A large weighting factor is applied if the reference picture is temporally close to the current picture and a smaller factor is applied if the reference picture is temporally far away from the current picture. This tool may be useful when the sequence has *fade transitions*, where one scene fades into another.

Another functionality available in the *Main Profile* is the interlaced video. The encoder can choose to encode each MB pair as two frame MBs or two field MBs and may select the optimum coding mode for each region of the picture. Coding a slice or macroblock pair in field mode requires modifications to a number of encoding and decoding steps. All the coded fields are treated as separate reference pictures for the P or B slice prediction. The prediction of coding modes in Intra macroblocks and motion vectors in Inter macroblocks require modification, depending on whether adjacent macroblocks are coded in frame or field mode.

The CABAC [107, 58], achieves good compression performance by selecting probability models for each syntax element according to the element's context, adapting probability estimates based on local statistics and using arithmetic coding rather than variable length coding. The definition of the decoding process is designed to facilitate low complexity implementations of arithmetic encoding and decoding. Besides, CABAC provides improved coding efficiency compared with VLC. The arithmetic operations for implementing the CABAC are described in the H.264 standard decoder [1].

The Extended Profile

The *Extended Profile* focuses on video streaming applications. As shown in Table 1.1, it includes all the *Baseline Profile* characteristics. The new features focus on supporting efficient streaming over packet switched networks, error resilience and noise environments.

SP and SI slices allow efficient switching between video streams and random access for the video decoders [46]. Over the Internet, where the data throughput may drop suddenly, the decoder can switch automatically between the same sequence encoded using different bit rates. This is the function of the SP slices. They are designed to support switching between similar coded sequences. For example, the same sequence at different bit rates. In this case, the motion compensated prediction may be very efficient. This solution is better than inserting I frames at switching points, improving the performance too. Besides, SP slices allow random access features. On the other hand, SI slices are used to pass between one sequence to a completely different sequence, in which case it will not be useful to use motion compensated images, because there is no relationship between them. More detailed

treatment of the process can be found in [47].

The *Data Partitioned Slices* is a feature designed to improve the robustness of the transmission of an H.264 encoded sequence. The coded data of a slice are distributed into three different partitions, each of them containing a subset of the data. The first one has the header of the slice and the header of the data for each macroblock. This partition is highly sensitive to transmission errors. The second partition contains the residual data for the Intra and SI slice macroblocks and the last one contains the coded residual data for Inter coded macroblocks, forward and bi-directional. The data of each partition can be placed in a separate NAL unit, i.e. they can be stored or transmitted separately. If some data from the two last partitions are lost, the sequence may be decoded and part of the missed information can be reconstructed.

1.2.3 Wavelet based video encoders

The first attempts to use Subband Coding, and in particular Wavelet Transform (WT), in video coding date back to late 80s [48]. It is quite easy to extend the WT to three-dimensional signals: it suffices to perform a further wavelet filtering along the time dimension. However, in this direction, the video signal is characterized by abrupt changes in luminance, often due to objects and camera motion, which would prevent an efficient de-correlation, reducing the effectiveness of subsequent encoding. In order to avoid this problem, MC is needed. Anyway, it was soon recognized that one of the main problems of WT video coding was how to perform MC in this framework, without falling again into the problem of closed loop predictive schemes, which would prevent exploiting the inherent scalability of WT.

Actually, in such schemes as [48, 50, 51] three-dimensional WT is applied without MC: this results in unpleasant ghosting artifact when a sequence with some motion is considered. The quality objective is just as well unsatisfactory. The idea behind Motion Compensated WT is that the low frequency subband should represent a coarse version of the original video sequence; motion data should inform about object and global displacements; and higher frequency subbands should give all the details not present in the low frequency subband and not caught by the chosen motion model as, for example, luminance changes in a (moving) object.

A first solution was due to Taubman and Zakhor [105], who proposed to apply an invertible warping (or deformation) operator to each frame in order to align objects. Then, they perform a three-dimensional WT on the warped frames, achieving temporal filtering which is able to operate along the motion trajectory defined by the warping operator. Unluckily, this motion model is

able to effectively catch only a very limited set of object and camera movements. It has been also proposed to violate the invertibility in order to make it possible to use a more complex motion model [108]. However, preventing invertibility makes high quality reconstruction of the original sequence impossible.

A new approach was proposed by Ohm in [68, 69], and later improved by Choi and Woods [17] and commonly used in the literature [112]. They adopt a block-based method in order to perform temporal filtering. This method can be considered as a generalization of the warping method, obtained by treating each spatial block as an independent video sequence. In the regions where motion is uniform, this approach gives the same results as the frame-warping technique, as corresponding regions are aligned and then undergo temporal filtering. On the contrary, if neighboring blocks have different motion vectors, we are no longer able to correctly align pixels belonging to different frames, since "unconnected" and "multiple connected" pixels will appear. These pixels need special processing, which does not correspond anymore to the subband temporal filtering along motion trajectories. Another limitation of this method is that motion model is restricted to integer-valued vectors, while it has long been recognized that sub-pixel motion vectors precision is remarkably beneficial.

A different approach was proposed by Secker and Taubman [89, 90, 91, 92] and, independently by Pesquet-Popescu and Bottreau [77]. This approach is intended to resolve the problems mentioned above, by using Motion Compensated Lifting Schemes (MC-ed LS). As a matter of fact, this approach proved to be equivalent to applying the subband filters along motion trajectories corresponding to the considered motion model, without the limiting restrictions that characterize previous methods. The MC-ed LS approach proved to have significatively better performances than previous WT-based video compression methods, thus opening the doors to highly scalable and performance-competitive WT video coding.

Chapter 2

Rate Control Tools

Contents

The problem of rate control for wavelet-based image coding is to create a compressed image of an exact size. Based on a performance metric or quantization strategy, this task requires the selection of subband quantization step sizes, such that when an entropy coder is applied to the quantized subbands, the compressed file size matches the desired size. For some applications, the problem of rate control has been avoided with embedded coders, such as SPIHT [84]. For low bit rate coding, however, a truncated embedded representation might not be optimal, especially in a perceptual setting such as in [14], where step-sizes cannot be described as a single function of bit rate. Given an optimization criterion (such as mean squared-error or a perceptual distortion measure) that dictates some relationship between subband step-sizes, a bisection search can be used to implement accurate rate-control. A set of step-sizes generates quantized subbands where the sum of the compressed subband sizes equals the target file size. A method for mapping step-sizes to bit rate, however, is needed to run any such rate control algorithm.

Traditional methods of mapping step-size to rate include compressing the quantized data and subband modeling. Simply recompressing an image until the rate constraint is met is neither elegant nor efficient. Gaussian subband modeling can be used to map step-size to rate with much less computation, but is not effective at low rates because heavily quantized subband data does not fit well with Gaussian distributions. In addition, characterizing a subband with only a distribution function does not address coefficient dependencies that are important when most coefficients in a quantized subband are zero. More sophisticated methods have been proposed for modeling subband rate-quantization step-size (R-Q) relationships. Gormish and Gill presented in [28] a model whereby each subband is treated as a quantized Laplacian process. This technique performs well but overestimates the entropy of subbands at low bit rates. Generalized Gaussian modeling has also been proposed as the basis for a stripe-based rate control procedure in wavelet coders [73], which is applied on-the-fly to successive rows in a given image.

Mallat and Falzon in [55] introduced an analytical framework for coarsely quantized subbands that models the entropy of the location and magnitude of non-zero quantized wavelet coefficients separately. The model is based on run-length encoding the locations of non-zero wavelet coefficients and entropy coding the remaining quantized coefficients.

Percentage of significant coefficients has also been suggested as a parameter to describe subband R-Q behavior [11, 33]. As a method of handling significant coefficient dependencies, the relationship between rate and percentage of significant coefficients is coupled with a method to map this percentage to step size. Nevertheless, this kind of model requires training data

(such as the rates associated with several step sizes) to be fit correctly.

In [27] the bootstrap rate-control algorithm is presented. The algorithm uses a probability model to obtain an accurate rate estimation scheme. The algorithm iterates on the models using Newton's method.

In this chapter, several lightweight rate-control algorithms for non-embedded coding with increasing complexity and accuracy are proposed. These algorithms will predict the proper quantization values that lead to a final bit rate close to the target one. In order to evaluate the proposed rate control methods, the LTW encoder, which was briefly described in Section 1.1.2, has been selected.

2.1 Zero-order entropy based rate control

This method is based on the zero-order entropy (Eq. 2.1) of the wavelet coefficients. The estimation of the quantization parameters is based on the correlation between DWT coefficients entropy, target bit rate and quantization parameters.

$$H(x) = -\sum_{x} p(x) log_2 \left(p(x)\right) \tag{2.1}$$

We use the Kodak image set [19] as a representative set for our purposes and the LTW encoder with both Q and *rplanes* quantization parameters. As there is a correlation between the DWT coefficients entropy and the quantization parameters, we can establish a relationship between them for a given target bit rate by means of curve and surface fitting techniques [118][23][52][102]. In particular, a surface fitting process was driven by polynomial bivariate (bit rate and entropy) equations due to its low computational complexity. So, equations (2.2), (2.3) and (2.4) represent the surface fitting expressions corresponding to the fine quantizer (Q) estimation for *rplanes* values of 2, 3 and 4, respectively. The variables *'x'* and *'y'* represent the coefficients entropy and the target bit rate, respectively, and constant values a, b, c, d, e, f, g, h, i and j are computed through the aforementioned surface fitting methods using the Kodak image set. For each equation, we also show the Coefficient of Determination (r^2) that measures the goodness of fit (ideally $r^2 = 1$).

$$Q_{rp2} = a + bx + c/y + dx^2 + e/y^2 + fx/y + gx^3 + h/y^3 + \\ + ix/y^2 + jx^2/y$$
$$a = 23.997462\ b = -23.685546\ c = -1.2740571\ d = 7.366104$$
$$e = 0.067482965\ f = 1.1057934\ g = -0.72773395\ h = 0.0033011333$$
$$i = -0.062838865\ j = -0.0092142026$$
$$\left(r^2\right) = 0.94993174 \quad (2.2)$$

$$Q_{rp3} = a + b/x + c\ln y + d/x^2 + e\left(\ln y\right)^2 + f\left(\ln y\right)/x + \\ + g/x^3 + h\left(\ln y\right)^3 + i\left(\ln y\right)^2/x + j\left(\ln y\right)/x^2$$
$$a = 13.044539\ b = -69.088897\ c = -5.9821471\ d = 129.6753$$
$$e = 0.58350021\ f = 19.072322\ g = -82.793779\ h = -0.070997754$$
$$i = -0.6638497\ j = -16.319499$$
$$\left(r^2\right) = 0.95041672 \quad (2.3)$$

$$Q_{rp4} = a + b/x + c\ln y + d/x^2 + e\left(\ln y\right)^2 + f\left(\ln y\right)/x + \\ + g/x^3 + h\left(\ln y\right)^3 + i\left(\ln y\right)^2/x + j\left(\ln y\right)/x^2$$
$$a = 5.296599\ b = -19.54813\ c = -2.2959214\ d = 25.767952$$
$$e = 0.15159821\ f = 4.7468293\ g = -11.718514\ h = -0.019589203$$
$$i = -0.019548644\ j = -2.54051$$
$$\left(r^2\right) = 0.968726 \quad (2.4)$$

Although this estimation method suffers from severe errors when working at low and high entropy values, its complexity is low. In Figure 2.1, the Entropy-based algorithm is shown. First, the algorithm sets the *rplanes* value depending on the target bit rate (T_{bpp}). This value has been calculated empirically for this algorithm. Particularly, the best choice for a target bit rate in the range $0.125 - 0.5$ bpp is $rplanes = 4$, $rplanes = 3$ in the range $0.5 - 1.5$ bpp and $rplanes = 2$ in the range $1.5 - 2$ bpp. But later on, we will define a more precise method to compute the *rplanes* parameter. Secondly, we apply the coarser quantization by removing the selected *rplanes* less significant bits from all wavelet coefficients. Then, the coarser quantized wavelet coefficients entropy is calculated, and finally we use the corresponding

rplanes value fitting equation to obtain the finer quantization (Q) value. Table 2.1 shows the minimum, maximum and average estimation error of

Input: Wavelet Coefficients ($C_{i,j}$), Target bit rate ($Tbpp$),
Surface Fitting Equations for each *rplanes* value(Eq_i)
Output: Q, *rplanes*
(E1) Determine *rplanes* using T_{bpp}
(E2) Remove the *rplanes* less significant bits to all wavelet coefficients ($C_{i,j}$)
(E3) Calculate Wavelet Coefficients Entropy, Se
(E4) Select surface equation Eq_i using *rplanes*
(E5) Determine quantization parameter (Q)
$Q_{rplanes}$ = Solve equation Eq_i where $x = Se$ and $y = T_{bpp}$

Figure 2.1: Entropy-based rate control algorithm

the Entropy-based algorithm at different target bit rates for all images in the Kodak set. As can be seen, there are discontinuities among the relative estimation errors with respect to the growing bit rate. This ocurs because the surface fitting equation suffer from severe errors at the surface boundary lines.

	0.125 bpp	0.25 bpp	0.5 bpp	1 bpp	2 bpp
Min.%	3.79	0.56	0.72	0.04	0.11
Max.%	150.88	62.65	62.09	108.6	50.29
Avg.%	44.64	14.25	16.76	15.35	9.04

Table 2.1: (Entropy-based) - Relative Estimation Error

2.2 Rate control based on a trivial coding model

It is very difficult to estimate the quantization parameters at a certain degree of accuracy for a target bit rate using only the wavelet coefficients entropy. So we decided to study how the encoder works in order to define a simplified statistical model of the encoding engine in a similar way as in [30] and [49].

In [30] the authors propose an expensive rate allocation scheme based on the Lagrangian optimization method that offers a low accurate rate control capability. In [49] another statistical model based on the Generalized-Gaussian Densities (GGD) approach is proposed, obtaining an expensive but highly accurate rate control behavior.

We will use the LTW coding engine in order to define a simple statistical model that will be able to supply a fast and accurate estimation of the resulting bit rate. Under this model, given a DWT transformed image and for each specific *rplanes* value (from 2 to 7), we calculate the probability that a coefficient is lower than $2^{rplanes}$ (in other words, insignificant) and also the probability that a coefficient needs *rplanes*+1 bit, *rplanes*+2, and so on. After that the probability distribution function (pdf) of the LTW symbol map is available, and as consequence we can get a lower bound bit rate estimation of the bit rate required by the symbol map encoding by means of its zero-order entropy (S_e).

Since we also know the number of significant wavelet coefficients and the number of bits needed for coding their value and sign, we can calculate the exact number of bits sent to the output bit-stream, as they are raw binary encoded ($Bits_{total}$).

So the final bit rate estimation for each *rplane* value ($E_{bpp}(rplanes)$) is obtained by adding the arithmetic encoder estimation (S_e) to the raw encoding bit count of significant coefficients (Eq. 2.5)

$$E_{bpp}(rplanes) = S_e(rplanes) + Bits_{total}(rplanes) \tag{2.5}$$

The resulting estimation gives a biased measure of the real bit rate for all the operative bit rate range (from 0.0625 to 1 bpp). The error between the model bit rate estimation (E_{bpp}) and target bit rate is mainly due to the assumption of a zero-order entropy coder whereas the LTW encoding scheme uses an adaptive arithmetic encoder with context modeling. We reduce this error by means of an adjust function that is defined heuristically from the entire Kodak image set. Figure 2.2 shows the bit rate estimation error for all images in the Kodak set and the corresponding curve fitting for *rplanes*=4 (no finer quantization is applied). This curve will determine the model estimation error at $rplanes = 4$. The same process was done for every *rplane* value from 2 to 7. So we have obtained the adjust function that we will apply to reduce the model estimation error for each *rplane* value. In Figure 2.3 we show the estimated and target bit rates resulting from coding the whole Kodak image set with an *rplane* value of 4. The estimation error is significantly reduced after applying the corresponding adjust function.

After that, the target bit rate (T_{bpp}) will determine the proper value of *rplanes* by choosing *rplanes* so that $E_{bpp}(rplanes) \geq T_{bpp} > E_{bpp}(rplanes{+}1)$.

Once the *rplanes* value is determined, we have to estimate the Q value that will produce a bit rate as close as possible to the target bit rate. Note that $Q = 0.5$ indicates no scalar uniform quantization as could be extrapolated from equation 2.6. On the other hand, $Q \cong 1.16$ is the maximum scalar uniform quantization at a given *rplane* value, because point $(Q \cong 1.16, E_{bpp}(rplanes))$ is roughly the same point as $(Q = 0.5, E_{bpp}(rplanes+1))$. For this purpose, we observed that the bit rate progression from *rplane* to *rplane* + 1 follows a second order polynomial curve that shares nearly the same x-value of the vertex (K_{min})[1] for all images in the Kodak set (see Figure 2.4).

Since we know three points of that curve $E_{bpp}(rplanes)$, $E_{bpp}(rplanes+1)$ and the curve vertex (K_{min}), we can build the corresponding expression that will supply the estimated value of Q for a given target bit rate.

$$Coef_q = \frac{Coef}{2Q} \tag{2.6}$$

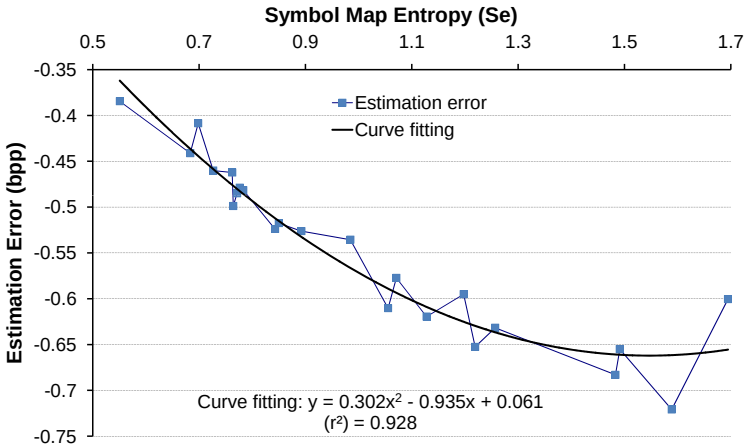


Figure 2.2: (Model-based) - Estimation error reduction by defining an adjust function from the entire Kodak image set (*rplanes*=4)

The whole algorithm, shown in Figure 2.5, works as follows:

- First (E1), we estimate the resulting bit rate $(E_{bpp}(rplanes))$ after applying only the coarser *rplanes* quantization to wavelet coefficients for *rplanes* values from 2 to 7.

[1]Note that there is a K_{min} value for every *rplane*, and it has been calculated as the average x-value of the vertex for all images in the Kodak set.

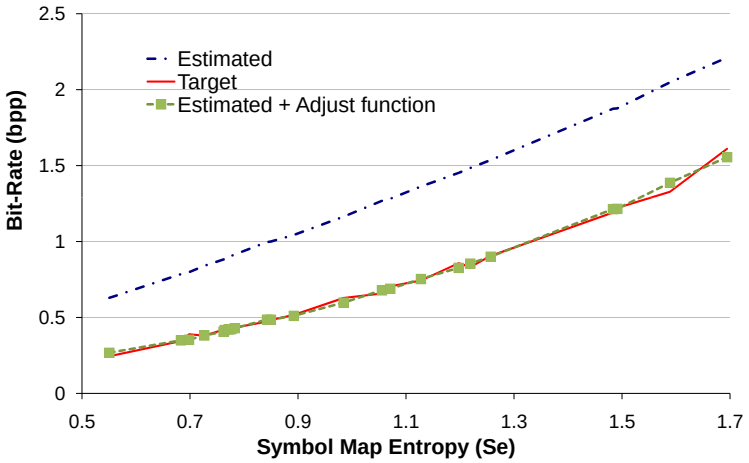


Figure 2.3: (Model-based) - Estimated vs Real bit per pixel for the entire Kodak image set (*rplanes*=4)

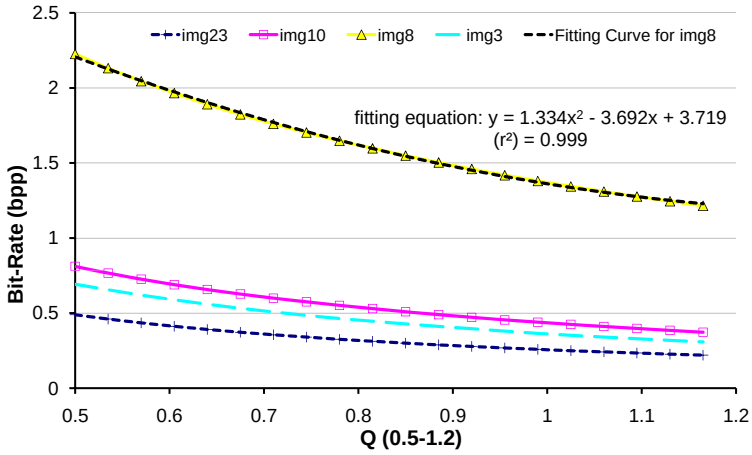


Figure 2.4: (Model-based) - Bit rate progression of four images from the Kodak set from *rplanes* 3 to *rplanes* 4

- Second (E2), we apply the corresponding adjust functions to these estimations.
- Third (E3), we set the appropriate *rplanes* value for the requested target bit rate (T_{bpp}).
  $(E_{bpp}(rplanes) \geq T_{bpp} > E_{bpp}(rplanes + 1))$.
- Next (E4), we obtain the quadratic expression so as to determine the value of Q by using the Newton interpolation algorithm.
- Then, we solve the expression to obtain the estimated Q value.

Input: Wavelet Coefficients ($C_{i,j}$), Target bit rate (T_{bpp}),
Curve minimum (K_{min})
Output: Q, *rplanes*
(E1) for each *rplanes* in [2..7]
for each $C_{i,j}$ coefficient
$nbits_{i,j} = \lceil \log(|C_{i,j}|) \rceil$
if $nbits_{i,j} > rplanes$
$Symbol_{(nbits_{i,j}-rplanes)} += 1$
$Bits_{total} += nbits_{i,j} - rplanes$
else
$Symbol_{non-significant} += 1$
Calculate the Symbols Map Entropy, S_e
$E_{bpp} = (Bits_{total}/sizeof(image)) + S_e$
(E2) for each *rplanes* in [2..7]
Apply_Adjust_Function;
(E3) Determine *rplanes*
$E_{bpp}(rplanes) > T_{bpp} > E_{bpp}(rplanes + 1)$
(E4) Determine quantization parameter (Q)
Obtain A,B,C using $E_{bpp}(rplanes)$, $E_{bpp}(rplanes + 1)$
and K_{min} for $T_{bpp} = A.Q^2 + B.Q + C$

Figure 2.5: Model-based rate control algorithm

The Model-based algorithm yields a lower error on the estimation process over the Kodak image set than the Entropy-based algorithm. Furthermore, the choice of the *rplanes* parameter has been included in the estimation process. Table 2.2 shows the average estimation error of the Model-based algorithm at different target bit rates. The relative error produced is around 5% on average at low compression ratios and it grows up to 9% at higher compression ratios. This is due to the high slope of the quadratic expression used to obtain Q when the *rplanes* parameter grows, so a slight change over the Q parameter implies a high variation on the bit rate and as a consequence, a higher estimation error.

2.3 Lightweight iterative rate control

With the Model-based rate control algorithm described in the previous subsection, we can define an iterative version to reduce the estimation error

Input: Wavelet Coefficients ($C_{i,j}$), Target bit rate (T_{bpp}),
Curve minimum (K_{min}) and
Maximum Allowed Error (MAE)
Output: Q, *rplanes*
(E1) Obtain *rplanes* and Q using algorithm from Figure 2.5
(E2) Error = Encode_And_Evaluate_Error
(E3) Iterative stage
if $error > MAE$
$NewT_{bpp} = T_{bpp}$
for i=1 to MAX_ITERATIONS
if i in [1..3]
$NewT_{bpp}$ += error
error= Encode_And_Evaluate_Error
Points[i]=(Q,$Real_{bpp}$)
else
Q=Newton($Points_{(Last\_three)}$)
error= Encode_And_Evaluate_Error
Points[i]=(Q,$Real_{bpp}$)

Figure 2.6: Iterative rate control algorithm

	0.125 bpp	0.25 bpp	0.5 bpp	1 bpp
Min.%	0.42	0.20	1.31	0.44
Max.%	15.69	16.26	13.63	12.89
Avg.%	8.50	7.48	5.11	4.46

Table 2.2: (Model-based) - Relative Bit rate estimation Error

with a moderate computational complexity increase. Thus, depending on the application requirements, we can get the proper trade-off in the prediction between both factors: complexity and accuracy. Now, we can define the maximum allowed estimation error as a relative or absolute error and the algorithm will perform coding iterations until this condition is satisfied or a maximum number of iterations is reached.

In the first iteration the proposed algorithm will estimate the *rplanes* and Q values for the target bit rate by using the algorithm described in the previous subsection (see Figure 2.5). Then the source image will be coded with the quantization parameters found. If the resulting bit rate error is

lower than the maximum allowed error (MAE), then the algorithm finishes; otherwise, it makes a new Q estimation based on the observed error. This is done during the first three iterations, getting for each one a new real point (Q, bit rate) from the Rate/Quantization second order polynomial curve associated to the image. In the following iterations, the new three real points obtained are used to get a new more accurate quadratic polynomial curve for Q by means of the Newton interpolation algorithm if the error has not been previously fitted (see algorithm in Figure 2.6).

Table 2.3 shows the iterations carried out by the iterative algorithm to produce the desired target bit rate in the range from 0.0625 to 1 bpp at different maximum allowed relative errors (MAE) for all images in the Kodak set.

Iterations	2%	1%	0.5%	0.25%
Min.	1	1	2	2
Max.	4	5	5	6
Avg.	2.36	2.89	3.18	3.6

Table 2.3: (Iterative) - Iterations at different values of the maximum allowed relative error

2.4 Rate control evaluation

Using a C++ implementation of the LTW encoder, the different proposals were developed and tested on an Intel PentiumM 1.6 Ghz processor. To determine the curve fitting and error adjustments in the first two methods, we have used the Kodak image set as a representative set of natural images. We restrict our proposals operation limits to the range from 0.0625 to 1 bpp. Finally, we used the Barbara, Bike, Boat, Cafe, Goldhill, Lena, Mandrill, Peppers, Woman and Zelda test images (outside the Kodak set) to validate the proposed methods.

In Figure 2.7 we see that the Model-based algorithm gets the best results for all bit rates in the range 0.625 to 1 bpp. Although for several images in the Kodak set the Entropy-based algorithm performs better, the maximum error peaks in the Model-based algorithm are significantly lower than in the Entropy-based one.

In Figure 2.8, we show the bit rate accuracy of the proposed rate control methods for the Goldhill test image. As can be seen in Table 2.4, the behavior is very similar in the other non-Kodak test images. In general, the Model-based method does not work efficiently at very low bit rates in the range from

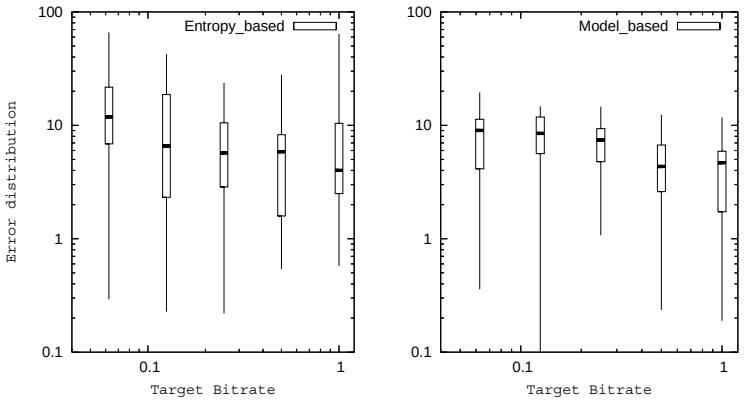


Figure 2.7: Entropy-based vs Model-based error prediction

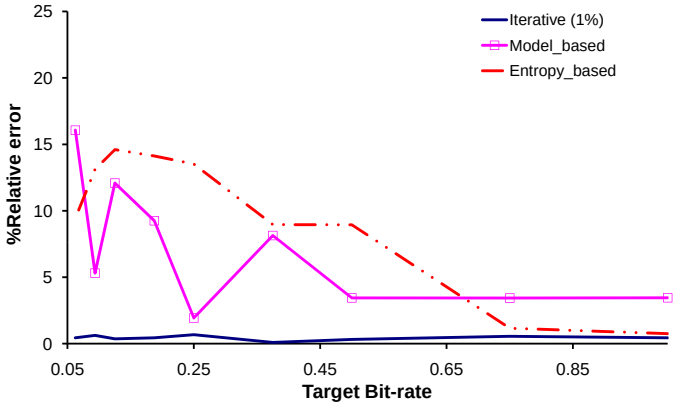


Figure 2.8: Bit rate accuracy for the GoldHill image

0.0625 to 0.125 bpp. This behavior is due to the model simplicity where there is no symbol differentiation in the insignificant coefficients set. In particular, the roots and members of lower trees (in the LTW encoder), which are very common symbols at these compression ratios, are not handled separately. However, at moderate and low compression ratios, the Model-based proposal is more accurate than the Entropy-based one.

In Figure 2.9 we measured the computational cost (in CPU cycles) of the proposed methods when coding the Goldhill test image. As can be seen, the Model-based method is the fastest one, due to the simplicity of computations required for issuing an estimation. The Entropy-based proposal is 3 times slower than the Model-based one, mainly due to the higher complexity of float type computations. In the case of iterative versions, we can observe that with a maximum 2% relative error, we obtain a very fast rate control

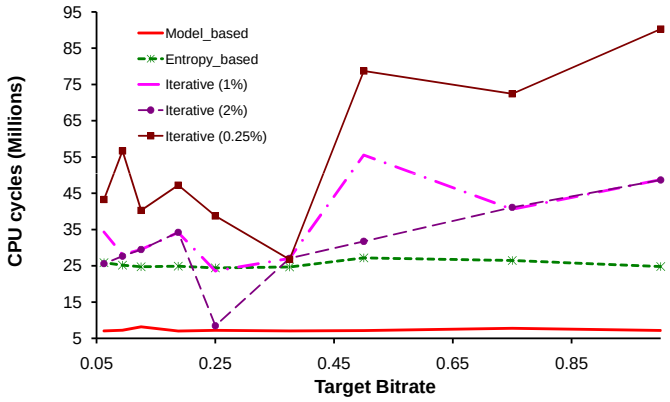


Figure 2.9: Complexity evaluation of different proposals for the GoldHill image

Bit rate	Entropy based	Model based	Iterative 1%	Entropy based	Model based	Iterative 1%
	Barbara(512x512)			Bike(2048x2560)		
0.0625	19.33	14.35	0.44	64.16	16.51	0.10
0.125	17.13	20.82	0.39	34.99	19.96	0.01
0.25	7.42	14.58	0.43	12.80	15.69	0.10
0.5	5.38	11.54	0.28	6.77	12.05	0.02
1	3.03	10.65	0.22	2.86	9.18	0.05
	Boat(512x512)			Cafe(2048x2560)		
0.0625	22.99	14.45	0.29	88.02	16.39	0.16
0.125	10.71	5.88	0.73	60.75	0.93	0.93
0.25	0.26	5.29	0.73	34.87	10.43	0.75
0.5	1.45	6.32	0.49	19.28	9.12	0.12
1	6.86	4.10	0.31	3.54	9.66	0.06
	Goldhill(512x512)			Lena(512x512)		
0.0625	9.42	16.06	0.44	14.11	14.20	0.48
0.125	14.60	12.08	0.36	2.19	12.59	0.00
0.25	13.51	1.92	0.67	12.04	10.62	0.15
0.5	8.94	3.44	0.31	14.14	9.78	0.16
1	0.75	3.44	0.43	16.34	2.51	0.09
	Mandrill(512x512)			Peppers(512x512)		
0.0625	24.16	2.83	0.87	27.88	15.43	0.09
0.125	18.87	17.72	0.04	3.56	11.45	0.24
0.25	11.60	1.28	0.26	12.28	9.48	0.00
0.5	13.20	6.79	0.36	16.29	8.42	0.34
1	7.07	4.83	0.37	6.03	8.74	0.09
	Woman(2048x2560)			Zelda(512x512)		
0.0625	1.68	18.04	0.18	16.60	11.86	0.58
0.125	2.36	6.28	0.64	28.14	11.98	0.31
0.25	0.73	10.20	0.54	31.23	12.58	0.11
0.5	3.70	9.47	0.21	4.09	11.87	0.03
1	5.44	8.40	0.65	268.53	3.50	0.36

Table 2.4: Bit rate accuracy for all test images (% error)

Bit rate	Entropy based	Model based	Iterative 2%	Iterative 1%	Iterative 0.25%
			Barbara(512x512)		
0.125	29.65	7.01	29.74	29.88	40.81
0.25	30.05	6.96	38.05	37.97	53.42
0.5	25.67	7.93	53.18	52.89	76.82
1	22.76	6.85	43.53	81.98	81.75
			Bike(2048x2560)		
0.125	515.08	135.66	449.86	709.75	715.62
0.25	501.36	136.91	524.78	864.79	863.45
0.5	528.78	137.38	670.16	1172.11	1167.62
1	523.34	141.02	957.60	960.01	952.69
			Boat(512x512)		
0.125	24.96	6.50	18.54	29.25	51.50
0.25	24.98	6.56	37.90	37.92	75.49
0.5	26.39	8.15	30.13	53.72	75.97
1	25.28	6.76	45.48	84.05	124.36
			Cafe(2048x2560)		
0.125	503.94	146.03	211.57	211.34	745.33
0.25	506.71	147.29	554.04	550.09	906.26
0.5	530.43	145.92	702.75	1218.59	1215.76
1	516.30	147.61	972.29	1782.59	1780.53
			Lena(512x512)		
0.125	26.61	7.36	22.93	37.50	30.28
0.25	25.46	6.59	26.59	37.73	43.73
0.5	26.81	7.05	30.07	58.96	58.91
1	26.65	6.93	52.81	53.44	52.59
			Mandrill(512x512)		
0.125	25.50	7.78	24.62	37.65	32.05
0.25	24.67	7.81	10.82	29.91	48.03
0.5	26.01	9.37	37.56	37.55	64.48
1	24.59	7.44	53.73	50.66	97.37
			Peppers(512x512)		
0.125	24.70	6.90	23.47	36.43	35.95
0.25	24.55	7.43	26.82	43.99	44.41
0.5	25.67	8.18	34.75	60.03	85.32
1	24.68	8.19	58.80	59.29	58.94
			Woman(2048x2560)		
0.125	505.87	141.82	733.82	735.96	1008.54
0.25	509.67	135.82	898.14	901.37	1255.52
0.5	523.85	136.72	696.95	694.69	691.23
1	503.98	138.41	978.82	975.30	1805.05
			Zelda(512x512)		
0.125	25.25	7.38	35.28	48.04	74.42
0.25	25.25	6.67	60.54	59.90	59.57
0.5	26.12	6.23	61.83	141.31	140.72
1	25.23	6.62	55.70	102.59	148.26

Table 2.5: Complexity evaluation of different proposals (Time in Millions of CPU cycles)

estimator (even faster than the Entropy-based). Also, we can state that the computational cost is not dependent on the target bit rate, although in the iterative versions the number of iterations may produce some deviations. The behavior is very similar in other non-Kodak test images as shown in Table 2.5.

2.5 Global performance evaluation

In order to analyze the impact of rate control proposals in the LTW encoder, we have performed several experiments comparing the obtained results with the original encoder. In addition to R/D performance, we also analyze other performance metrics like coding delay and memory consumption.

So as to perform a fair evaluation, we have chosen SPIHT (original version), JPEG 2000 (Jasper 1.701.0) and LTW version 1.1, since their source code is available for testing. The correspondent binaries were obtained by means of Visual C++ (version 6.0) compiler with the same project options. All the evaluated encoders have been tested on an Intel PentiumM Dual Core 3.0 GHz with 1 Gbyte RAM memory. The test images used in the evaluation were Lena (512x512), Barbara (512x512), GoldHill (512x512), Cafe (2560x2048) and Woman (2560x2048).

Table 2.6 shows the coding and decoding delay for all the encoders under evaluation. LTW_RC is the Model-based rate control version of LTW (described in Section 2.2). LTW_RCi is the iterative rate control version of LTW (described in Section 2.3) with a relative (% value) and an absolute (ABS suffix) rate control maximum allowed error that we experimentally have fixed to $\pm 0.04 bpps$. We discard the first rate control method (Entropy-based) due to its lower accuracy with respect to the Model-based one. Although not shown here, the behavior is very similar in all tested images as the coding delay is affected overall by the image size, as the coding delay differences between images with the same size were negligible (0.52 million CPU cycles on average).

As expected, JPEG 2000 is the slowest encoder and the original LTW is one of the fastest encoders. As shown in Table 2.6, the LTW_RC version does not introduce great overhead and it has an acceptable accuracy. If this rate control algorithm precision is not enough for the application, LTW_RCi is the candidate at the expense of increasing complexity. In general, all rate control versions of LTW are faster than SPIHT, specially the non-iterative version, LTW_RC, that performs the encoding process twice as fast as SPIHT.

Although it could be thought that the original LTW should be faster than the LTW_RC version due to rate control overhead, the results show just the

Codec/ bit rate	SPIHT	JPEG 2000	LTW Orig.	LTW-RC	LTW-RCi 2%	LTW RCi-ABS
	CODING Lena (512x512)					
0.125	20.82	158.79	9.75	**8.316**	20.25	10.03
0.25	29.12	161.92	14.09	**11.726**	27.08	13.42
0.5	45.82	167.14	22.46	**18.356**	62.07	40.74
1	79.56	175.64	41.46	**36.656**	75.75	38.61
	DECODING Lena (512x512)					
0.125	11.3	11.46	8.28	**6.77**	7.2	6.71
0.25	19.38	18.11	13.39	**10.8**	11.49	10.79
0.5	34.9	30.78	23.48	**18.84**	20.61	20.12
1	66.8	50.99	46.52	**41.11**	39.64	40.73
	CODING Cafe (2560x2048)					
0.125	469.73	4546.51	210.41	**192.06**	265.49	265.81
0.25	687.67	4527.09	299.78	**255.80**	670.28	327.44
0.5	1128.43	4591.08	459.48	**392.74**	947.77	950.7
1	2017.8	4736.99	733.21	**630.70**	1444.62	1443.28
	DECODING Cafe (2560x2048)					
0.125	232.53	234.10	182.97	168.09	**160.88**	161.01
0.25	397.98	362.96	295.56	**252.02**	264.03	260.03
0.5	745.69	593.92	491.96	**431.59**	442.87	945.90
1	1453.89	1040.39	830.87	**738.91**	764.77	763.98

Table 2.6: Execution time comparison of the coding process excluding DWT (time in millions of CPU cycles)

opposite. The reason for this behavior is based on the differences between the quantization process on both encoders. An image is encoded with the original LTW using a fixed value of the *rplanes* parameter (*rplanes*=2) and moving the Q parameter through a wide range. However, the LTW_RC version uses the estimated value for the *rplanes* parameter (from 2 to 7), limiting the value of the Q parameter to a shorter range. So, as more non-significant symbols are produced with this method, faster the algorithm becomes. However, as a 'side effect', the coding efficiency decreases slightly as we will see later.

In the iterative rate control versions, we have found two ways of defining the maximum allowed error: a relative or an absolute MAE (Maximum Allowed Error). The relative maximum error shows non-linear behavior, since a rate control precision of 1% is not the same at 2 bpp than at 0.125 bpp. For very low bit rates, achieving an accuracy of 1% has no effects upon R/D performance. The maximum absolute error is fixed independently of the tar-

get bit rate, so it produces different relative errors at different bit rates. It is important to take into account that proposed rate control methods have an average precision error around 5% at 1 bpp and 9% at 0.125 bpp, as shown previously.

Figure 2.10: Lena compressed at 0.125 bpp and 0.0625 bpp - (a,e) LTW_RCi_2%; (b,f) SPIHT; (c,g) JPEG 2000; and, (d) Original (not compressed)

Table 2.7 shows the R/D evaluation of the proposed encoders. In general, the original LTW obtains very good performance results, specially in the Lena and Woman test images. The rate control versions of LTW have slightly lower PSNR results than SPIHT and JPEG 2000, being the LTW_RCi at 2% the one that shows better R/D behavior. Table 2.7 also shows the absolute

bit rate error in brackets for all LTW rate control versions. The lower performance of the rate control algorithm versions is mainly due to the achieved final bit rate that is always lower than the target one (obviously, the more accuracy, the better R/D performance). Note that original LTW has been tuned to provide the exact bit rate.

Codec/ bit rate	SPIHT	JPEG 2000	LTW Orig.	LTW RC	LTW-RCi 2%	LTW-RCi ABS
	Lena(512x512)					
0.125	31.10	30.81	**31.28**	30.59(−0.016)	31.06(−0.001)	30.59(−0.016)
0.25	34.15	34.05	**34.33**	33.65(−0.027)	34.05(−0.004)	33.65(−0.026)
0.5	37.25	37.26	**37.39**	36.76(−0.049)	37.15(−0.008)	37.08(−0.011)
1	40.46	40.38	**40.55**	40.34(+0.035)	40.20(−0.001)	40.34(+0.035)
	Cafe(2560x2048)					
0.125	20.67	20.74	**20.76**	20.63(−0.001)	20.63(−0.001)	20.63(−0.001)
0.25	23.03	23.12	**23.24**	22.60(−0.026)	23.08(+0.002)	22.60(−0.027)
0.5	26.49	26.79	**26.85**	26.04(−0.046)	26.53(−0.006)	26.53(−0.007)
1	31.74	**32.03**	32.02	30.89(−0.097)	31.64(−0.010)	31.64(−0.014)
	Barbara(512x512)					
0.125	24.86	**25.25**	25.21	24.30(−0.026)	25.04(+0.002)	24.21(−0.027)
0.25	27.58	**28.33**	28.04	27.09(−0.036)	27.76(−0.001)	27.09(−0.036)
0.5	31.39	**32.14**	31.72	30.80(−0.055)	31.47(−0.002)	31.34(−0.012)
1	36.41	**37.11**	36.67	35.61(−0.101)	36.39(−0.004)	36.25(−0.021)
	GoldHill(512x512)					
0.125	28.48	28.35	**28.59**	28.15(−0.015)	28.45(−0.001)	28.15(−0.015)
0.25	30.56	30.51	**30.66**	30.48(−0.001)	30.48(−0.001)	30.48(−0.001)
0.5	33.13	33.19	**33.29**	32.84(−0.028)	33.13(+0.006)	32.84(−0.028)
1	36.55	36.53	**36.71**	36.17(−0.046)	36.45(−0.000)	36.45(+0.001)
	Woman(2560x2048)					
0.125	27.33	27.33	**27.51**	27.19(−0.007)	27.30(−0.003)	27.19(−0.007)
0.25	29.95	29.98	**30.15**	29.45(−0.028)	30.02(+0.004)	29.45(−0.027)
0.5	33.59	33.62	**33.82**	32.94(−0.050)	33.55(−0.002)	33.54(−0.001)
1	38.27	38.43	**38.52**	37.52(−0.098)	38.32(−0.003)	38.23(−0.002)

Table 2.7: PSNR (dB) with different bit rate and coders

In Table 2.8, memory requirements of the encoders under test are shown. The original LTW needs only the amount of memory to store the source image. LTW_RC also requires an extra 1.2 KB, basically used to store the histogram of significant symbols needed to accomplish the Model-based rate

control algorithm. On the other hand, the LTW_RCi version requires twice the memory space as LTW and LTW_RC, since at each iteration the original wavelet coefficients must be restored to avoid a new DWT time consuming procedure. SPIHT requires nearly the same memory as LTW_RCi, and JPEG 2000 needs three times the memory of LTW (results obtained with Windows XP task manager, peak memory usage column).

Codec/image	SPIHT	JPEG 2000	LTW Orig.	LTW-RC	LTW-RCi
Lena	3228	4148	**2048**	2092	3140
Cafe	46776	65832	**21576**	21632	42188

Table 2.8: Memory requirements for all evaluated encoders (KB)

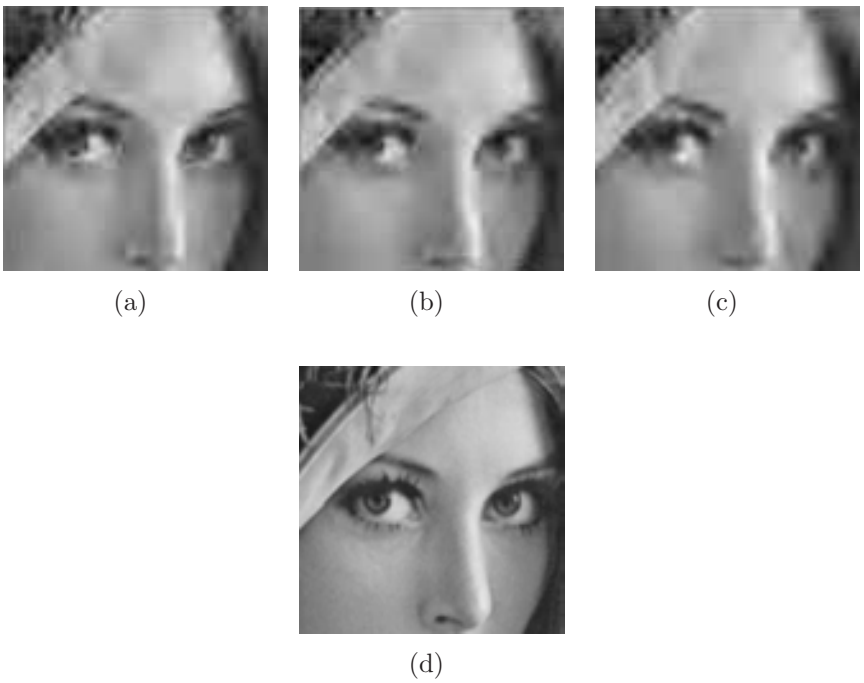
Figure 2.10 shows the Lena test image (512x512) compressed at 0.125 bpp and 0.0625 bpp with (a,e) LTW_RCi, (b,f) SPIHT and (c,g) JPEG 2000. Although the SPIHT encoder is, in terms of PSNR, slightly better than LTW_RCi and JPEG 2000, the subjective test does not show perceptible differences between reconstructed versions of the Lena image. At 0.0625 bpp the difference in PSNR between LTW_RCi or SPIHT and JPEG 2000 is nearly 0.5 dB. This difference is only visible if we carry out a zoom over the eye zone as can be seen in Figure 2.11. Both SPIHT and LTW_RCi have similar behavior.

2.5.1 Optimized encoders

All LTW encoder versions were developed finding the optimizations for maximizing R/D performance, so its software code is not optimized, just like the JPEG 2000 reference software. However, we have compared its performance with respect to a full optimized implementation of JPEG 2000: Kakadu [45], in order to evaluate whether a full optimization of LTW is worth the effort. For that purpose, we have used the 5.2.5 version, one of the latest Kakadu versions which is fully optimized including multi-thread and multi-core hardware support, processor intrinsics like MMX/SSE/SSE2/SIMD and fast multicomponent transforms.

As shown in Figure 2.12, LTW_RC is a very fast encoder even though not being fully optimized. The speed of LTW_RC lies in the simple engine coding model. LTW_RC is on average 1.2 times faster than Kakadu-5.2.5.

Regarding memory requirements, LTW_RC needs only the amount of memory to store the source image and 1.2 KB to store the model histogram as mentioned before, while Kakadu memory requirements are independent of



(a) (b) (c)

(d)

Figure 2.11: Zoom over eye zone in reconstructed Lena at 0.0625 bpp - (a) LTW_RCi_2%; (b) SPIHT; (c) JPEG 2000; and, (d) Original (not compressed)

the image size due to its DWT block-based implementation and they are on average 1660 KB.

In terms of R/D, there are slight differences between all codecs as Table 2.9 shows. For images with lots of high frequency components, like Barbara, Kakadu provides a better PSNR than LTW, but for images like Lena or Woman, LTW outperforms Kakadu.

2.6 Conclusions

In this chapter, we have presented three different rate control tools suitable for non-embedded wavelet-based encoders. We have implemented them over the LTW encoder in order to evaluate their behavior and compare the performance results with SPIHT and JPEG 2000 encoders in terms of R/D, execution time and memory consumption. Furthermore, we have shown that we can add rate control functionality to non-embedded wavelet encoders without a significant increase in complexity and little performance loss. Among the proposed simple rate control tools, the Model-based rate control algorithm implemented in the LTW_RC proposal is the one that exhibits the best trade-

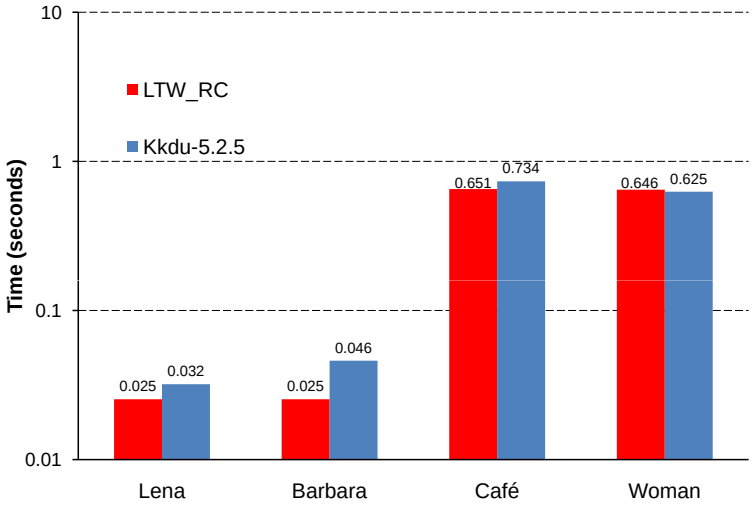


Figure 2.12: Execution time comparison (end-to-end) of the coding process at 0.5 bpp

off between R/D performance, coding delay (twice as fast as SPIHT and 8.8 times faster than JPEG 2000) and overall memory usage (similar to the original LTW).

Also, we have compared the LTW_RC coder with a highly optimized version of JPEG 2000 (Kakadu), resulting competitive in terms of coding delay (up to 1.8 times faster than Kakadu for medium size images) with slightly lower R/D performance. So, a full optimization process will make LTW_RC even faster and with lower memory requirements. These optimizations will be mainly focused on the DWT coding step by using fast and low memory demanding DWT techniques like line-based [70] or block-based ones and exploiting the parallel capabilities of modern processors (like multithreading and SIMD instructions).

Codec/ bit rate	KAKADU 5.2.5	LTW Orig.	LTW RC
	Lena(512x512)		
0.125	30.95	**31.28**	30.59(−0.0157)
0.25	34.11	**34.33**	33.65(−0.0266)
0.5	37.30	**37.39**	36.76(−0.0489)
1	40.40	**40.55**	40.34(+0.0350)
	Cafe(2560x2048)		
0.125	**20.78**	20.76	20.63(−0.0012)
0.25	23.15	**23.24**	22.60(−0.0261)
0.5	26.84	**26.85**	26.04(−0.0456)
1	**32.03**	32.02	30.89(−0.0967)
	Barbara(512x512)		
0.125	**25.24**	25.21	24.30(−0.0264)
0.25	**28.36**	28.04	27.09(−0.0363)
0.5	**32.17**	31.72	30.80(−0.0552)
1	**37.15**	36.67	35.61(−0.1008)
	Woman(2560x2048)		
0.125	27.36	**27.51**	27.19(−0.0070)
0.25	30.05	**30.15**	29.45(−0.0280)
0.5	33.64	**33.82**	32.94(−0.0505)
1	38.43	**38.52**	37.52(−0.0980)

Table 2.9: PSNR (dB) comparison between Kakadu and LTW_RC

Chapter 3

Sign Coding

Contents

The energy of a wavelet transform coefficient is restricted to non-negative real numbers, but the coefficients themselves are not, and they are defined by both a magnitude and a sign. Shapiro stated in [94] that a quantized transform coefficient is equally likely to be positive or negative and thus one bit must be used to transmit the sign. In recent years, several authors have begun to use context modeling for sign coding [114][104][24].

In [24] sign coding is examined in detail in the context of an embedded wavelet image coder. The paper shows that a PSNR improvement up to 0.7 dB is possible when sign entropy coding and a new extrapolation technique to improve the estimation of insignificant coefficients are combined. However, the contribution of sign coding to the PSNR improvement is only up to 0.4 dB for Crowd image. As reported in the paper, the PSNR improvement for Lena image is up to 0.14 dB at low compression ratios and up to 0.36 dB for Barbara image at medium compression ratios.

In [104] the Embedded Block Coding with Optimized Truncation of the embedded bit-streams (EBCOT), core coding tool of the JPEG 2000 standard, encodes the sign of wavelet coefficients using context information from the sign of horizontal and vertical neighbor coefficients (N, S, E, W directions).

In [114] a high order context modeling encoder is presented. In this coder, the sign and the textures share the same context modeling. This model is based on a different neighborhood for the HL, LH and HH wavelet subbands. For the HH subband, an inter-band prediction is used besides the intra-band prediction used by the HL and LH subbands.

In this chapter, we perform a study about the usefulness of sign coding techniques for fast non-embedded image encoders, showing the benefits and drawbacks of adding sign coding capability to the encoder. For that purpose, we use the LTW encoder as the reference framework for our study, with results useful for other non-embedded encoders.

3.1 Context-based sign coding approach

In the former wavelet image encoders, sign coding of wavelet coefficients was not considered because those coefficients located at the high frequency subbands form a zero-mean process, and therefore are equally likely positive as negative.

Schwartz, Zandi and Boliek were the first authors to consider sign coding, using the sign of one neighboring pixel in their context modeling algorithm [88]. The main idea behind this approach is to find correlations along and across edges.

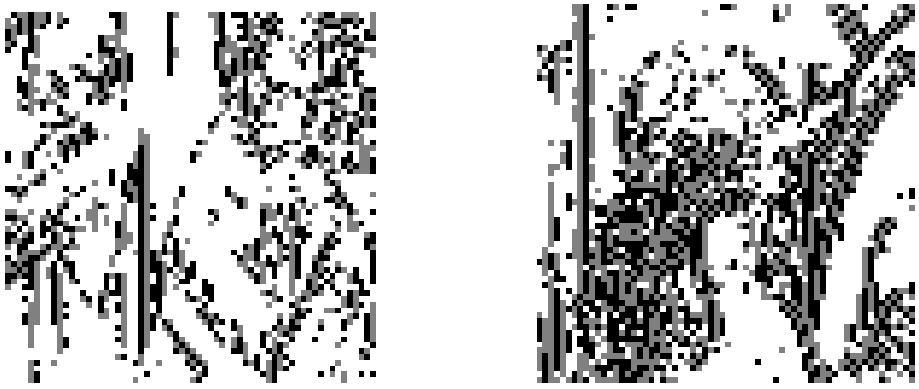
The HL subbands of a multi-scale 2-D wavelet decomposition are formed from low-pass vertical filtering and high-pass horizontal filtering. The high-pass filtering detects vertical edges and so the HL subbands contain mainly vertical edge information. Oppositely defined are the LH subbands that contain primarily horizontal edge information.

As explained in [24], given a vertical edge in an HL subband, it is reasonable to expect the transform coefficients along this edge to be positively correlated. This is because vertical correlation often remains very high along vertical edges in images. When a low-pass filter is applied along the image columns, it results in a series of similar rows, as elements in a row tend to be very similar to elements directly above or below due to the high vertical correlation. Subsequent high-pass filtering along similar rows is expected to yield vertically correlated transform coefficients. In Figure 3.2, vertical edges exist along the vertical picture frame edge. The pixels along this edge remain highly correlated in the vertical direction. Hence it is predicted that the associated strong edge in the HL subband will possess positive correlation in the vertical direction along this edge. Neighboring coefficients along the edge are considered valuable context information, and are expected to have the same sign as the coefficient being coded. This is independent of the type of wavelet filter employed.

It is also important to consider correlation across edges. In this case, the nature of the correlation is directly affected by the structure of the high pass filter. For Daubechies' 9/7 filters, wavelet coefficient signs are strongly negatively correlated across edges. This strong negative correlation can be seen in Figure 3.2, where lighter colored wavelet coefficients are positive, and darker colored wavelet coefficients are negative. An explanation for negative correlation across edges comes from the theory of zero crossings and edge detection. Most edge detectors utilize a first or second order derivative of a Gaussian-like function to identify edges. The signal is smoothed, and then extrema in the first derivative, corresponding to zero crossings of the second derivative, are classified as edges [56]. As it is well known, wavelet decomposition used in Daubechies' 9/7 filter is very similar to a second derivative of a Gaussian, so it is expected that wavelet coefficients will change sign as the edge is crossed. Although the discrete wavelet transform involves subsampling, the subsampled coefficients remain strongly negatively correlated across edges. In this manner, when the wavelet coefficient in question is optimally predicted as a function of its across-edge neighbors (e.g. left and right neighbors in HL subbands), the optimal prediction coefficients are negative, indicating an expected sign change. This conclusion is general for any wavelet with a shape similar to a second derivative of a Gaussian.

In Figure 3.1 we plot the spatial distributions of signs in the HL subband

of two popular test images, Barbara and Lena. The visible sign structures suggest that the sign bits of wavelet coefficients are compressible.



(a) Sign map for Barbara (b) Sign map for Lena

Figure 3.1: Sign patterns in the HL subband for Barbara (left) and Lena (right). Black for negative signs, grey for positive signs and white for non-significant coefficients

In order to identify the neighborhood with the highest sign correlation in the LTW encoder, we have defined a neighboring set with a maximum distance of two, that is W, N, WW, NN, NW and NN-W (see Figure 3.3). Remark that LTW uses a Morton order (Z-order)[64] in the coding stage, so no information is available about S (South) and E (East) neighbors. This is a restriction when looking for sign correlation among the neighborhood shared by most of the non-embedded encoders. All the neighbors have three possible sign values, positive (+), negative (-) or non-significant (*), but the current coefficient being evaluated has only two possible signs, positive (+) or negative (-). This forms a total of $2 * 6^3$ neighbor sign combinations. To estimate sign correlation in a practical way, we have performed a 6-level DWT decomposition of the source image (Lena) and after that we have applied a very low quantization process to the resulting wavelet coefficients ($rplanes = 2$).

In order to build an histogram with the occurrences of the neighbor sign combinations, we follow a raster order inside each subband to determine the signs of the current significant (non-zero) coefficient and its neighbors. Once we have obtained the probability distribution of sign combinations, we have dismissed several neighbors trying to group probabilities as much as possible. This process has been done independently for each subband type and for each wavelet decomposition level. After compacting the probability distribution function of sign combinations, we noticed that sign neighborhood correlation depends on the subband type (HL,LH,HH) as Deever assesses in [24]. So,

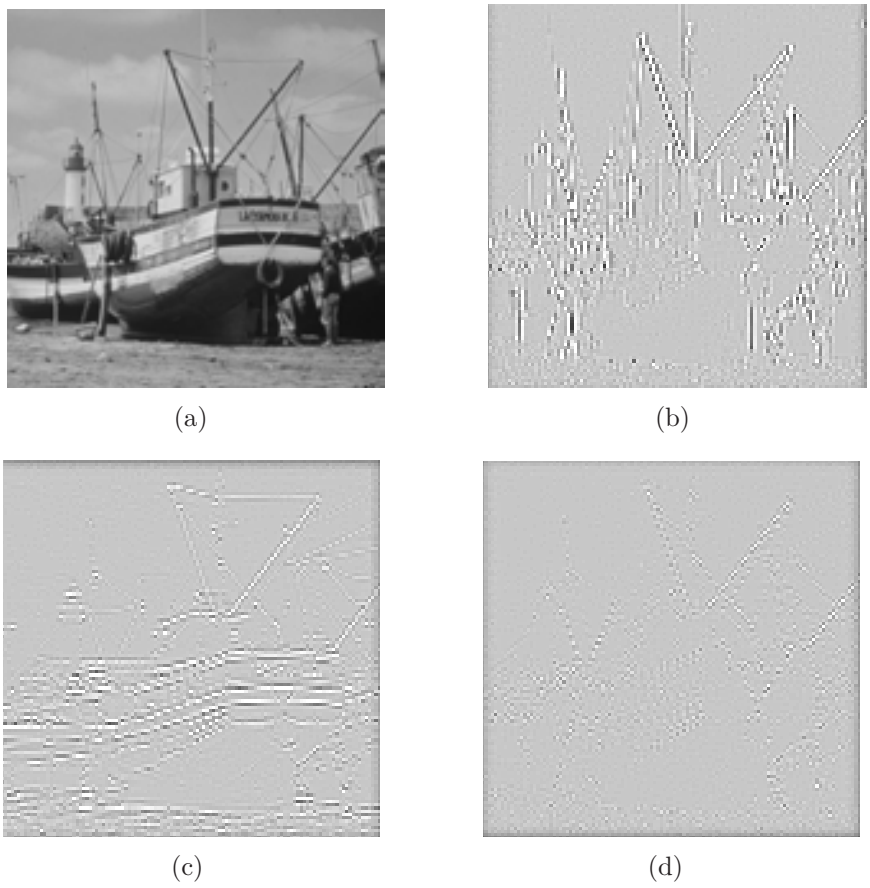


Figure 3.2: Boat image and HL(b), LH(c) and HH(d) wavelet subbands

for the HL subband, the neighbors, whose sign is more correlated with the sign of the current coefficient are N, NN and W. Taking symmetry into account, for the LH subband, those neighbors are W, WW, and N. For the HH subband they are N, W, and NW, exploiting the correlation along and across the diagonal edges. At this point, we have a maximum of 3^3 neighbor sign combinations for each subband type.

In table 3.1 we show neighbors sign probabilities for the HL subband in the Lena image at the last wavelet decomposition level. As can be seen, the probability that the current coefficient (C) sign was positive when its N, NN, and W were positive is aproximately 20%. If we observe with more detail, we can see that there is a high probability when the N, and NN neighbors have the same sign and the W neighbor has the opposite sign.

After having analyzed all the possible combinations of neighbors for each subband type and for each wavelet decomposition level, we could make a

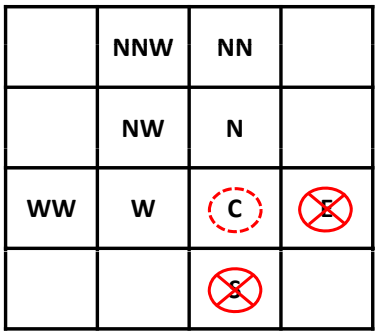


Figure 3.3: Sign intraband neighborhood

Level	C	N	NN	W	Occurrences	Total occurrences	%Probability
6	+	+	+	+	13	64	20.31
6	+	+	+	-	8		12.50
6	-	-	-	+	8		12.50
6	-	+	+	+	6		9.38
6	-	-	+	+	6		9.38
6	+	+	-	-	4		6.25
6	+	-	-	-	4		6.25
6	-	+	+	-	4		6.25
6	+	-	+	+	2		3.13
6	-	+	-	-	2		3.13
6	-	-	+	-	2		3.13
6	-	-	-	-	2		3.13
6	+	-	+	-	1		1.56
6	+	-	-	+	1		1.56
6	-	+	-	+	1		1.56

Table 3.1: Neighbor's sign probability distribution for the HL subband in the Lena image at the last DWT decomposition level

prediction of the current coefficient sign. As an example in table 3.2 we show the sign predictions for the HL subband. With the prediction obtained for the current coefficient ($\hat{SC}_{i,j}[k]$) based on the neighborhood sign information (context info), what we are going to encode is the correctness of this sign prediction. That is, a binary valued symbol from $\hat{SC}_{i,j}[k] \cdot SC_{i,j}$. In order to compress this binary valued symbol as much as possible, we have used two contexts for every subband type. So as to minimize the zero order entropy of both contexts, we have distributed all sign coding predictions from

the neighborhood between them. The selection criteria is to isolate in one context those neighbors combinations with the highest correctness prediction probability and highest number of occurrences. The remaining combinations are grouped into the other context. However, there are certain combinations with low correctness probability but with a great amount of occurrences, so we have to heuristically determine the convenience of including them in the first context or not.

Combination (k)	**N**	**NN**	**W**	**Prediction** ($\hat{SC}_{i,j}[k]$)
0	*	*	*	+
1	*	*	+	+
2	*	*	-	+
3	*	+	*	+
4	*	+	+	+
5	*	+	-	+
6	*	-	*	-
7	*	-	+	-
8	*	-	-	-
9	+	*	*	+
10	+	*	+	+
11	+	*	-	+
12	+	+	*	+
13	+	+	+	+
14	+	+	-	+
15	+	-	*	+
16	+	-	+	+
17	+	-	-	+
18	-	*	*	+
19	-	*	+	+
20	-	*	-	+
21	-	+	*	+
22	-	+	+	+
23	-	+	-	+
24	-	-	*	+
25	-	-	+	-
26	-	-	-	+

Table 3.2: Sign prediction for the HL subband

3.2 Evaluation

In this section we analyze the behavior of the sign coding when implemented on the original LTW. This new encoder implementation is called S-LTW. For further evaluation we will also compare the S-LTW encoder versus JPEG 2000 (Jasper 1.701.0) and SPIHT (Spiht 8.01) in terms of R/D and coding and decoding delay. All the evaluated encoders have been tested on an Intel PentiumM Dual Core 3.0 GHz with 1 Gbyte RAM memory. The correspondent encoders binaries were obtained by means of Microsoft Visual C++ (2005 version) compiler with the same project options.

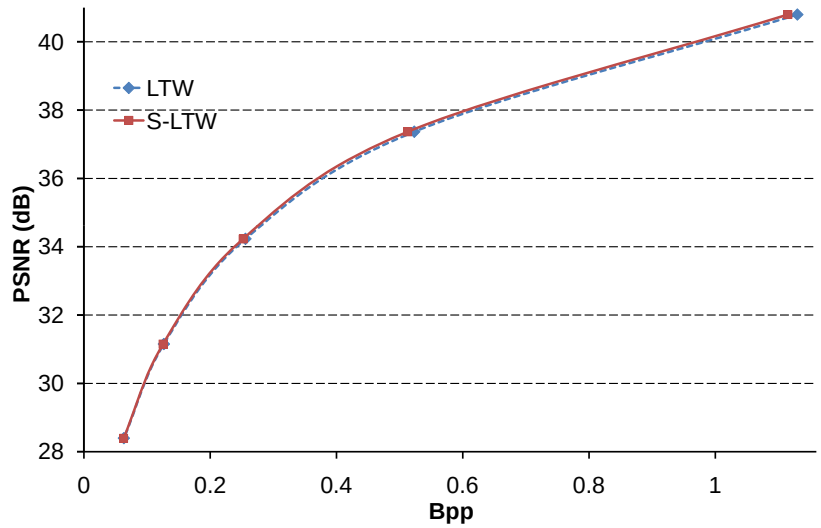
In Table 3.4 we show the relative compression gain with respect to the original LTW due only to the sign coding capability in several test images. As we can see, the maximum compression gain is 17.35% for Barbara at 1 bpp. For further evaluation in Table 3.3, the maximun, minimum and average percentage (%) gains for the whole Kodak set is presented and as can be seen it presents a similar behavior as in the test images.

Bpp	Maximum % gain	Minimum % gain	Average % gain
1	14.47	4.76	9.08
0.5	13.25	3.52	8.11
0.25	16.53	2.38	7.32
0.125	18.45	0.72	7.67
0.0625	22.68	0	9.25

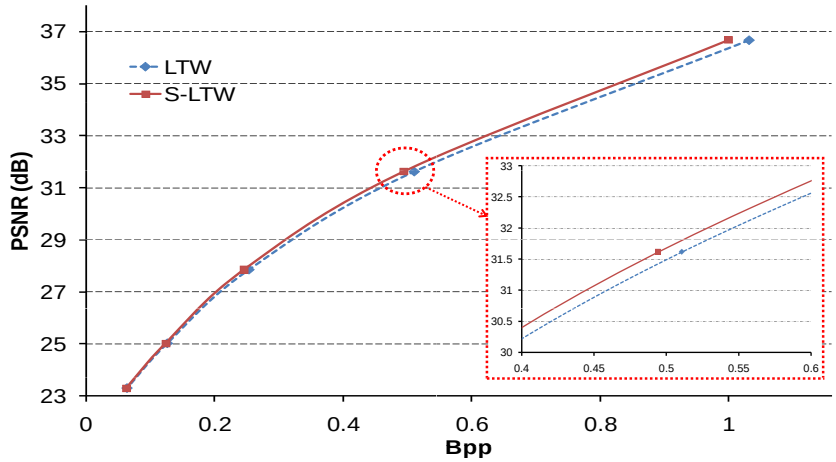
Table 3.3: Summary of sign compression gain for Kodak image set at different bit rates

In order to obtain the real compression gain of the new codec, we have evaluated this new encoder version (S-LTW) in terms of R/D and global compression rate gain. In Figure 3.4 we show the R/D performance of the new codec when compared to the original LTW for the Lena and Barbara images at diferent compression ratios. As can be seen, S-LTW has slightly better PSNR behavior than the original LTW, specially for high textured images like Barbara, where there are more significant coefficients. Specifically, the PSNR improvement is up to 0.1 dB for Lena image and up to 0.28 dB for Barbara image at low compression ratios.

In Figure 3.5 we show the R/D increment when comparing in pairs, on the one hand, the original LTW versus JPEG 2000 and SPIHT, and on the other hand, S-LTW versus JPEG 2000 and SPIHT. As shown, there is an increase in the PSNR difference between the new S-LTW encoder and SPIHT, and



(a) Lena



(b) Barbara

Figure 3.4: PSNR (dB) comparison between LTW and S-LTW for Lena and Barbara test images

regarding JPEG 2000, we can see that now S-LTW has a lesser loss in PSNR than the original LTW.

As could be expected, the use of higher context modeling in the arithmetic encoder in the S-LTW encoder, implies a higher computational cost in the coding and decoding process, as shown in Table 3.5. However, this increase in the coding time still keeps S-LTW competitive. At high compression ratios,

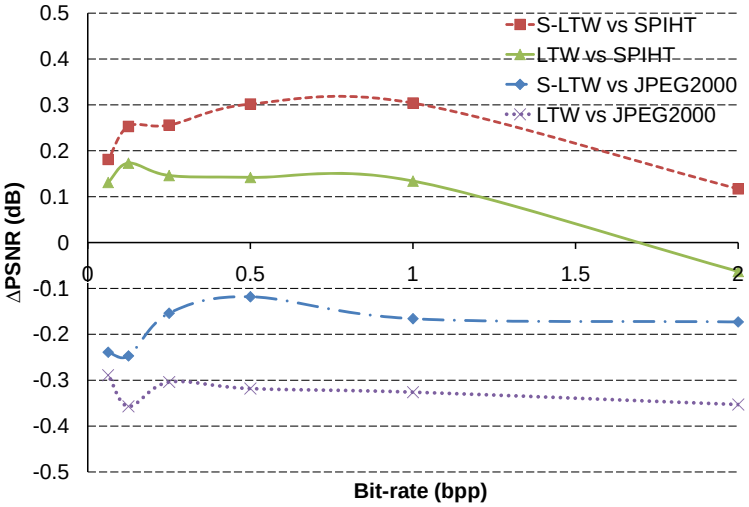


Figure 3.5: PSNR-Gain for Bike image

SPIHT has a similar behavior as S-LTW, but at medium and low compression ratios, S-LTW still remains faster. As shown, the behavior is similar in all tested images, and better results were obtained in high textured images like Cafe.

Bit rate (bpp)	Significant Coefficients	Bits Gain	%Gain	Significant Coefficients	Bits Gain	%Gain
	Barbara (512x512)			Bike (2048x2560)		
2	95299	12768	13.40	1887252	169864	9.00
1	45740	7936	**17.35**	855266	115200	13.47
0.5	22331	3648	16.34	412212	64424	15.63
0.25	10484	1520	14.50	198943	30472	15.32
0.125	4343	304	7.00	91767	11992	13.07
0.0625	2180	72	3.30	46543	4664	10.02
	Cafe (2048x2560)			GoldHill (512x512)		
2	1723583	154160	8.94	117713	4400	3.74
1	866839	80240	9.26	46563	2248	4.83
0.5	426670	29560	6.93	21777	960	4.41
0.25	200617	9408	4.69	11106	456	4.11
0.125	107553	4216	3.92	4747	184	3.88
0.0625	42890	1488	3.47	2126	24	1.13
	Lena (512x512)			Mandrill (512x512)		
2	109014	3696	3.39	105351	3128	2.97
1	51113	4032	7.89	48326	1232	2.55
0.5	20886	2328	11.15	22223	240	1.08
0.25	10038	880	8.77	11487	32	0.28
0.125	4724	256	5.42	4088	40	0.98
0.0625	2178	64	2.94	2246	32	1.42
	Peppers (512x512)			Woman (2048x2560)		
2	131399	2240	1.70	1964238	159952	8.14
1	57008	1880	3.30	909513	93192	10.25
0.5	19902	1512	7.60	443131	43472	9.81
0.25	9647	720	7.46	210433	17656	8.39
0.125	4717	328	6.95	106977	7416	6.93
0.0625	2189	88	4.02	42272	2056	4.86

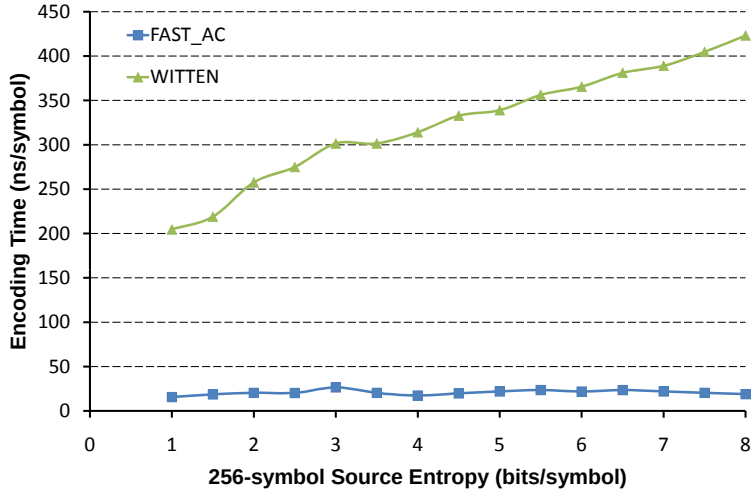
Table 3.4: Relative bit rate compression gain at different bit rates for several test images

Bit rate (bpp)	JPEG 2000	SPIHT	LTW Orig.	S-LTW	JPEG 2000	SPIHT	LTW Orig.	S-LTW
	CODING Barbara (512x512)				DECODING Barbara			
1	0.080	0.042	**0.037**	0.045	0.022	0.031	**0.021**	0.029
0.5	0.076	0.026	**0.022**	0.026	0.012	0.016	**0.012**	0.015
0.25	0.074	0.018	**0.013**	0.016	0.007	0.009	**0.006**	0.008
0.125	0.073	0.014	**0.010**	0.011	0.005	0.005	**0.003**	0.005
0.0625	0.072	0.011	**0.008**	0.008	0.003	0.003	**0.002**	0.003
	CODING Bike (2048x2560)				DECODING Bike			
1	2.423	0.915	**0.649**	0.787	0.481	0.676	**0.437**	0.568
0.5	2.397	0.539	**0.354**	0.428	0.282	0.343	**0.230**	0.300
0.25	2.371	0.314	**0.220**	0.248	0.170	0.183	**0.131**	0.161
0.125	2.312	0.218	**0.136**	0.159	0.109	0.103	**0.072**	0.093
0.0625	2.337	0.172	**0.105**	0.118	0.078	0.063	**0.049**	0.061
	CODING Cafe (2048x2560)				DECODING Cafe			
1	2.623	0.920	**0.647**	0.795	0.576	0.674	**0.438**	0.580
0.5	2.543	0.521	**0.381**	0.460	0.329	0.349	**0.249**	0.323
0.25	2.507	0.323	**0.224**	0.269	0.201	0.188	**0.135**	0.176
0.125	2.518	0.221	**0.158**	0.184	0.130	0.106	**0.086**	0.112
0.0625	2.509	0.172	**0.105**	0.119	0.056	0.062	**0.048**	0.062
	CODING GoldHill (512x512)				DECODING GoldHill			
1	0.082	0.045	**0.039**	0.047	**0.022**	0.033	0.024	0.032
0.5	0.078	0.028	**0.023**	0.027	**0.012**	0.018	0.013	0.017
0.25	0.076	0.022	**0.015**	0.017	0.008	0.009	**0.007**	0.010
0.125	0.075	0.016	**0.011**	0.011	0.005	0.005	**0.004**	0.005
0.0625	0.074	0.012	**0.007**	0.008	0.004	0.003	**0.002**	0.003
	CODING Lena (512x512)				DECODING Lena			
1	0.072	0.043	**0.042**	0.051	**0.022**	0.032	0.026	0.034
0.5	0.069	0.026	**0.022**	0.025	0.013	0.017	**0.012**	0.015
0.25	0.066	0.018	**0.014**	0.016	0.008	0.009	**0.006**	0.008
0.125	0.065	0.014	**0.010**	0.011	0.005	0.005	**0.004**	0.005
0.0625	0.064	0.011	**0.007**	0.008	0.004	0.003	**0.002**	0.003
	CODING Mandrill (512x512)				DECODING Mandrill			
1	0.103	0.044	**0.040**	0.050	**0.019**	0.032	0.025	0.033
0.5	0.099	0.027	**0.024**	0.028	**0.011**	0.018	0.013	0.018
0.25	0.097	0.019	**0.016**	0.019	**0.007**	0.009	0.008	0.010
0.125	0.095	0.014	**0.010**	0.012	**0.004**	0.005	**0.004**	0.005
0.0625	0.095	0.011	**0.008**	0.009	**0.003**	**0.003**	**0.003**	0.004
	CODING Woman (2048x2560)				DECODING Woman			
1	2.423	0.944	**0.668**	0.808	0.573	0.687	**0.444**	0.582
0.5	2.345	0.546	**0.375**	0.459	0.259	0.364	**0.242**	0.322
0.25	2.345	0.325	**0.231**	0.273	0.161	0.190	**0.137**	0.177
0.125	2.337	0.219	**0.157**	0.185	0.100	0.105	**0.084**	0.110
0.0625	2.337	0.169	**0.109**	0.123	0.073	0.063	**0.051**	0.064

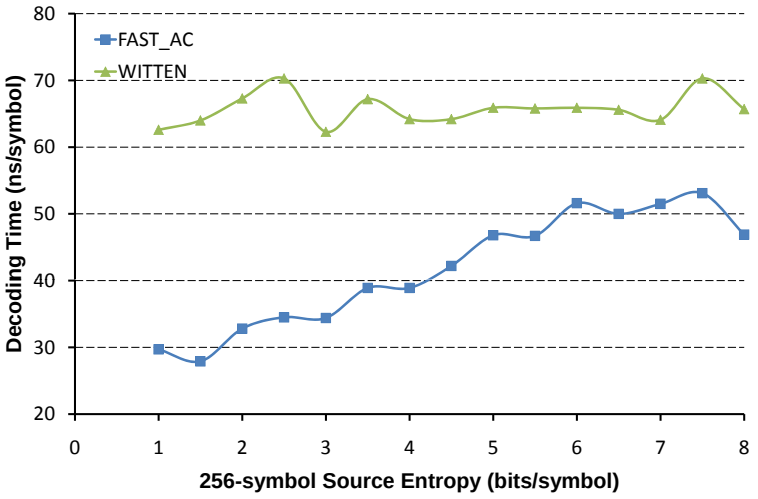
Table 3.5: Coding and decoding delay (time in seconds)

3.2.1 Fast arithmetic encoder evaluation

In order to compensate the coding speed loss we have changed the arithmetic encoder stage by a fast arithmetic encoder [85]. So in this subsection we will evaluate both S-LTW encoder versions: S-LTW (uses Witten's arithmetic encoder[111]) and S-LTW-Fast (uses Said's fast arithmetic encoder [85]).



(a) Encoding



(b) Decoding

Figure 3.6: Coding and Decoding delay comparison between Witten's and Said's arithmetic encoders

Firstly, we have tested both arithmetic encoders ([111] and [85]) inde-

pendently using a test aplication, where given a number of data symbols it defines several values of source entropy, and for each value it generates millions of pseudo-random source samples. The times to encode and decode this data are measured, and it finally compares the decoded with the original to make sure the code is correct.

In Figure 3.6 we show the test results when encoding a 256 symbol source. As shown, the encoding speed of Said's fast arithmetic encoder is up to 22 times faster than Witten's. Regarding the decoding process, the fast arithmetic encoder is up to 2 times faster than Witten's.

Now we will compare the S-LTW and the S-LTW-Fast encoders in terms of coding/decoding delay and compression performance. As shown in Figure 3.7, both S-LTW encoder versions have similar behavior, being S-LTW slightly better than S-LTW-Fast, mainly due to the lower compression rate achieved by the fast arithmetic encoder.

Regarding coding and decoding delay, the S-LTW-Fast version is 28% faster on average in the coding process and 47% faster on average in the decoding process than the S-LTW version as shown in Figure 3.8.

3.3 Conclusions

In this chapter we have presented a study about sign coding for non-embedded image encoders. We propose a simplified context model formation that is oriented to maximize the successful prediction of the sign for every non-zero wavelet coefficient. The prediction result is encoded with an adaptive arithmetic encoder to compact the sign as much as possible. We have implemented it over the LTW encoder in order to evaluate the sign context model behavior. The new S-LTW encoder proposed has slightly better performance in R/D (up to 0.28 dB), with the improvements being greater at low and medium compression ratios. Regarding coding delay, the use of six contexts (two contexts for each subband type) implies an increase in the coding and decoding delay (around 17% on average), and still competitive against SPIHT and JPEG 2000. In order to compensate for the coding speed loss, we have changed the arithmetic coder stage. The new arithmetic encoder [85] is 28% faster on average when coding and 47% faster on average in the decoding process with minimum compression performance loss (3% on average).

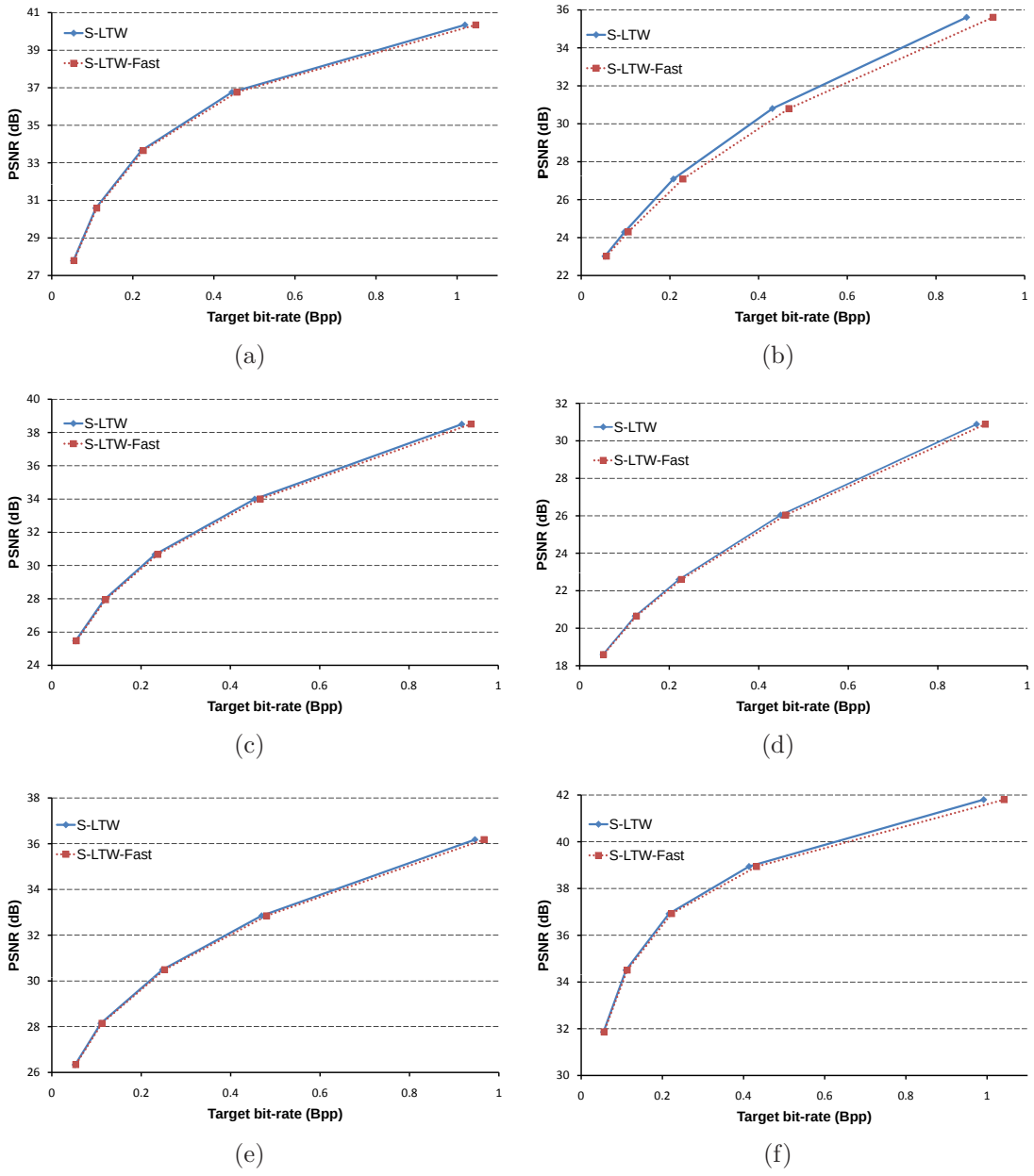


Figure 3.7: PSNR comparison between S-LTW and S-LTW-Fast for a) Lena; b) Barbara; c) Boat; d) Cafe; e) GoldHill; and, f) Zelda test images

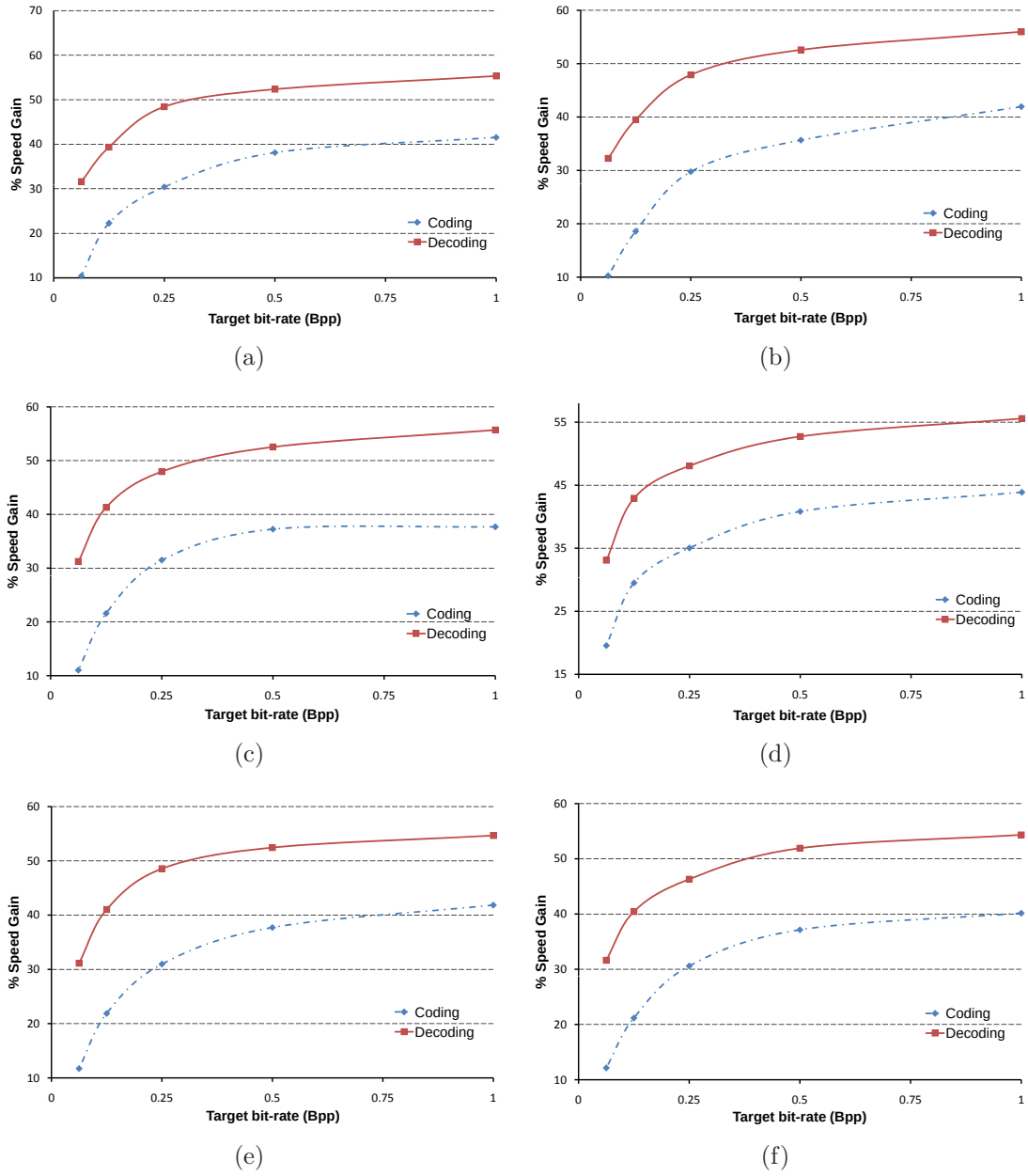


Figure 3.8: Coding and Decoding delay Gain (%) with S-LTW-Fast for a) Lena; b) Barbara; c) Boat; d) Cafe; e) GoldHill; and, f) Zelda test images

Chapter 4

Enhanced LTW

Contents

In this chapter, we propose an enhanced version of the LTW encoder (E-LTW) that includes the sign coding tool presented in Section 3.1 and a new precise rate control method. These new features will improve the R/D behavior, providing non-embedded encoders with a highly accurate rate control tool.

4.1 Rate control improvement

The new rate control method is based on a modified version of the Model-based rate control algorithm presented in Section 2.2. Given a source image and the target bit rate, the proposed model should supply an accurate estimation of both quantization parameters (Q and *rplanes*). As presented previously, the Model-based rate control method has an intrinsic average error of 5% at 1 bpp and 9% at 0.125 bpp. It is important to say that the resulting bit rate is always lower than the target one leading to a PSNR performance loss.

The modified rate control method should take into account the new sign coding tool presented in Section 3.1 and the Model-based rate control intrinsic error. The idea is to fix the underestimation error in order to match the final bit rate with the target one. To do that, we first estimate a 'Delta_Q' reduction factor in such a way that the resulting bit rate is under the target one, but this time very close to it. The main idea behind this approach is to overcompensate the scalar uniform quantization parameter (Q). In order to obtain this 'Delta_Q' value, we have obtained the best Q reduction value for all images in the Kodak set. There is a different 'Delta_Q' value at each T_{bpp} range and for each *rplanes* value. Also, we will apply this reduction factor only when the original Q value is in the range [0.56-0.9] in order to avoid an *rplanes* change. The 'Delta_Q' value is in the range [0.02-0.08].

After applying the new scalar uniform quantization factor, we encode the source image using the LTW encoder engine, but this time, when a significant coefficient is found, we will store its *rplanes* less significant bits. Finally, if we have not reached the target bit rate, then we append the number of bits required to match the target bit rate to the bit-stream by using those bits of the significant coefficients that correspond to bit planes lower than equal to the *rplanes* quantization parameter (bits that were removed after applying *rplanes* coarser quantization). These extra bits are appended to the bit-stream in a bit plane order (from bit plane '*rplane*' to 0), scanning the significant coefficients from the lowest frequency subbands to the highest ones (see Figure 4.1).

The whole algorithm, shown in Figure 4.2, works as follows:

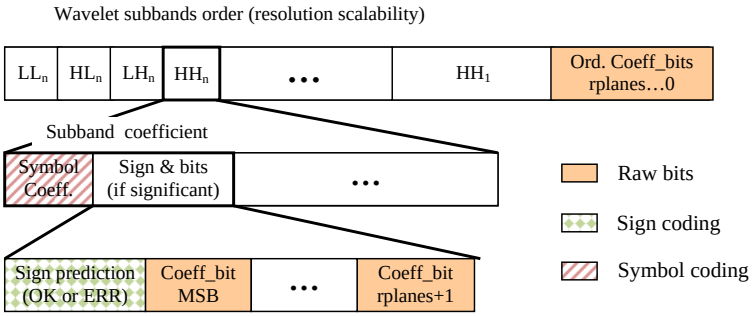


Figure 4.1: E-LTW bistream format

- First (E1), we obtain the quantization parameters *rplanes* and Q using the Model-based algorithm presented in Section 2.2.
- Second (E2), we apply the corresponding overcompensation ('Delta_Q') to the estimated Q parameter, taking into account the target bit rate (T_{bpp}) and the previously obtained *rplanes* parameter.
- Third (E3), we encode with the LTW encoder using the *rplanes* and the new Q quantization parameters. But, if the bit rate obtained is under the target one (T_{bpp}), extra bits are added to match the target bit rate. These extra bits that come from the *rplanes* less significant bits from the wavelet coefficients ($C_{i,j}$) are appended to the bit-stream in bit plane order, scanning the significant coefficients from the low frequency subbands to the highest ones.

Input: Wavelet Coefficients ($C_{i,j}$), Target bit rate (T_{bpp}), Curve minimum (K_{min})
Output: Q, *rplanes*
(E1) Obtain *rplanes* and Q from Model-based algorithm
($rplanes$,Q)=Model-based($C_{i,j}$, T_{bpp},K_{min})
(E2) Delta_Q = Overcompensate(Q,$rplanes$,T_{bpp})
Q = Q - Delta_Q
(E3) Encode (Append extra bits from *rplanes* to 0 if needed)

Figure 4.2: Enhanced Model-based rate control algorithm

In Figure 4.3, we show the bit rate accuracy of the proposed rate control method for the Woman test image. As shown in Figure 4.4, the behavior is

very similar in the other test images, with the average relative error below 0.5%.

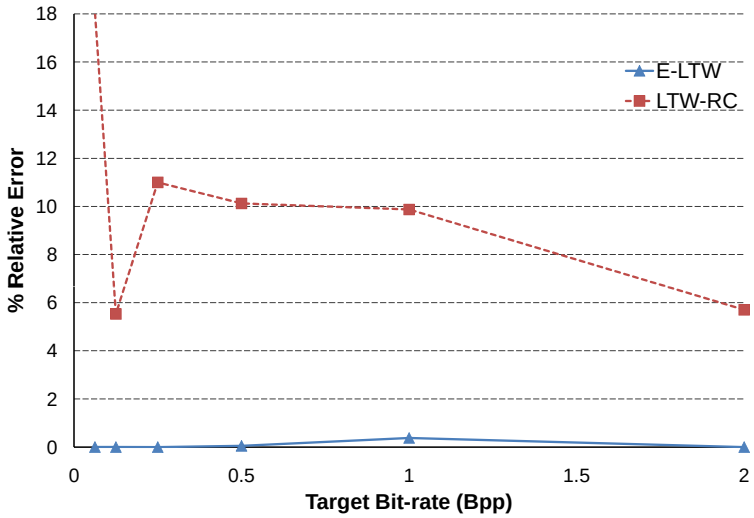


Figure 4.3: E-LTW bit rate accuracy for Woman test image

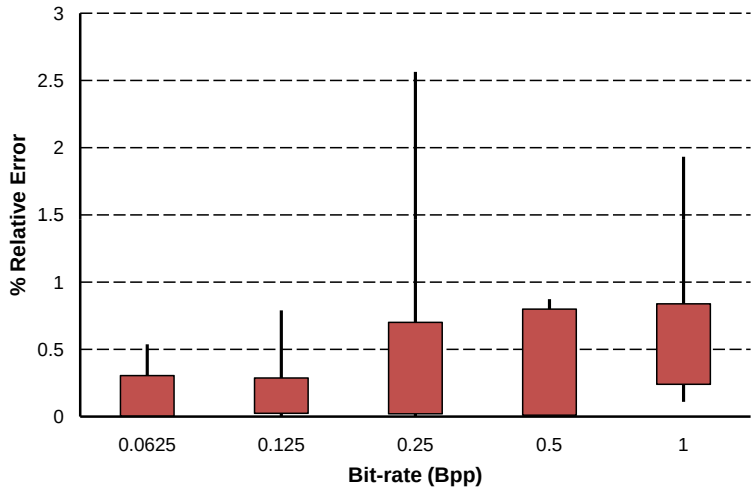


Figure 4.4: Bit rate accuracy for all tested images

4.2 Evaluation

In this section we analyze the behavior of the proposed encoder (E-LTW). We will compare the E-LTW encoder versus JPEG 2000 (Jasper 1.701.0), SPIHT

Bit rate	E-LTW	LTW	Gain	E-LTW	LTW	Gain
	Barbara (512x512)			Bike (2048x2560)		
2	42.59	41.51	1.08	43.59	42.75	0.84
1	36.69	35.61	1.08	37.82	36.97	0.85
0.5	31.63	30.80	0.83	33.28	32.36	0.92
0.25	27.92	27.09	0.82	29.45	28.40	1.04
0.125	25.02	24.29	0.72	26.09	25.09	1.00
0.0625	23.28	23.02	0.26	23.40	22.85	0.55
	Cafe (2048x2560)			GoldHill (512x512)		
2	39.10	37.66	1.44	42.34	42.21	0.13
1	31.95	30.88	1.07	36.55	36.18	0.37
0.5	26.77	26.04	0.73	33.10	32.85	0.26
0.25	23.13	22.60	0.53	30.54	30.49	0.05
0.125	20.70	20.65	0.05	28.49	28.15	0.34
0.0625	18.93	18.60	0.32	26.75	26.34	0.41
	Lena (512x512)			Mandrill (512x512)		
2	44.88	44.75	0.13	34.98	34.77	0.21
1	40.34	40.34	0	29.19	28.83	0.36
0.5	37.29	36.76	0.53	25.54	25.26	0.28
0.25	34.18	33.65	0.53	23.28	23.28	0.00
0.125	31.14	30.59	0.55	21.65	21.34	0.31
0.0625	28.36	27.79	0.57	20.67	20.64	0.03
	Woman (2048x2560)			Zelda (512x512)		
2	43.75	43.05	0.70	46.30	44.70	**1.60**
1	38.41	37.56	0.84	41.89	41.80	0.09
0.5	33.66	32.96	0.70	39.49	38.94	0.55
0.25	30.05	29.44	0.61	37.45	36.93	0.52
0.125	27.40	27.20	0.20	34.98	34.51	0.47
0.0625	25.55	25.09	0.46	32.33	31.87	0.47

Table 4.1: PSNR (dB) gain for several test images

(Spiht 8.01) and the original LTW_RC (Model-based rate control version), in terms of R/D, coding and decoding delay and memory requirements. All the evaluated encoders have been tested on an Intel PentiumM Dual Core 3.0 GHz with 1 Gbyte RAM memory. The encoders binaries were obtained by means of Microsoft Visual C++ (2005 version) compiler with the same project options.

In Table 4.1 we show the PSNR (dB) gain when comparing E-LTW with LTW-RC. As can be seen there is great improvement in PSNR (1.08 dB for the Barbara image), with the improvement greater at low compression ratios. The maximum PSNR improvement obtained is 1.6 dB for the Zelda image at 2 bpp. In Figures 4.5, 4.6 and 4.7 we show the E-LTW behavior

in R/D when compared to SPIHT and JPEG 2000. As shown, for high textured images like Barbara, JPEG 2000 has better behavior than E-LTW and SPIHT, and E-LTW better than SPIHT (up to 0.34 dB for the Barbara image at 0.25 bpp). On the other hand, in images like Lena or Zelda, both SPIHT and E-LTW have better behavior than JPEG 2000 (up to 0.52 dB at high compression ratios).

Codec/image	SPIHT	JPEG 2000	LTW-RC	E-LTW
Lena	3228	4148	**2092**	2212
Cafe	46776	65832	**21632**	25392

Table 4.2: Memory requirements for evaluated encoders (KB)

Bit rate	E-LTW	LTW-RC	JPEG 2000	SPIHT
	Barbara (512x512)			
1	0.051	**0.037**	0.080	0.042
0.5	0.031	**0.023**	0.076	0.026
0.25	0.026	**0.018**	0.074	0.018
0.125	0.015	**0.010**	0.073	0.014
0.0625	0.014	**0.008**	0.072	0.011
	Cafe (2048x2560)			
1	0.914	**0.648**	2.623	0.920
0.5	0.527	**0.382**	2.543	0.521
0.25	0.349	**0.225**	2.507	0.323
0.125	0.198	**0.158**	2.518	0.221
0.0625	0.140	**0.105**	2.509	0.172

Table 4.3: Coding delay (seconds) excluding DWT time

In Table 4.2, the memory requirements of the encoders under test are shown. The original LTW-RC needs only the amount of memory to store the source image and an extra 1.2 KB, basically used to store the histogram of significant symbols needed to accomplish the Model-based rate control algorithm. On the other hand, the E-LTW version requires little extra memory space to store the *rplanes* less significant bits of the significant coefficients. SPIHT requires more memory space than LTW-RC and E-LTW, and JPEG 2000 needs twice the memory of LTW-RC for medium size images and three times for high definition images (results obtained with Windows XP task manager, peak memory usage column).

As could be expected, E-LTW uses the arithmetic encoder more than LTW when coding the sign, so this fact implies a higher computational cost

in the coding and decoding process as shown in Table 4.3. The computational cost increase is 40% on average. For high resolution images like Bike or Cafe, E-LTW still remains competitive with respect to SPIHT and JPEG 2000.

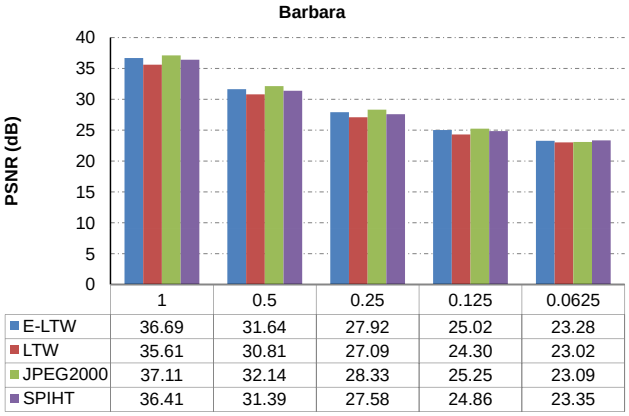
4.2.1 Fast arithmetic encoder evaluation

In a similar way like in the previous chapter, we have changed the arithmetic encoder stage by a fast arithmetic encoder [85] in order to compensate the coding speed loss due to the higher arithmetic encoder use. So in this subsection we will evaluate the original E-LTW encoder which uses an implementation of Witten's arithmetic encoder [111] against the E-LTW-Fast encoder which includes Said's fast arithmetic encoder [85].

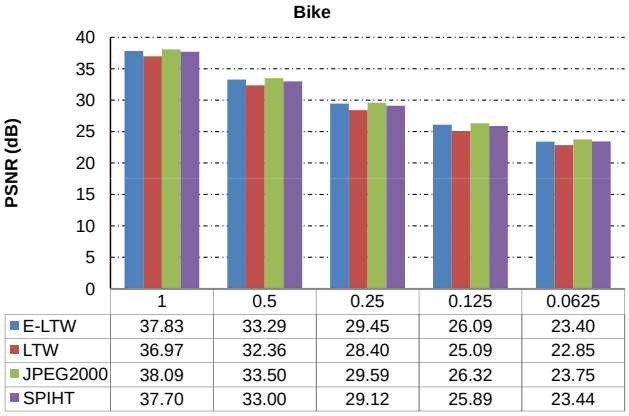
As shown in Figure 4.8 and Figure 4.9, E-LTW-Fast is faster than the original E-LTW regarding coding and decoding delay (up to 43% faster in the coding process and up to 67% faster in the decoding process). As presented in Chapter 3, there is slight compression loss by the use of the fast arithmetic encoder. In Figure 4.10 the PSNR difference between both encoders is shown. As can be seen, there is practically no difference between them.

4.3 Conclusions

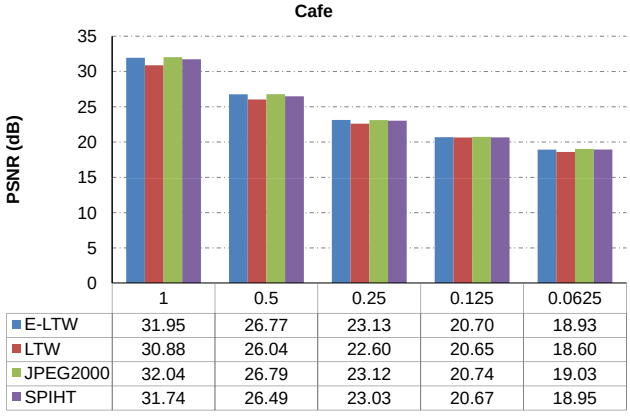
In this chapter we have presented a new LTW encoder version (E-LTW), and we have compared its performance with SPIHT and JPEG 2000 encoders in terms of R/D performance, execution time and memory consumption. The E-LTW encoder exhibits good R/D performance (up to 0.52 dB at high compression ratios compared to JPEG 2000 and up to 0.34 dB for high textured images compared to SPIHT). The use of high context modeling for sign coding together with the extra bit plane passes over the image has a high computational cost (40% overhead) but E-LTW still remains competitive for high resolution images (similar coding time as SPIHT and up to 2.3 times faster than JPEG 2000). Even more, the computational cost of our proposal is dominated by the arithmetic encoder processing, so in order to compensate the coding speed loss, we have changed the arithmetic encoder stage for a fast arithmetic encoder [85]. This new implementation is on average 30% faster in the coding process and 50% faster in the decoding process. Regarding memory requirements, E-LTW needs only a little extra amount of memory to store the *rplanes* less significant bits, and these requirements are 1.5 times fewer than SPIHT and 2 times fewer than JPEG 2000.



	1	0.5	0.25	0.125	0.0625
E-LTW	36.69	31.64	27.92	25.02	23.28
LTW	35.61	30.81	27.09	24.30	23.02
JPEG2000	37.11	32.14	28.33	25.25	23.09
SPIHT	36.41	31.39	27.58	24.86	23.35

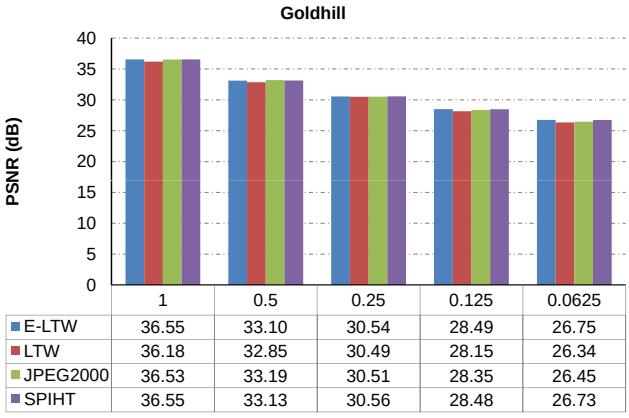


	1	0.5	0.25	0.125	0.0625
E-LTW	37.83	33.29	29.45	26.09	23.40
LTW	36.97	32.36	28.40	25.09	22.85
JPEG2000	38.09	33.50	29.59	26.32	23.75
SPIHT	37.70	33.00	29.12	25.89	23.44

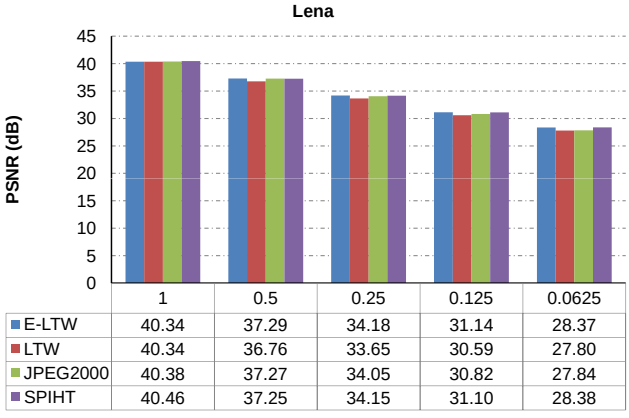


	1	0.5	0.25	0.125	0.0625
E-LTW	31.95	26.77	23.13	20.70	18.93
LTW	30.88	26.04	22.60	20.65	18.60
JPEG2000	32.04	26.79	23.12	20.74	19.03
SPIHT	31.74	26.49	23.03	20.67	18.95

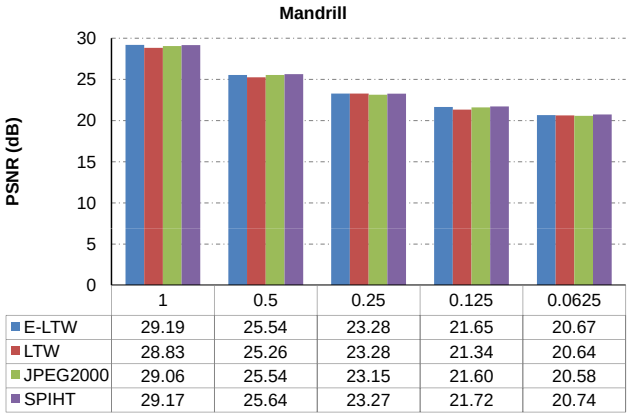
Figure 4.5: PSNR (dB) with different bit rate and coders



	1	0.5	0.25	0.125	0.0625
E-LTW	36.55	33.10	30.54	28.49	26.75
LTW	36.18	32.85	30.49	28.15	26.34
JPEG2000	36.53	33.19	30.51	28.35	26.45
SPIHT	36.55	33.13	30.56	28.48	26.73

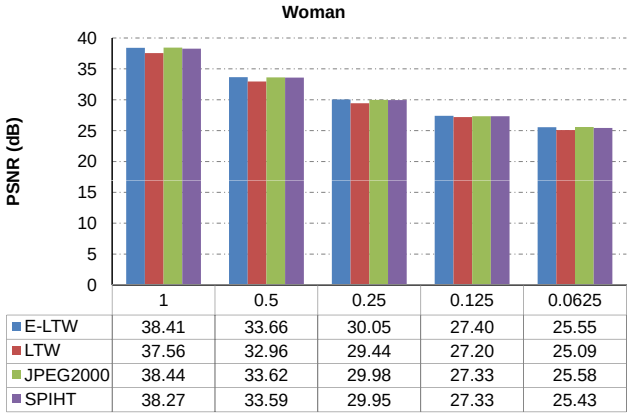


	1	0.5	0.25	0.125	0.0625
E-LTW	40.34	37.29	34.18	31.14	28.37
LTW	40.34	36.76	33.65	30.59	27.80
JPEG2000	40.38	37.27	34.05	30.82	27.84
SPIHT	40.46	37.25	34.15	31.10	28.38

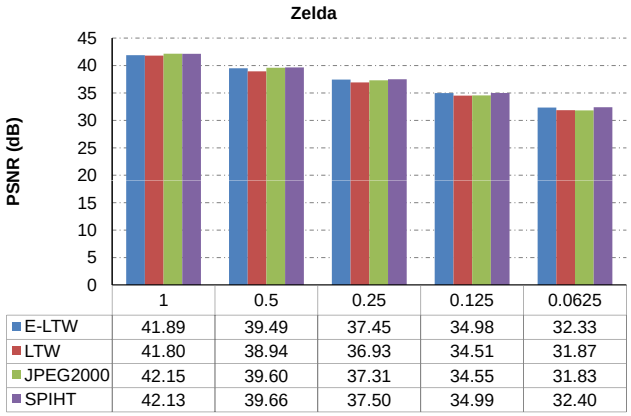


	1	0.5	0.25	0.125	0.0625
E-LTW	29.19	25.54	23.28	21.65	20.67
LTW	28.83	25.26	23.28	21.34	20.64
JPEG2000	29.06	25.54	23.15	21.60	20.58
SPIHT	29.17	25.64	23.27	21.72	20.74

Figure 4.6: PSNR (dB) with different bit rate and coders



	1	0.5	0.25	0.125	0.0625
E-LTW	38.41	33.66	30.05	27.40	25.55
LTW	37.56	32.96	29.44	27.20	25.09
JPEG2000	38.44	33.62	29.98	27.33	25.58
SPIHT	38.27	33.59	29.95	27.33	25.43



	1	0.5	0.25	0.125	0.0625
E-LTW	41.89	39.49	37.45	34.98	32.33
LTW	41.80	38.94	36.93	34.51	31.87
JPEG2000	42.15	39.60	37.31	34.55	31.83
SPIHT	42.13	39.66	37.50	34.99	32.40

Figure 4.7: PSNR (dB) with different bit rate and coders

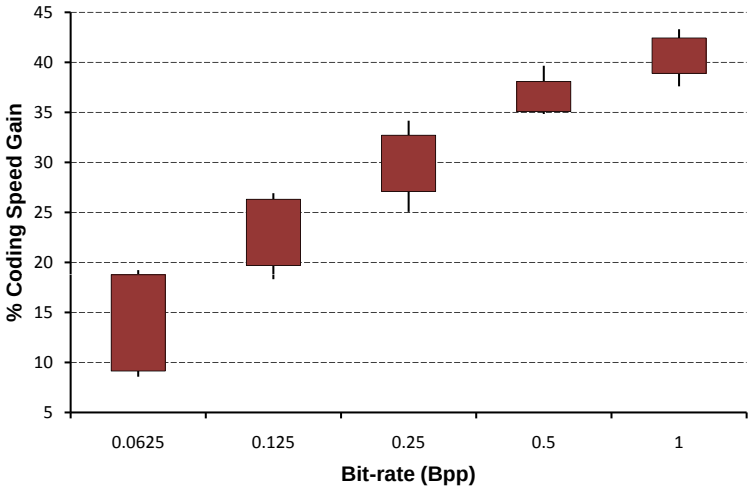


Figure 4.8: % Coding speed gain using Said's arithmetic encoder for all tested images

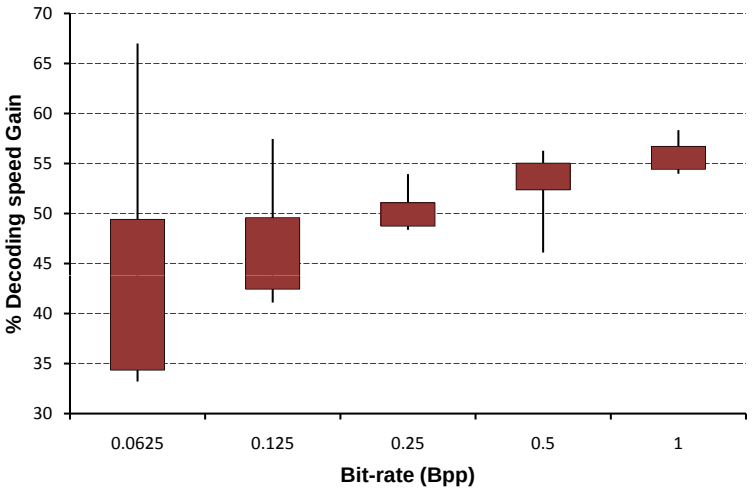


Figure 4.9: % Decoding speed gain using Said's arithmetic encoder for all tested images

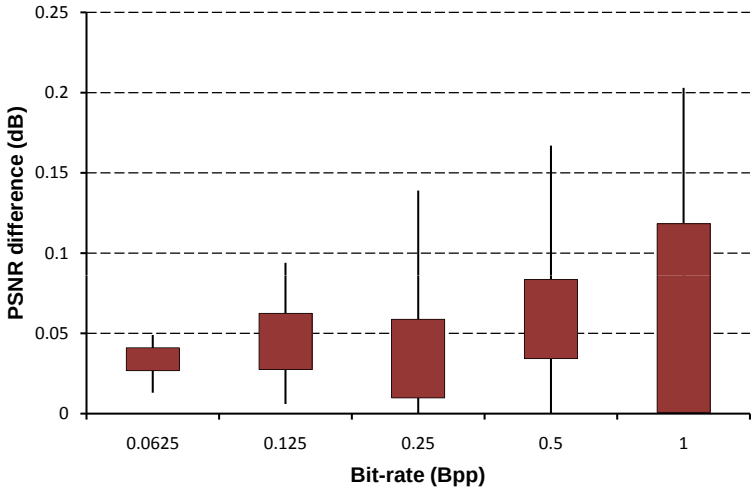


Figure 4.10: PSNR differences between E-LTW and E-LTW-Fast encoders for all tested images

Chapter 5

Motion LTW

Contents

Applications such as wireless video communications, wireless video cameras, disposable video cameras, and networked camcorders require low complexity encoders due to memory, computation, and power consumption limitations. Many compression approaches are currently being investigated targeting such applications, including distributed video coding [2, 78, 61] and low complexity hybrid video coding [109]. In distributed video coding, interframe coding based on motion estimation and compensation is performed in the decoder side in order to reduce the encoder complexity. However, typically such approaches still use traditional intra coding schemes such as H.264/AVC intra coding and JPEG 2000. Thus, intra coding is a potential bottleneck to achieving very low complexity video coding. In [60] a fast intra video encoder is used as the reference information in a distributed video coding scheme. Of course, beyond such advanced approaches, intra video coding is also widely used as a coding method in its own right, particularly in surveillance video applications.

It is well known that the H.264/AVC standard achieves much higher coding efficiency than previous video coding standards such as MPEG-1/2/4 [110]. The key feature of H.264/AVC intra coding is the prediction which is used to find spatial correlation. Although MPEG-4 uses a prediction method in the transform domain using the DC and several AC coefficients, its performance is not particularly good compared to H.264/AVC since new coding tools are deployed in H.264/AVC such as prediction based on pixel interpolation and rate distortion (RD) optimization. These kinds of spatial prediction coding methods are designed for reducing spatial redundancy by using the neighboring pixels of the current block. However, the resulting compression gain comes at the cost of a significant increase in processing time and memory access.

In this chapter, we present a new fast intra video encoder which includes an accurate rate control algorithm. This new encoder is able to encode an ITU D1 size image in real time with good R/D.

5.1 Extension of the Model-based rate control to intra video coding

In order to perform the rate control in the overall video sequence, we have extended the rate control algorithm explained in Section 2.2 (Figure 2.5) using a very simple approach. Firstly, we apply the rate control algorithm to the first frame to estimate the values of *rplanes* and Q quantization parameters that fit the frame bit rate budget. After coding the first frame, we compute

the estimation error, so we will try to compensate it when coding the following frames. We will do that keeping the same *rplanes* value and estimating the appropriate value for Q based on the observed error. During the video sequence coding, when the observed error reaches a threshold (SC_{th}), the algorithm launches the initial estimation algorithm to re-estimate more suitable *rplanes* and Q parameters so as to converge to the desired bit rate as fast as possible; then the accumulated error will be corrected gradually so as to avoid great R/D alterations. The threshold fixed on 20% of the target bit rate has been obtained from the Model-based algorithm inherent estimation error, that tends to be below 9% as concluded in Section 2.2.

In Figure 5.1 we show the accuracy of the proposed algorithm for the Container sequence with a CIF size and we compare it with respect to the embedded encoder SPIHT and JPEG 2000. Both SPIHT and JPEG 2000 obtain the exact target bit rate. With respect to the proposed rate control implemented in M-LTW and M-LTW_Integer, its accuracy was always better than 98.5%, and the worst case at very low target bit rates where the average relative error is approximately 0.4%. In Table 5.1 we show the accuracy of the proposed algorithm for several sequence frame sizes. As shown, the proposed rate control has similar behavior for all video sequences and with other frame sizes.

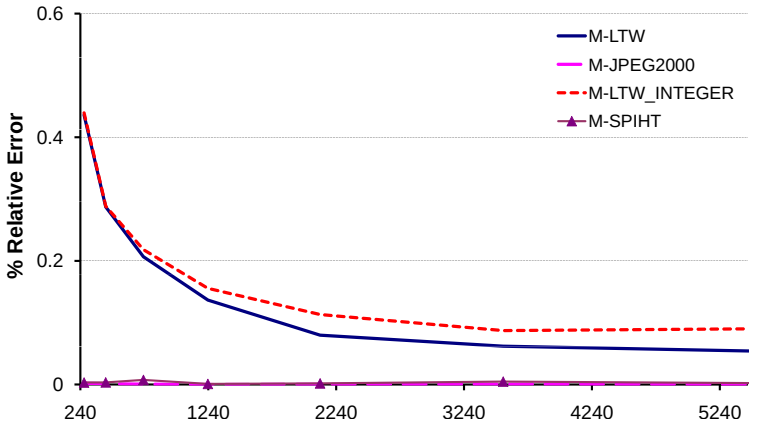


Figure 5.1: Rate-control accuracy for Container CIF sequence

In Figure 5.2 we can see the effect of the rate control algorithm over the CIF Coastguard sequence. As shown, during most parts of the sequence, the bit rate remains constant, but when significant alterations in the scene like camera movements, appearing objects or illumination changes occur, the algorithm produces a bit rate that reaches the fixed threshold. Then, the algorithm detects this situation and converges quickly to the desired bit rate.

Bit rate (Kbps)	M-JPEG 2000	M-SPIHT	M-LTW	M-LTW Integer
		Container (QCIF)		
83.16	0.004	0.158	1.236	1.243
136.87	0.004	0.051	0.774	0.775
233.21	0.009	0.093	0.467	0.476
382.29	0.020	0.003	0.285	0.294
627.95	0.002	0.007	0.199	0.202
996.42	0.001	0.008	0.141	0.156
1494.67	0.003	0.004	0.107	0.142
		Foreman (CIF)		
209.08	0.001	0.071	0.500	0.505
370.92	0.012	0.025	0.203	0.208
638.84	0.004	0.007	0.082	0.088
1112.68	0.006	0.001	0.049	0.039
1943.06	0.000	0.002	0.261	0.126
3340.75	0.007	0.003	0.271	0.313
5323.21	0.001	0.001	0.150	0.122
		Mobile (ITU D1)		
542.72	0.015	0.013	0.732	0.798
1142.41	0.066	0.011	0.543	0.658
2100.33	0.021	0.004	0.249	0.280
3598.16	0.008	0.004	0.416	0.586
6400.96	0.002	0.001	0.254	0.606
11587.03	0.012	0.002	0.043	0.371
20653.04	0.009	0.001	0.003	0.147
		Station2 (HD)		
1553.86	0.001	0.009	0.070	0.084
2348.20	0.003	0.000	0.073	0.085
4500.19	0.004	0.004	0.017	0.001
8400.44	0.000	0.001	0.075	0.065
15116.05	0.000	0.001	0.040	0.013
27938.34	0.000	0.000	0.113	0.081
53295.38	0.000	0.000	0.067	0.498

Table 5.1: Rate-control accuracy (% relative error) for several test sequences

If we focus on frames between 65-75 in Figure 5.2 we can see this behavior. This figure also shows the M-LTW encoder behavior when no rate control is applied.

Note that with the proposed rate control algorithm, we choose the quantization parameters of the last encoded frame as a reference to encode the current frame and reduce the accumulated rate control error. This works

fine while consecutive frames have similar contents (general case). However, when there are dramatic content changes compared to the previous frame (frames 65-75 at Figure 5.2), large bit rate oscillations are produced because our proposal does not know the complexity or similarity of the next frame.

In order to reduce the rate oscillations, several approaches may be followed: (1) Apply the algorithm presented in Figure 2.5 to all video frames so rate fluctuations are confined to the rate control algorithm precision and as a consequence the encoder becomes slower (45% complexity increment); and, (2) obtain the Mean Absolute Difference (MAD) of the lowest frequency subband (LL_n) between the current frame and the previously encoded one so as to detect scene changes. As Figure 5.3 shows, the use of MAD works fine and large rate oscillations are avoided at the expense of a 1.5% complexity increment.

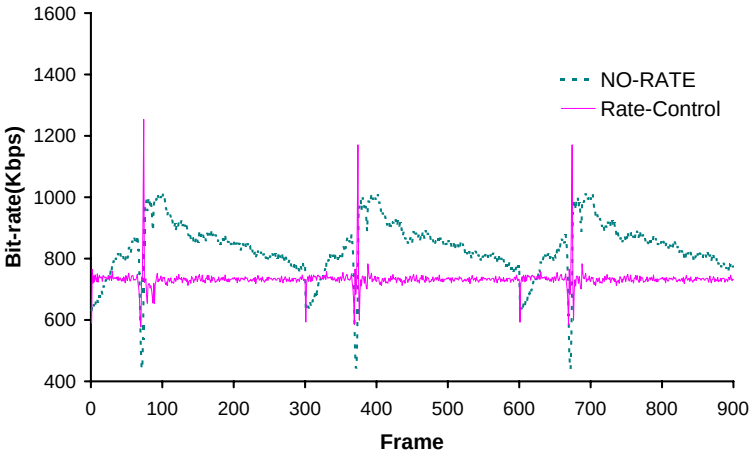


Figure 5.2: Rate-control progression for three concatenated Coastguard sequences (CIF)

The video rate control algorithm (Figure 5.4) will lead as follows:

- First, we obtain *rplanes* and Q parameters for the first frame by means of Model-based algorithm (see Figure 2.5).
- Secondly, we encode and evaluate the estimation error (P_{Err}).
- Then, for the remaining frames, we obtain the LL_n MAD from the current and following frames.
- If the previously evaluated error (P_{Err}) is greater than the threshold input parameter (SC_{th}), or the MAD is greater than the MAD threshold parameter(MAD_{th}), then we force new *rplanes* and Q estimation.

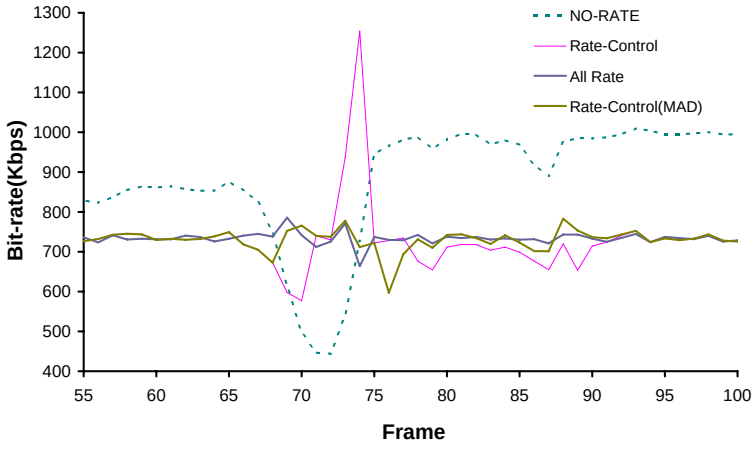


Figure 5.3: Rate-control proposals progression for Coastguard sequences (focused on frames 65-75) (CIF)

So, in the following frames we will gradually correct the bit rate error (Adjust%(P_{Err})).

- On the other hand, if the previously evaluated error (P_{Err}) is lower than the threshold input parameter (SC_{th}), we correct the bit rate error estimating with only a new Q value and fixing the *rplanes* parameter to the one obtained in previous encoded frames.

5.2 Integer Version

To carry out a fast integer version of LTW, we have developed the DWT with an integer-to-integer lifting scheme based on [12] and [22]. We have implemented the normalization factor of the lifting scheme (K) as an approximation to integer operations (multiply and shift). In this manner we avoid three extra lifting steps at the expense of making the DWT unreversible. Since we are interested in lossy compression, the fact of performing that approximation does not introduce a meaningful error, because the difference with respect to the regular lifting scheme is negligible.

Concerning the LTW encoder engine, we have converted all float operations to integer ones, and relating to the quantization process, this is similar to the one used by LTW described previously. The main difference lies in the scalar uniform quantization process, which is performed using only fixed point arithmetic operations.

Before adopting this approach, we tried other integer-to-integer DWT filters so as to obtain a fast version of DWT with similar R/D values. From

Input: Video Sequence Frames, Target bit rate (T_{bpp}), Threshold(SC_{th}) and MAD_{th} Threshold.
(E1) Obtain rplanes and Q using Algorithm 2.5 for the first frame.
P_{Err} = Encode_And_Evaluate_Error
Initialize SC_{Change}**= false**
(E2) For all remaining frames
Obtain LL_n **MAD** from current and next frame.
if $MAD > MAD_{th}$
SC_{Change}= true
if ($P_{Err} > SC_{th}$) or (SC_{Change})
Obtain **new** rplanes and Q using Algorithm 2.5.
SC_{Change}= true
else
if SC_{Change}== true
$NewT_{bpp} = T_{bpp}$+Adjust%(P_{Err})
else
$NewT_{bpp} = T_{bpp} + P_{Err}$
P_{Err}= Encode_And_Evaluate_Error

Figure 5.4: Video rate control algorithm

the study carried out by M.D. Adams and F. Kossentini in [4], we decided to implement 13/7T and 9/7M filters because these filters only use two lifting steps instead of the four lifting steps required by the 9/7F filter. After evaluating the filters we concluded that the use of other filters lead us to an R/D loss of approximately 1 dB with respect to the 9/7F filter, as shown in Figure 5.5. However, we do not use the original 9/7F filter but an approximation (9/7F_int_Aprox.), which replaces the three extra lifting steps required for the normalization factor (K) by an integer approximation that truncates the wavelet coefficients to integers and thus introduces a small quantization noise. Contrary to what could be thought, it does not introduce any R/D loss, even showing slightly better behavior than the original one at low compression ratios. This behavior is similar in all tested images and appears only at low to very low compression ratios (*rplanes*=2).

This curious effect is due to the asymmetric error produced by the proposed integer approximation of the normalization factor. In the DWT, the introduced error is slightly greater in low frequency subbands than in high frequency subbands (error propagation in successive decomposition levels). However, this error has higher influence over the high frequency subbands

producing a bit plane change in several significant coefficients (notice that at low compression ratios there are many small significant coefficients in those subbands), and as a consequence fewer bits are needed to represent them. On the other hand, the Inverse Discrete Wavelet Transform (IDWT) introduces a greater error over high frequency subbands. So, the previously introduced DWT error is further compensated, obtaining slightly better behavior than in the fully reversible DWT version. This behavior disappears at higher compression ratios (*rplanes*>2) when the encoder quantization noise is greater than the error introduced by the DWT.

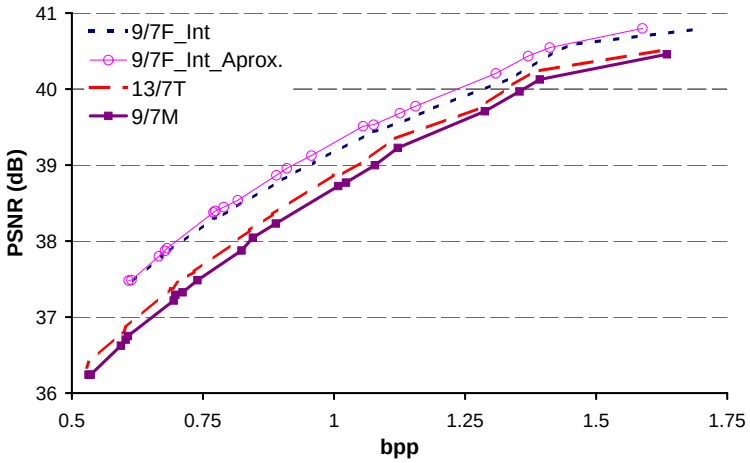


Figure 5.5: R/D evolution using different filters for Lena (512x512)

In order to determine the most appropriate DWT kernel, we have implemented three versions of the LTW encoder. The first one implements the DWT 9/7F filter with a traditional convolution, the second one implements a lifting scheme of DWT with floating point arithmetic operations and the third one implements an integer-to-integer lifting scheme of DWT with an integer approximation of the normalization factor (K). So as to measure the R/D loss due to fixed-point arithmetic operations, in Figures 5.6 and 5.7 we compare the R/D performance for both floating-point and fixed-point implementations. As shown in Figure 5.6, for the Barbara test image there is almost no difference in R/D between all proposals (note that both floating point DWT implementations obtain the same PSNR because they use the same filter D(9/7F), so only convolution implementation is represented in figures 5.6 and 5.7), but if we focus on Figure 5.7, for the Lena test image there is a loss of approximately 0.5 dB at a compression rate of 0.5 bpp. These results are in concordance with the ones presented by Grangetto in [29].

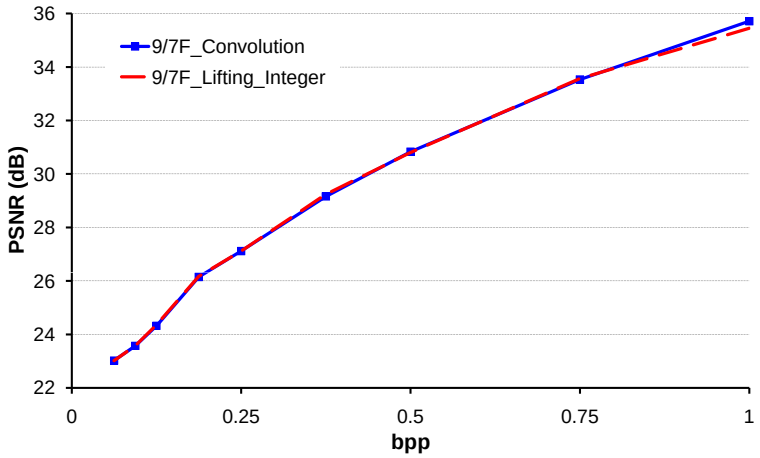


Figure 5.6: R/D evaluation for different DWT proposals for the Barbara (512x512) test image

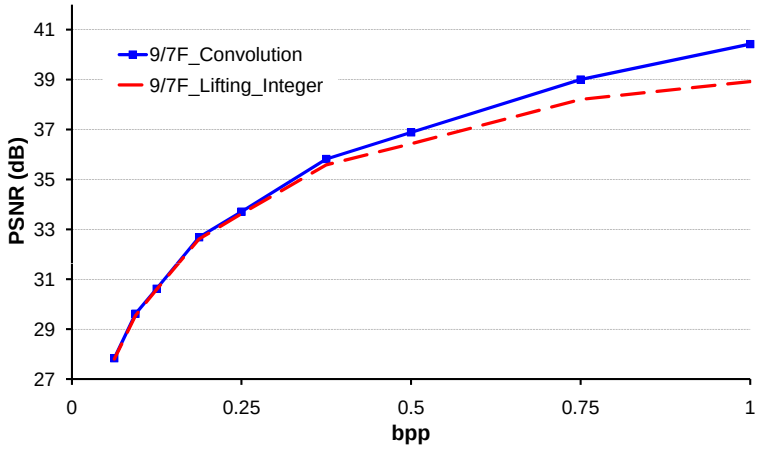


Figure 5.7: R/D evaluation for different DWT proposals for the Lena (512x512) test image

With regard to execution time, as we can see in Figure 5.8, both lifting scheme DWT implementations are faster than traditional convolution. In this figure, the execution time of the floating point implementations also includes a cast from float to integer, because the rate control algorithm presented in Section 2.2 operates only with fixed point values. This fact causes that differences in execution time between lifting scheme implementation and traditional convolution are not as significant as the results presented by Grangetto in [29]. The integer-to-integer lifting scheme implementation of DWT is the fastest one, being 50% faster than the floating point arithmetic lifting scheme implementation due to the cast to integer differences mentioned previously.

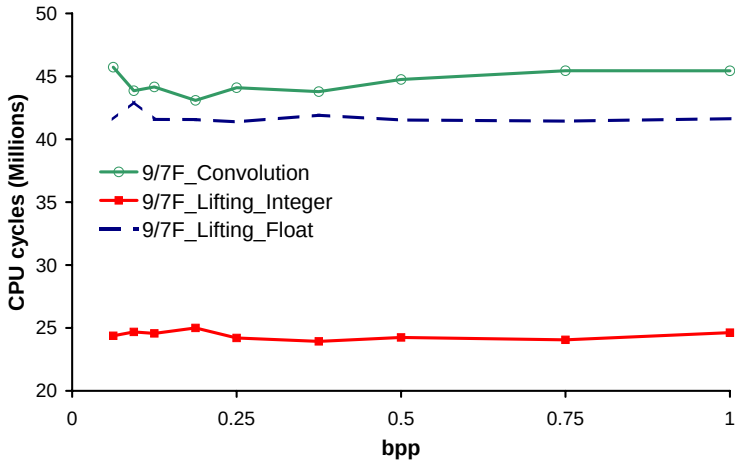


Figure 5.8: Execution time comparison between different proposals of DWT for the Barbara test image

As a consequence of this study, we have finally planned to carry out two versions of LTW, one based on floating point arithmetic operations that implements DWT with a lifting scheme and another based on fixed point arithmetic operations that implements DWT with an integer-to-integer lifting scheme approximation.

5.3 Evaluation

In addition to R/D performance we will also employ other performance metrics like coding delay and memory consumption. All the evaluated encoders have been tested on an Intel PentiumM Dual Core 3.0 GHz with 1 Gbyte RAM memory. We have selected H.264 (Baseline, JM10.2) working in intra mode, M-JPEG 2000 (Jasper 1.701.0), M-SPIHT (Spiht 8.01), M-LTW and M-LTW_Int (integer version of M-LTW), since their source code is available for testing. The correspondent binaries were obtained by means of Microsoft Visual C++ (2005 version) compiler with the same project options and under the aforementioned machine.

The test video sequences used in the evaluation are: Foreman (QCIF and CIF) 300 frames, Container (QCIF and CIF) 300 frames, News (QCIF and CIF) 300 frames, Mobile (ITU D1 576p30) 40 frames and Station2 (HD 1024p25) 312 frames.

5.3.1 Objective/Subjective quality evaluation

Table 5.2 shows the R/D evaluation of all proposed encoders. Although H.264 obtains better results for sequence sizes smaller than CIF at several compression ratios, it is for ITU D1 and HD sizes where encoders based on DWT can exploit optimal DWT decompositions obtaining better results (up to 2 dB with respect to H.264 in Station2 HD at high compression ratios). The M-LTW_Int encoder produces slightly lower PSNR results than H.264. The lower performance of the integer version is mainly due to the arithmetic precision loss, which is more noticeable at lower compression ratios.

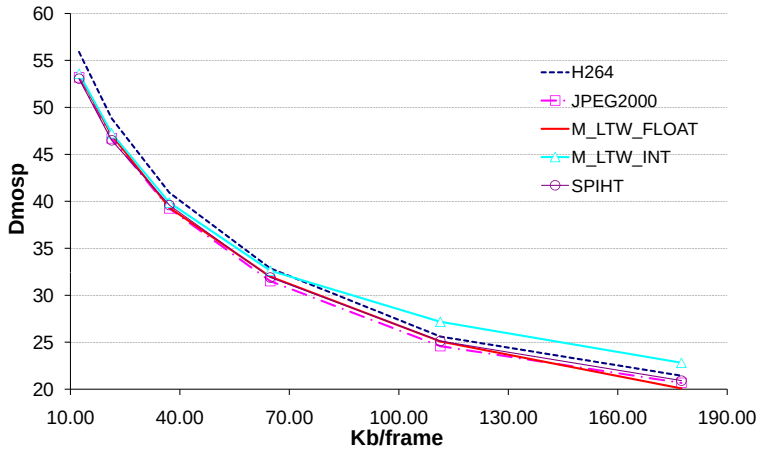


Figure 5.9: R/D evaluation with VIF metric on DMOSp space for Foreman (CIF) sequence

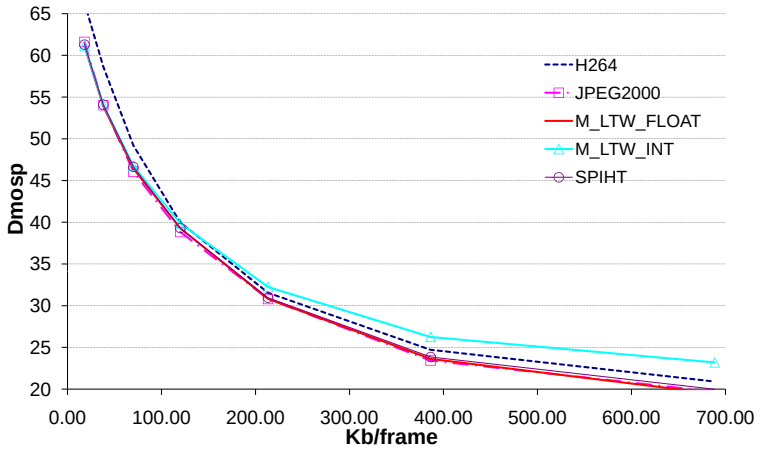


Figure 5.10: R/D evaluation with VIF metric on DMOSp space for Mobile (ITU D1) sequence

We have also compared all proposed encoders using the Visual Information Fidelity (VIF) distortion metric [95]. We have chosen this metric because it is one of the best 'full reference' objective quality metrics as concluded in [62]. In Figure 5.9 and Figure 5.10 we can observe the behavior of evaluated encoders for Foreman (CIF) and Mobile (ITU D1) sequences. As can be seen, all encoders based on DWT have similar performance at high compression ratios with a lower DMOSp (Predicted Differential Mean Opinion Score) value (better quality) than H.264. Only at lower compression ratios does H.264 outperform M-LTW_Int, although at these compression ratios, the differences are not visually perceptible. The remaining coders show very similar results under the VIF quality metric, so they can be considered equivalent in terms of R/D performance.

In Figure 5.11 we present a subjective evaluation between M-LTW and M-LTW_Int in order to determine the R/D loss in the fixed point version. Although differences in PSNR between both M-LTW and M-LTW_Int encoders are approximately 0.4 dB, it is difficult to determine which one has better subjective quality at low compression ratios. On the other hand, at

(a) 34.66dB (b) 34.24dB (c) Original

Figure 5.11: Subjective comparison between a) M-LTW; and, b) M-LTW_Int for Foreman (QCIF) at 20.49 Kb/frame, frame # 33

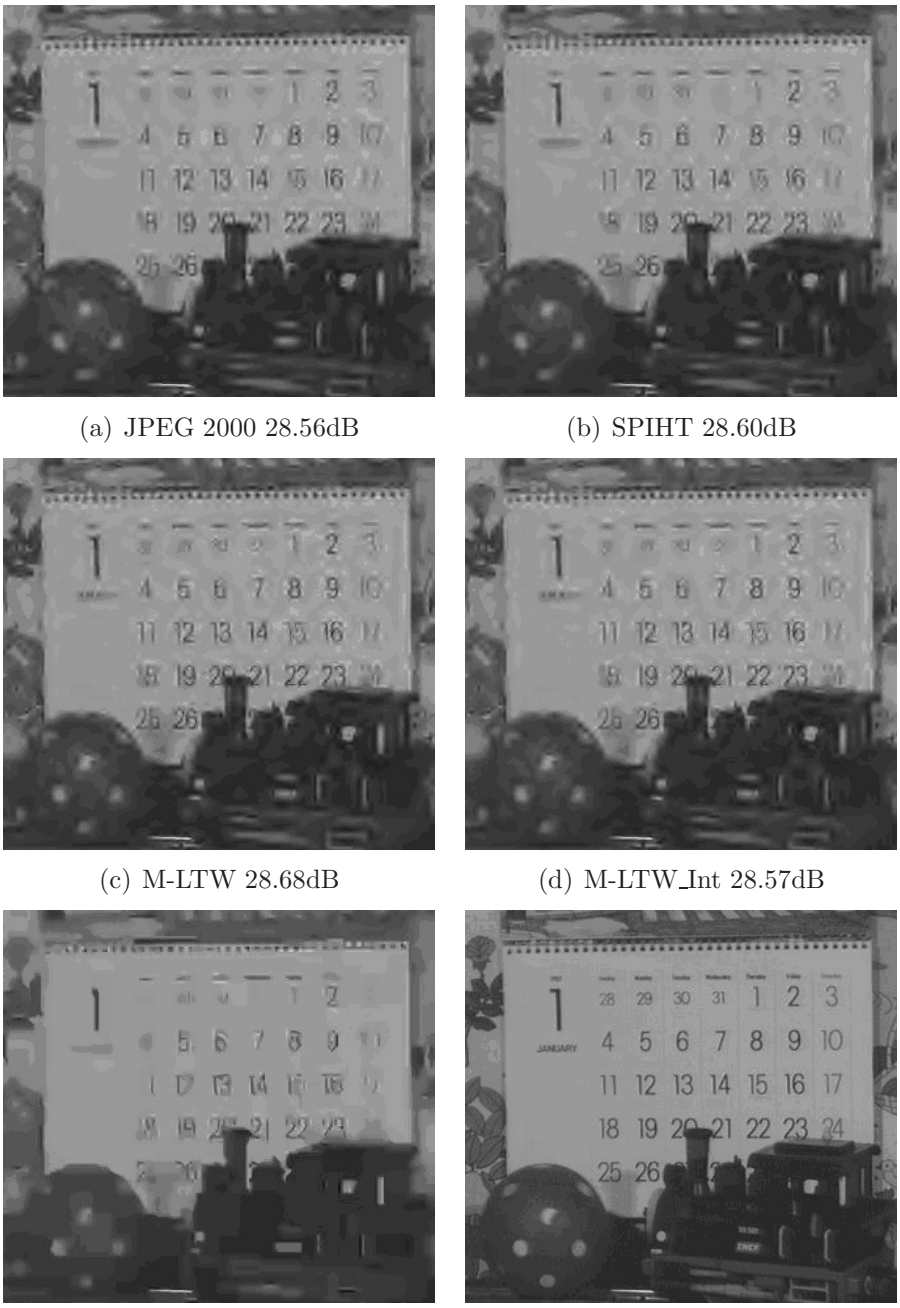
high compression ratios differences of 1 dB on PSNR are visually perceptible as shown in Figure 5.12. All wavelet based encoders show similar behavior, but if we focus on the calendar (number 15), we could asses that M-LTW (c) and M-LTW_Int (d) have better subjective quality than JPEG 2000 (a) and SPIHT (b). In spite of the fact that SPIHT shows a better PSNR value, M-LTW_Int is visually slightly better at this compression rate.

5.3.2 Execution time and memory consumption comparison

In Table 5.3 we show the coding delay for all encoders under evaluation. As expected, H.264 is the slowest encoder and M-LTW is one of the fastest.

Codec/Bit rate (Kb/frame)	H.264	M-JPEG 2000	M-SPIHT	M-LTW	M-LTW _Int
			Container (QCIF, 30Hz)		
2.77	22.85	22.07	**23.08**	22.08	22.06
7.77	**28.94**	27.36	27.54	27.23	27.17
20.93	**35.76**	34.71	34.70	34.89	34.37
33.21	**39.52**	39.03	38.76	39.39	37.99
			Container (CIF, 30Hz)		
8.97	23.58	24.43	**24.47**	24.38	24.34
24.44	**29.44**	28.69	28.40	28.53	28.42
70.49	**35.64**	35.53	34.94	35.28	34.70
118.20	39.36	**39.47**	38.74	39.44	38.01
			Foreman (QCIF, 30Hz)		
2.36	22.86	21.10	**24.16**	23.03	23.01
7.40	**28.72**	28.21	28.69	28.69	28.58
20.49	**35.36**	34.68	34.59	34.99	34.40
33.73	39.24	38.89	38.47	**39.37**	37.62
			Foreman (CIF, 30Hz)		
6.97	24.57	25.95	**26.34**	26.20	26.13
21.29	29.97	30.43	30.55	**30.66**	30.47
64.77	36.00	36.06	35.86	**36.31**	35.54
111.36	39.59	39.43	39.09	**39.98**	37.96
			News (QCIF, 30Hz)		
3.22	22.24	21.60	**23.00**	21.97	21.91
9.12	**28.73**	27.53	27.86	27.98	27.88
22.72	**36.11**	34.74	34.84	35.19	34.54
34.22	**40.22**	39.31	39.02	39.90	38.22
			News (CIF, 30Hz)		
9.27	24.43	25.21	25.54	**25.57**	25.51
14.91	27.37	27.34	**27.76**	27.73	27.63
36.76	**33.97**	33.11	33.01	33.37	33.01
89.91	**41.14**	40.63	40.08	41.00	38.91
			Mobile (ITU D1, 30Hz)		
38.08	27.04	28.48	28.53	**28.59**	28.48
119.93	32.29	32.41	32.36	**32.57**	32.26
213.36	35.29	35.09	35.05	**35.40**	34.75
386.23	38.59	38.43	38.29	**38.87**	37.21
			Station2 (HD, 25Hz)		
93.92	30.49	32.37	32.29	**32.45**	32.19
180.00	32.58	34.38	34.25	**34.49**	34.06
604.64	37.55	38.67	38.39	**39.02**	37.73
1117.53	40.37	40.78	40.44	**41.38**	39.08

Table 5.2: Average PSNR (dB) with different bit rate and coders



(a) JPEG 2000 28.56dB (b) SPIHT 28.60dB

(c) M-LTW 28.68dB (d) M-LTW_Int 28.57dB

(e) H.264 27.11dB (f) Original

Figure 5.12: Subjective comparison between a) JPEG 2000; b) SPIHT; c) M-LTW; d) M-LTW_Int; e) H.264; and, f) Original for Mobile (ITU D1) at 38.08 Kb/frame, frame # 20

All M-LTW versions are faster than M-JPEG 2000, specially the fixed point version that performs the encoding process six times faster on average than M-JPEG 2000. On the other hand, M-SPIHT is faster than M-LTW only in HD video format.

Codec/Bit rate (Kb/frame)	H.264	M-JPEG 2000	M-SPIHT	M-LTW	M-LTW _Int
	CODING Container (QCIF, 30Hz)				
2.77	121.59	3.95	1.68	0.84	**0.54**
7.77	137.14	4.15	2.13	1.09	**0.72**
20.93	167.38	4.26	2.86	1.63	**1.16**
33.21	189.90	4.43	3.55	2.13	**1.54**
	CODING News (CIF, 30Hz)				
14.91	531.40	15.63	3.77	3.96	**2.62**
23.62	559.45	15.20	4.33	4.26	**2.81**
57.73	650.47	15.54	6.67	5.98	**3.94**
89.91	720.44	16.43	8.23	7.17	**4.95**
	CODING Mobile (ITU D1, 30Hz)				
38.08	233.27	8.99	**1.06**	1.83	1.25
119.94	266.94	7.88	1.83	2.38	**1.68**
213.37	297.11	8.02	2.63	2.93	**2.13**
386.23	351.41	8.29	4.24	3.97	**2.96**
	CODING Station2 (HD, 25Hz)				
93.93	11840.89	326.05	**34.13**	75.78	53.86
180.01	12106.46	327.18	**41.12**	79.24	56.59
604.64	13573.54	326.04	**73.75**	104.67	76.13
1117.53	15067.72	330.81	113.66	129.25	**93.13**

Table 5.3: Execution time comparison of the coding process including DWT (time in seconds)

Figure 5.13 shows the maximum frame rate for all evaluated encoders at different sequence sizes for an average PSNR video quality of 30 dB. The integer version of M-LTW is one of the fastest encoders and it can encode an ITU D1 size sequence in real time. For HD images, M-LTW is slower than M-SPIHT. This behavior is due to the cache page miss fail of the lifting DWT implementation where a lazy transform is carried out for both rows and columns. In the lazy transform, the input samples are split into two data sets, one with the even samples of a row or column and the other one with the odd ones. This causes a significative cache page miss fail increase, being more noticeable for columns, as the frame size becomes larger. In order to

measure the impact of the cache page miss fail, in Figure 5.14 we evaluate the performance of M-LTW implemented with both lifting and convolution DWT in a processor with an L2 cache size of 1MB. As can be seen, lifting DWT is 16.6% slower than convolution DWT for the HD format and 10% slower for the ITU D1 format. With the increase of the L2 cache size, these differences are significatively reduced, being the difference for HD format of 9% as shown in Figure 5.15.

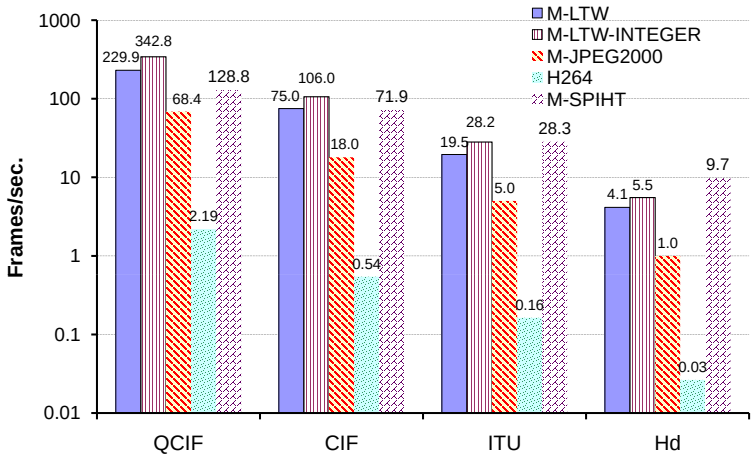


Figure 5.13: Maximum Frames/sec. for an average R/D of 30dB

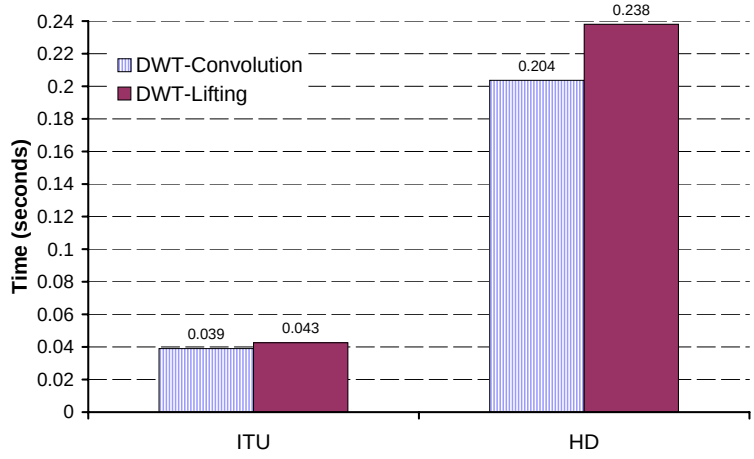


Figure 5.14: Execution time comparison between DWT Lifting and DWT Convolution (1 frame, 1MB L2 cache)

In Table 5.4 the memory requirements of different encoders under test are shown. M-LTW needs only the amount of memory to store the source image

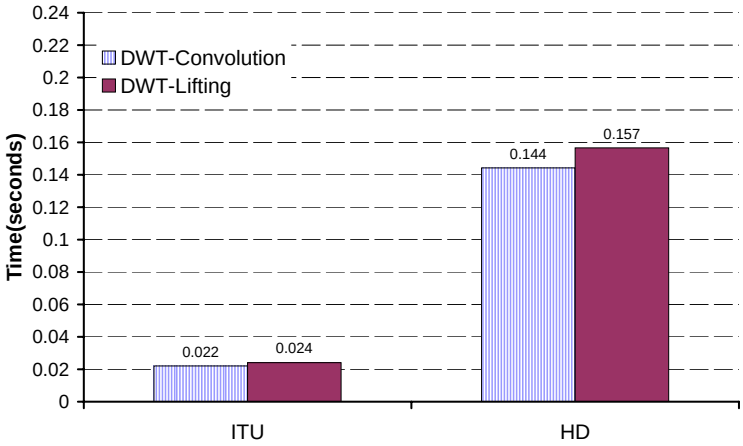


Figure 5.15: Execution time comparison between DWT Lifting and DWT Convolution (1 frame, 2MB L2 Cache)

Codec/Format	H.264	M-JPEG 2000	M-SPIHT	M-LTW	M-LTW_Int
QCIF	6508	2264	1864	**1104**	**1104**
CIF	13016	3920	2880	**1540**	**1540**

Table 5.4: Memory requirements for evaluated encoders (KB) (results obtained with Windows XP task manager, peak memory usage column)

(in-line processing) and an extra 100 KB basically used to store the histogram of significant symbols required by the rate control algorithm, variables and structures needed to accomplish the coding process. M-JPEG 2000 requires twice the memory amount of M-LTW, and H.264 needs six times the memory amount of M-LTW for the QCIF format and eight times for the CIF format. M-SPIHT uses 1.7 times the memory amount of M-LTW. Note that M-LTW_Int could be implemented using a 16-bit integer, reducing the memory requirements by one-half.

5.3.3 Optimized encoders

The M-LTW implementation was developed finding the optimizations for maximizing R/D performance, so its software code is not optimized, just like H.264 and JPEG 2000 reference software. However, we have compared its performance with respect to a full optimized implementation of JPEG 2000: Kakadu [45], in order to evaluate whether a full optimization of M-LTW is worth the effort. For that purpose, we have used two versions of Kakadu

software: (a) version 2.2.3, compiled without optimization options, and (b) the 5.2.5 version which is fully optimized including multi-thread and multi-core hardware capabilities, processor intrinsics like MMX/SSE/SSE2/SIMD and fast multicomponent transform.

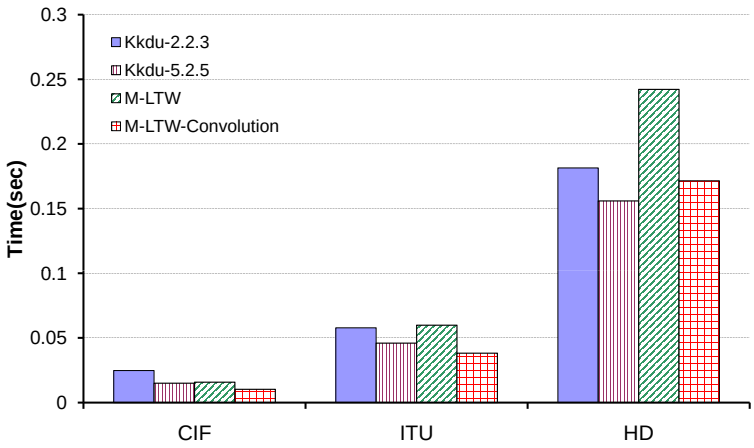


Figure 5.16: Execution time comparison (end-to-end) of the coding process

As shown in Figure 5.16, M-LTW is a very fast encoder even though not being fully optimized. The speed of M-LTW lies in the simple engine coding model. M-LTW is approximately equally as fast as Kakadu-5.2.5 for News CIF sequence at a PSNR of 32dB. For HD images, M-LTW is slower than Kakadu-2.2.3, due to the cache page miss fail of the lifting DWT implementation, as shown in the previous subsection. Therefore, if we use a convolution implementation of DWT, M-LTW would be slightly faster than Kakadu-2.2.3 and slightly slower than Kakadu-5.2.5 for the Station2 HD sequence.

Regarding memory requirements, M-LTW needs only the amount of memory to store the source image as was said before, while Kakadu memory requirements are independent of the image size due to its DWT block-based implementation, and it is on average 1420KB.

In terms of R/D, there are slight differences between all codecs as Table 5.5 shows. For small and medium size images, M-LTW outperforms Kakadu at medium and high compression ratios. For larger images, M-LTW provides slightly lower PSNR than both versions of Kakadu.

So, a full optimization of M-LTW codec will certainly increase coding speed and will reduce the memory requirements even more, making the codec a very competitive intra video coding solution.

Codec/(Kb/frame)	KAKADU 2.2.3	KAKADU 5.2.5	M-LTW
	News (CIF, 30Hz)		
14.91	27.63	27.44	**27.74**
23.62	30.27	29.96	**30.42**
36.75	33.33	33.31	**33.36**
57.73	**37.26**	37.10	36.89
	Mobile (ITU D1, 30Hz)		
38.08	28.59	28.39	**28.61**
119.93	32.56	32.52	**32.62**
213.36	35.34	35.34	**35.47**
386.23	38.85	38.89	**38.90**
	Station2 (HD, 25Hz)		
93.92	**33.79**	33.70	33.62
180.00	**36.16**	36.15	36.08
604.64	**41.11**	41.11	40.96
1117.53	**43.18**	43.18	42.94

Table 5.5: PSNR (dB) comparison between Kakadu and M-LTW

5.4 Conclusions

In this chapter we have presented a fast an efficient intra video coder, M-LTW, which is based on the non-embedded LTW image coder. We have proposed a fast and lightweight rate control algorithm for both M-LTW encoder versions, a float-point implementation and another one implemented with integers. After evaluating M-LTW performance in terms of R/D, execution time and memory consumption, it exhibits the best trade-off between R/D performance, coding delay (3 times faster than M-JPEG 2000 and 108 times faster than H.264) and overall memory usage (half the amount of memory of M-JPEG 2000 and 6 times less than H.264). In addition, the M-LTW coder is able to encode an ITU D1 video signal in real time with very low memory demands and good R/D performance at moderate to high compression ratios (2 dB better than H.264 for HD video format).

For further evaluation, we have compared the M-LTW coder with a highly optimized version of JPEG 2000 (Kakadu), being also competitive in terms of coding delay (similar to Kakadu for small and medium size images) and R/D performance (0.4 dB for CIF, and 0.1 dB for ITU D1 at medium and high compression ratios). So, a fully optimization process will make M-LTW even faster and with lower memory requirements. These optimizations will be mainly focused on the DWT coding step by using fast and low memory

demanding DWT techniques like line-based or block-based ones and exploiting the parallel capabilities of modern processors (like multithreading and SIMD instructions).

Chapter 6

3D Video Coding

Contents

In recent years, three-dimensional wavelet transform (3D-DWT) has focused the attention of the research community, most of all in areas such as video watermarking [13] and 3D coding (e.g., compression of volumetric data [86] or multispectral images [25], 3D model coding [8], and especially, video coding).

In video compression, some early proposals were based on merely applying the wavelet transform on the time axis after computing the 2D-DWT for each frame [51]. Then, an adapted version of an image encoder can be used, taking into account the new dimension. For instance, instead of the typical quad-trees of image coding, a tree with eight descendants per coefficient is used in [51] to extend the SPIHT image encoder to video coding. However, the coding efficiency of these video encoders is poor in moderate-to-high motion sequences due to the appearance of misaligned objects in the time direction, causing an energy increase in high-frequency subbands, and thereby preventing energy concentration in low-frequency subbands. A more efficient strategy for video coding with time filtering is Motion Compensated Temporal Filtering (MCTF) [89, 15]. In these techniques, in order to compensate object (or pixel) misalignment between frames, and hence avoid the significant amount of energy that appears in high-frequency subbands, a motion compensation algorithm is introduced to align all the objects (or pixels) in the frames before being temporally filtered.

In all these applications, the first problem that arises is the extremely high memory consumption of the 3D wavelet transform if the regular algorithm is used, since a group of frames must be kept in memory before applying temporal filtering, and in the case of video coding, we know that greater the temporal decorrelation, the more number of frames are needed in memory. Another drawback is the necessity of grouping images in small GOPs to prevent very high memory usage, because the 3D-DWT must be computed along a set of images which are held in memory. This division of the video sequence in GOPs containing only a few images hinders the decorrelation of the temporal dimension and causes boundary effects between GOPs.

Even though several proposals have been made to avoid the aforementioned problems, most of them are not general (for any wavelet transform) and/or complete (the wavelet coefficients are not the same as those from the usual dyadic wavelet transform). In addition, software implementation is not always easy. In this paper, we extend to video the line-based approach introduced in [18] to compute the 2D-DWT. This approach is general so any wavelet transform can be computed using this new approach. To ease software implementation, we use the same recursive strategy as in [72].

6.1 3D Wavelet Transform

In the regular 3D-DWT, the wavelet transform is applied in the three directions, i.e., in the spatial directions (horizontal and vertical) and in the time direction (which is known as temporal filtering), resulting in eight first level wavelet subbands (typically named LLL_1, LHL_1, LLH_1, LHH_1, HLL_1, HHL_1, HLH_1, HHH_1). Afterwards, the same decomposition can be done, focusing on the low-frequency subband (LLL_1), achieving in this way a second-level wavelet decomposition, and so on (see example in Figure 6.1(b)).

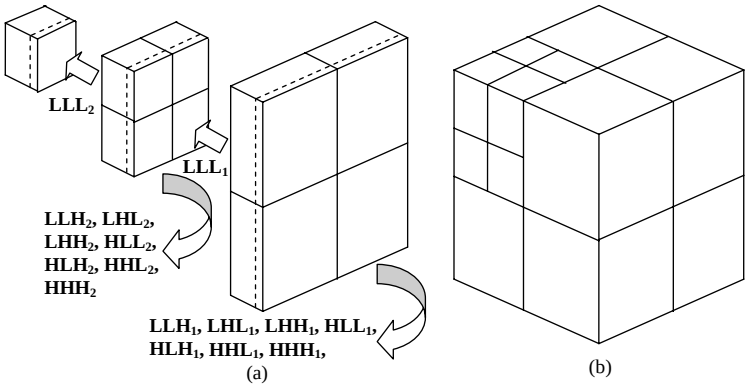


Figure 6.1: Overview of the 3D-DWT computation in a two-level decomposition, (a) following a frame-by-frame scheme as shown in Algorithm 1; or, (b) the regular 3D-DWT algorithm

Because this algorithm is clearly memory intensive, with very high memory requirements, and exhibits high coding delay (the whole 3D-DWT needs to be computed before starting the coding stage) several alternative proposals have been made.

Some of these alternatives are based on modifying the order in which the temporal filtering is calculated. E.g., in [65] the authors propose computing the wavelet transform in the time direction with only a few frames; then the resulting high-frequency frames are released as a part of the final result, and the low-frequency frames are employed along with a few more frames so as to continue computing the wavelet transform in the time direction. A similar example is [67], where the temporal decomposition is done by interleaving frames in small groups, getting a low-frequency frame per group, which is stored to be decomposed later with the low-frequency frames from the rest of groups. Although both algorithms ([65] and [67]) require less memory, the resulting coefficients are far from being the same as in the regular algorithm.

Other proposals rely on blocking algorithms [9], in which the transform is computed in working subsets to reduce memory usage and exploit data

locality. Despite the use of overlapping techniques to avoid typical blocking artifacts, the coding efficiency decreases because the redundancy among neighboring blocks is not exploited.

In Motion Compensation Temporal Filtering (MCTF) [89, 15], the temporal decomposition is usually carried out with a very simple transform based on the lifting scheme [101]. When using filters with only a prediction and an update step (or even sometimes the update step is skipped), only a few frames need to be handled in MCTF. However, if longer filters with several decomposition levels are applied in the temporal dimension, memory handling becomes difficult.

6.2 Frame-By-Frame 3D-DWT

In this section we propose an extension to a three-dimensional wavelet transform of the line-based approach [18], which computes the 2D-DWT with reduced memory consumption. In the new approach, frames are continuously input with no need to divide the video sequence into GOPs. Moreover, the algorithm yields slices of wavelet subbands (which we call subband frames) as soon as it has enough frames to compute them. This approach works as follows:

For the first decomposition level the algorithm directly receives frames one by one. On every input frame, a one-level 2D-DWT is applied. Then, this transformed image is stored in a buffer associated to the first decomposition level. This buffer must be able to keep 2N+1 frames, where 2N+1 is the number of taps for the largest analysis filter bank. We only consider odd filter lengths because they have higher compression efficiency; however, this analysis could be extended to even filters as well.

When there are enough frames in the buffer to perform one step of a wavelet transform in the temporal direction (z-axis), the convolution process is calculated twice, first using the low-pass filter and then the high-pass filter. The result of this operation is the first frame of each high-frequency subband (the HHL_1, HLH_1, HHH_1, HLL_1, LHL_1, LLH_1 and LHH_1 wavelet subbands), and the first frame of the LLL_1 subband. At this moment, for a dyadic wavelet decomposition, we can process and release the first frame of the wavelet subbands. However, the first frame of the LLL_1 subband does not belong to the final result, but it is needed as incoming data for the following decomposition level. On the other hand, once the frames at the first level buffer have been used, this buffer is shifted twice (using a rotation operation) so that two frames are discarded while another two frames are input at the other end. Once the buffer is updated, the process can be repeated

and more subband frames are obtained.

At the second level, its buffer is filled with the LLL_1 frames that have been computed in the first level. Once the buffer is completely filled, it is processed in the very same way as we have described for the first level. In this manner, the frames of the second level wavelet subbands are achieved, and the low-frequency frames from LLL_2 are passed to the third level. As depicted in Figure 6.1(a), this process can be repeated until the desired decomposition level (*nlevel*) is reached.

In this algorithm a major problem arises when it is implemented. This drawback is the synchronization among buffers. Before a buffer can produce frames, it must be completely filled with frames from previous buffers, therefore they start working at different moments, i.e., they have different delays. Moreover, all the buffers exchange their result at different intervals, according to their level.

Handling several buffers with different delays and rhythms becomes a hard task. The next section proposes a recursive algorithm that clearly specifies how to perform this communication between buffers.

6.3 A recursive implementation of the frame-by-frame 3D-DWT

In this section, we present an algorithm based on [72] that automatically solves the synchronization problem among levels that has been addressed in the previous section. To solve this problem, this algorithm defines a recursive function that obtains the next low-frequency subband frame (LLL) from a contiguous level.

The algorithm starts requesting LLL frames to the last level (*nlevel*). As seen in Figure 6.1, the *nlevel* buffer must be filled with subband frames from the *nlevel*-1 level before it can generate frames. In order to get them, this function recursively calls itself until level 0 is reached. At this point, it no longer needs to call itself since it can return a frame from the video sequence, which can be directly read from the input/output system.

The complete recursive algorithm is formally described in Figure 6.2, while Figure 6.3 sets up the variables and performs the DWT by calling the recursive algorithm. Let us see the first algorithm.

The first time that the recursive function is called at every level, it has its buffer ($buffer_{level}$) empty. Then, its upper half (from N to 2N) is recursively filled with frames from the previous level. Recall that once a frame is received, it must be transformed using a 2D-DWT before being stored. Once the upper

function GetLLLFrame (*level*)
1) First base case: No more frames to read at this level
  if $FramesRead_{level} = MaxFrames_{level}$
    return EOF
2) Second base case: The current level belongs
to the space domain and not to the wavelet domain
  else if $level = 0$
    return InputFrame()
  else
3) Recursive case
3.1) Recursively fill or update the buffer for this level
  if $buffer_{level}$ is empty
    for $i = N \ldots 2N$
      $buffer_{level}(i) = 2DFWT(GetLLLFrame(level - 1))$
    FullSymmetricExtension($buffer_{level}$)
  else
    repeat twice
      Shift($buffer_{level}$)
      $frame$ = GetLLLFrame($level - 1$)
      if $frame$ = EOF
        $buffer_{level}(2N)$ = SymmetricExt($buffer_{level}$)
      else
        $buffer_{level}(2N)$ = 2DFWT($frame$)
3.2) Calculate the WT for the time direction from the frames
in buffer, then process the resulting high frequency subband frames
  $\{LLL, LLH, LHL, LHH\}$ =Z-axis_FWT_LowPass($buffer_{level}$)
  $\{HLL, HLH, HHL, HHH\}$ =Z-axis_FWT_HighPass($buffer_{level}$)
  ProcessSubFrames($\{LLH, LHL, LHH, HLL, HLH, HHL, HHH\}$)
  set $FramesRead_{level}$=$FramesRead_{level} + 1$
  return LLL
end of fuction

Figure 6.2: GetLLLFrame Recursive function

half is full, the lower half is filled by using symmetric extension (the N+1 frame is copied into the N-1 position,. . . , the 2N is copied into the 0 position). On the other hand, if the buffer is not empty, it simply has to be updated. In order to update it, it is shifted one position so that the frame contained in the first position is discarded and a new frame can be introduced in the

function LowMemUsage3D_FWT(*nlevel*)
  set $FramesRead_{level} = 0\ \forall level \in nlevel$
  set $FramesLines_{level} = \frac{Nframes}{2^{level}}\ \forall level \in nlevel$
  set $buffer_{level}$ = empty $\forall level \in nlevel$
  repeat
    LLL = GetLLLFrame(*nlevel*)
    if (LLL != EOF) ProcessLowFreqSubFrame(LLL)
  until LLL = EOF
end of fuction

Figure 6.3: Perform the 3DFWT by calling GetLLLFrame recursive function

last position (2N) by using a recursive call. This operation is repeated twice.

However, if there are no more frames in the previous level, this recursive call will return End of Frame (EOF). That points out that we are about to finish the computation at this level, but we still need to continue filling the buffer. We fill it by using symmetric extension again.

Once the buffer is filled or updated, both high-pass and low-pass filter banks are applied to the frames in the buffer. As a result of the convolution, we get a frame of every wavelet subband at this level, and an *LLL* frame. The high-frequency coefficients are processed according to the application (compressed, saved to secondary storage, etc.) and this function returns the LLL frame.

Every recursive function needs at least one base case to stop backtracking. This function has two base cases. The first case is when all the frames at this level have been read. It is detected by keeping an account of the number of frames read and the maximum number of frames that can be read at every level. In this case, the function returns EOF. An alternative when the number of frames in the sequence is unknown a priori is propagating the EOF label. The second base case is reached when the level gets 0 and then no further recursive call is needed since a frame can be directly read from the input video sequence.

The inverse DWT algorithm (IWT) is similar to the forward DWT, but applied in reverse order. An important difference between this proposal and those based on GOPs is how the video can be decoded from the middle of the bit-stream, that is, if the user begins receiving the video broadcast while it is already in progress. In the regular algorithm, the current group of frames being received is ignored, and then, the following group is stored in memory. After it has been entirely received, it can be decoded, and the

3D-IWT can be applied. On the other hand, for the inverse transform in the frame-by-frame scheme, the decoding process begins immediately by filling up the highest-level buffer (*nlevel*) with the information received from the bit-stream. During this process, other information from the bit-stream is ignored. Afterwards, once this buffer is full, we also begin to accept information from the previous level, and so forth, until all the buffers are full. At that moment, the video can be sequentially decoded as usual. The latency of this process is determinist and depends on the filter length and the number of decomposition levels (the higher they are, the higher latency). However, for the regular 3D algorithm, the latency depends on the remaining number of frames in the current group when the process begins, and the GOP size.

A drawback that has not been considered yet is the need to reverse the order of the subbands, from the forward DWT to the inverse one. This problem can be solved by using some buffers at both ends, so that data are supplied in the right order [18]. Other simpler solutions are to save every level in secondary storage separately so that it can be read in a different order and, if the WT is used for compression, to keep the compressed bit-stream in memory.

6.4 Coding stage

A fast encoding algorithm for the frame-by-frame 3D wavelet transform has been developed. This new coding stage is based on a run-length approach. The quantization used in this coding stage is similar to the one used in LTW-RC previously presented in Chapter 2. There are two quantization parameters, one coarser (*rplanes*) and one finer (Q). The finer consists in applying a uniform scalar quantization to all wavelet coefficients, while the coarser one consists in removing the *rplanes* less significant bits to all the coefficients.

Because of the need to reverse the order of the subbands, we store the compressed bit-stream in buffers until we reach the maximum decomposition level. At this moment, an LLL subband is obtained, and so it is encoded and sent to the final bit-stream in conjunction with the previously stored bit-stream.

6.5 Evaluation

In this section we analyze the behavior of the proposed encoder (3D-RLW). We will compare the 3D-RLW encoder versus the M-LTW Intra video en-

coder presented in Chapter 5, 3D-SPIHT [51] and H.264 (JM16.1 version), in terms of R/D, coding and decoding delay and memory requirements. All the evaluated encoders have been tested on an Intel PentiumM Dual Core 3.0 GHz with 1 Gbyte RAM memory.

6.5.1 Memory consumption comparison

In this new algorithm (frame-by-frame 3D wavelet transform), each buffer must be able to keep either $2N+1$ low frequency frames at every level (recall that $2N+1$ is the filter length), or even less if the lifting scheme is used as shown in [72]. As presented in Figure 6.1(a), each buffer at a level i needs a quarter of coefficients if compared with the previous level $(i-1)$. Therefore, for a frame size of $(w \times h)$ and an *nlevel* time decomposition, the number of coefficients required by this algorithm is:

$$(2N+1)\times(w\times h)+(2N+1)\times(w\times h)\,/4+\cdots+(2N+1)\times(w\times h)\,/4^{nlevel-1} \tag{6.1}$$

which is asymptotically (as *nlevel* approaches infinity)

$$\sum_{n=0}^{\infty}\frac{(2N+1)\times(w\times h)}{4^n}=(2N+1)\times(w\times h)\times\frac{4}{3} \tag{6.2}$$

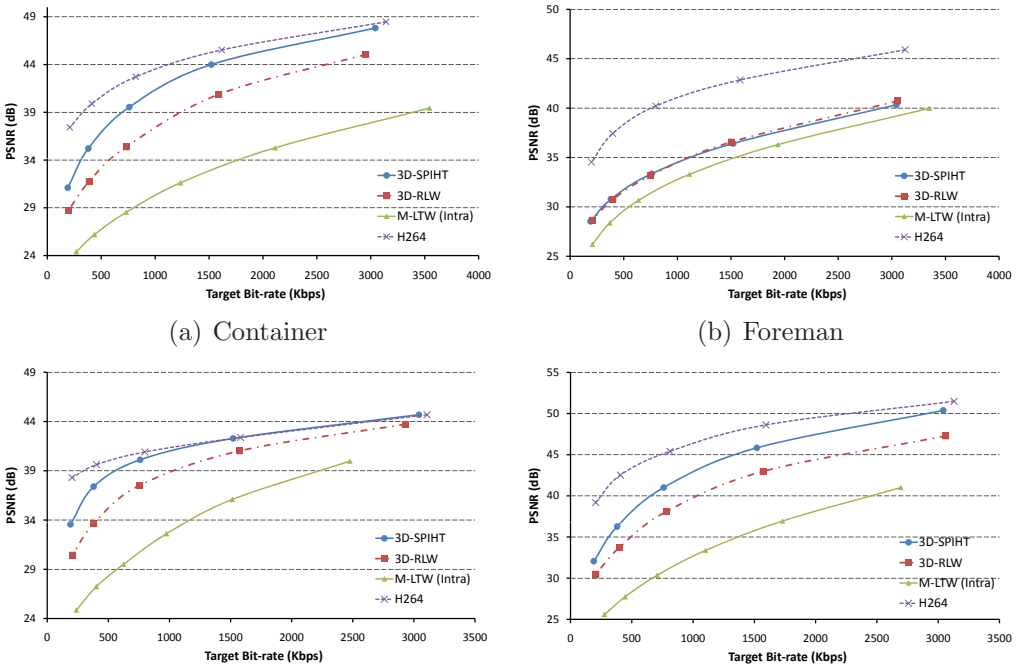
independent of the number of frames to be encoded, less than the regular case, which needs $(w \times h \times G)$, being G the number of frames in a GOP. For instance, in a C implementation, a 3D-DWT of a CIF sequence (with B5/3 and three decomposition levels) has required 2.5 MB with our new proposal, while the regular algorithm with 32 frames/GOP needs 12.4 MB, and introduces discontinuities in the transform, being less efficient for coding purposes. In Table 6.1, the memory requirements of different encoders under test are shown. 3D-RLW (using 9/7F time filter) uses 3 times less memory than 3D-SPIHT and up to 10 times less memory than H.264 for QCIF sequence size.

Codec/Format	H.264	3D-SPIHT	3D-RLW
QCIF	35824	10152	**3556**
CIF	86272	34504	**11616**

Table 6.1: Memory requirements for evaluated encoders (KB) (results obtained with Windows XP task manager, peak memory usage column)

6.5.2 Coding delay and R/D

In Figure 6.4 and 6.5 we can see the R/D behavior of all evaluated encoders. As shown, H.264 is the one that obtains best results, mainly due to the ME/MC stage included in this encoder, contrary to 3D-SPIHT and 3D-RLW that do not include any ME/MC stage. It is interesting to see the improvement of 3D-SPIHT and 3D-RLW when compared to an INTRA video encoder. As mentioned, no ME stage is included in 3D-SPIHT and 3D-RLW, so this improvement is accomplished by exploiting only the temporal redundancy among video frames.

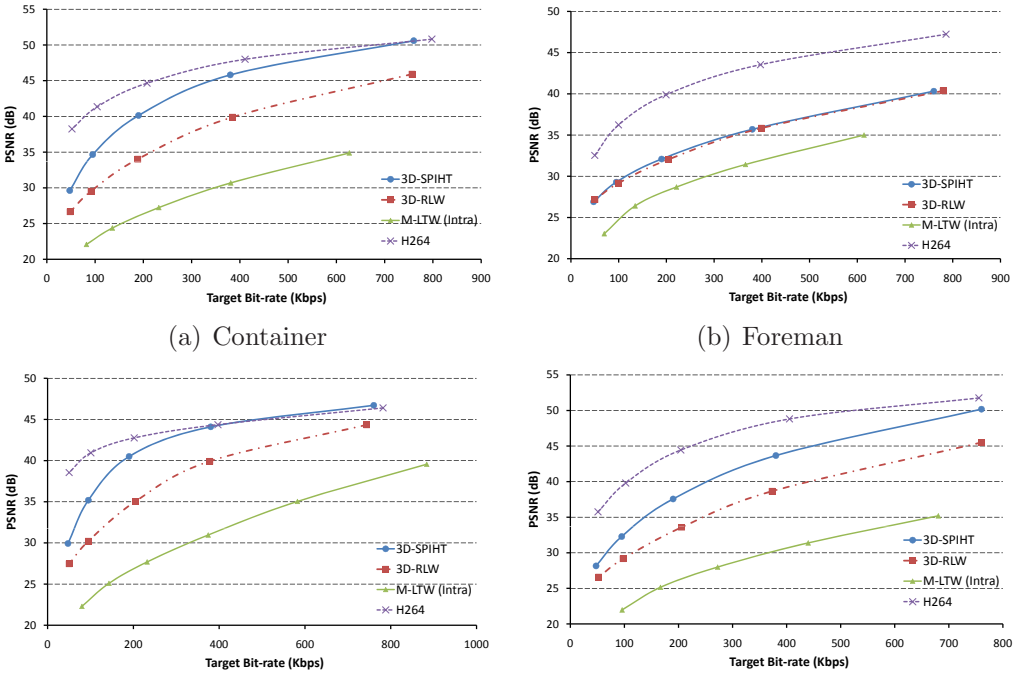


(a) Container (b) Foreman

(c) Hall (d) News

Figure 6.4: PSNR (dB) for all evaluated encoders for a) Container; b) Foreman; c) Hall; and, d) News sequences in CIF format

Regarding coding delay, in Figure 6.6 we can see that the 3D-RLW encoder is the fastest one, being 6 times faster than 3D-SPIHT and 3 times faster than the M-LTW INTRA video encoder.



(a) Container (b) Foreman

(c) Hall (d) News

Figure 6.5: PSNR (dB) for all evaluated encoders for a) Container; b) Foreman; c) Hall; and, d) News sequences in QCIF format

6.6 Conclusions

In this chapter a frame-by-frame 3D-DWT transform algorithm has been presented, considering the existing problems about different delays and rhythms among the buffers. The new algorithm reduces the memory requirements compared with the regular one, computing exactly the same coefficients. In addition, there is no need to artificially divide the video sequence in constant-size groups of pictures. Even more, a fast run-length encoder stage has been included in this 3D scheme. The new 3D encoder is very fast (6 times faster than 3D-SPIHT) and it has better R/D behavior than the INTRA video coder M-LTW. In order to improve the coding efficiency, an ME/MC stage could be added. In this manner, the objects/pixels of the input video sequence will be aligned, and so, fewer frequencies will appear at the higher frequency subbands.

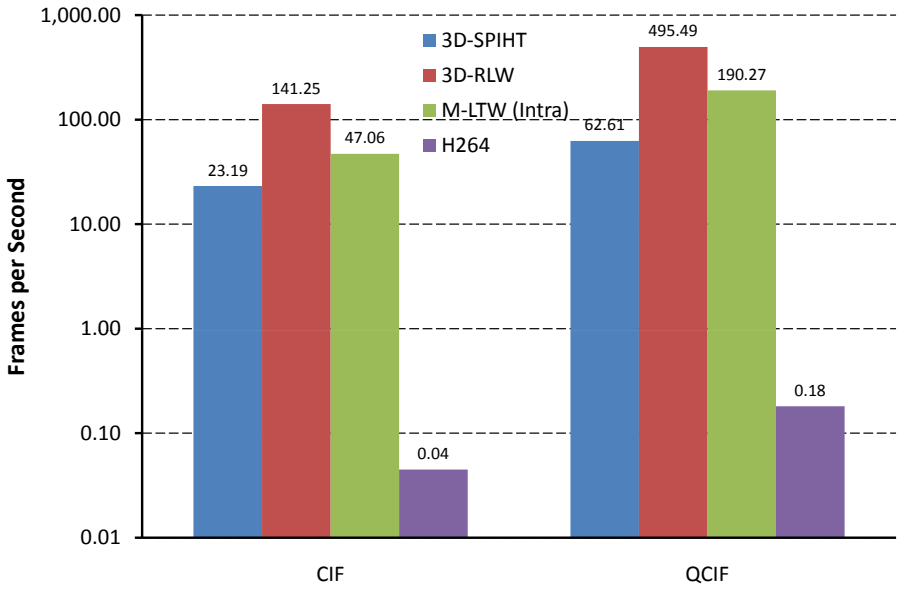


Figure 6.6: Execution time comparison of the coding process including DWT (time in seconds)

Chapter 7

Conclusions

Contents

7.1 Conclusions

Although a detailed summary section with the main contributions and conclusions is presented at the end of each chapter, it is interesting to summarize some of the main contributions introduced in this thesis.

In Chapter 2 we propose three different rate control tools for non-embedded wavelet-based encoders with increasing complexity and accuracy. We have shown that we can add rate control functionality to non-embedded wavelet encoders without a significant increase in complexity and little performance loss. Among the proposed simple rate control tools, the LTW_RC proposal presented in Section 2.2 is the one that exhibits the best trade-off between R/D performance and coding delay (twice faster than SPIHT and 8.8 times faster than JPEG 2000).

In Chapter 3 we perform a study about the usefulness of sign coding techniques for non-embedded image encoders. Furthermore, we present a simplified context model formation oriented to maximize the successful prediction of the sign for every non-zero wavelet coefficient. As shown in this study, the R/D performance improves (up to 0.25 dB), with the improvement greater at low and medium compression rates. Of course, there is an increase in coding delay due to the higher arithmetic encoder use (two contexts for every subband type), but the proposed encoder (S-LTW) is still competitive against SPIHT and JPEG 2000. Moreover, if we change the arithmetic encoder stage by a faster arithmetic encoder like the one presented in [85], the coding process becomes 28% faster on average and the decoding process becomes 47% faster on average than the original S-LTW encoder.

In Chapter 4 a new enhanced LTW encoder version (E-LTW) is presented. This new encoder incorporates an improved rate control algorithm with high accuracy (less than 0.5% error) and the sign coding stage presented in Chapter 3. The E-LTW encoder exhibits good R/D performance (up to 0.52 dB at high compression rates compared to JPEG 2000 and up to 0.34 dB for high textured images compared to SPIHT).

Regarding video coding, in Chapter 5 we present a fast INTRA video encoder called M-LTW. This encoder includes an extension of the rate control algorithm proposed in Section 2.2. The new video rate control algorithm has great accuracy (always better than 98.5%) and is also able to detect scene changes. This fast INTRA video encoder is able to encode an ITU D1 video signal in real time with good R/D performance.

Also, in Chapter 6, we present a fast coding stage (6 times faster than 3D-SPIHT) for a 3D wavelet recursive transform with low memory usage (3 times less memory than 3D-SPIHT and up to 10 times less memory than H.264).

7.2 Future work

There are several open problems and more work to be done related with the subject of this thesis. Future work includes:

- Bio-inspired quantization in order to improve the visual subjective quality of the proposed image encoders.
- Regarding sign coding, we are planning to use genetic algorithms or data mining to separate all neighbor sign combinations (K) in the two contexts used by the S-LTW codec to encode the sign.
- Presently there is an integer implementation of the INTRA video encoder M-LTW, but it is implemented using 32-bit integer precision. We want to obtain an implementation using 16-bit integer precision, reducing by one-half the total amount of memory needed by the encoder.
- With respect to 3D video encoding, we have just began our research in this field, so we are planning to introduce an ME stage just before applying the temporal filter to the wavelet transform.
- Also, related with 3D video encoding, we wanted to develop an encoder stage for the frame-by-frame 3D-DWT using the LTW coding engine so as to improve the coding efficiency. Furthermore, a rate control stage for the encoder stage could be developed.

7.3 Publications resulting from this thesis

Some fragments of the work presented in this thesis have been published in proceedings of both international and national conferences and in research journals. In particular, the main contributions are:

→ From Chapter 2:

- **A Heuristic Bitrate Control for Non-embedded Wavelet Image Encoders**
  O. López, M. Martinez-Rach, J. Oliver, M.P. Malumbres
  48th International Symposium ELMAR-2006 (ELMAR 2006), Zadar (Croatia), June 2006, *ISBN:953-7044-03-3*, pp. 13-16.
  This paper presents a preliminary study about rate control for non-embedded wavelet image encoders. In this paper, several rate control algorithms for non-embedded image encoders are presented. They correspond to the ones presented in this thesis in Sections 2.1, 2.2 and 2.3.

- **Analysis of fast Bitrate Control Algorithms for Non embedded Wavelet Image Encoders**
  O. López, M. Martinez-Rach, J. Oliver, M.P. Malumbres
  XVII Jornadas de Paralelismo, Albacete (Spain), September 2006 *84-690-0551-0*, pp. 491-496.
  In this paper, the algorithms presented in the previous paper are explained in greater detail.

- **Impact of Rate Control Tools on very fast Non-Embedded Wavelet Image Encoders**
  O. López, M. Martinez-Rach, M.P. Malumbres and J.Oliver
  Visual Communications and Image Processing, San Jose (USA), January 2007 *ISBN:0277-786X*, pp. 291-298.
  In this paper, the rate control algorithms are evaluated in detail and a further evaluation is performed when implemented on the LTW encoder, including both an objective and a subjective evaluation. The paper corresponds to the results presented in Section 2.5.

→ From Chapter 3:

- **Embedded Lower Tree Wavelet Encoder**
  O. Lopez; M. Martinez-Rach; P. Piñol; J. Oliver; M. P. Malumbres
  XIX Jornadas de Paralelismo, Castellon (Spain), 2008, *ISBN:978-84-8021-676-0*.
  This paper shows a preliminary study of sign coding in non-embedded image encoders. This preliminary results are related to the sign coding neighborhood presented in Section 3.1.

→ From Chapter 4:

- **E-LTW: An Enhanced LTW Encoder with Sign Coding and precise Rate Control**
  O. Lopez; M. Martinez-Rach; P. Piñol; J. Oliver; M. P. Malumbres
  International Conference on Image Processing, Cairo (Egypt), November 2009.
  This paper presents the improved E-LTW image encoder which includes a sign coding stage and a precise rate control algorithm.

- **Enhanced Non-embedded Lower Tree Wavelet Encoder**
  O. Lopez; M. Martinez-Rach; P. Piñol; J. Oliver; M. P. Malumbres
  XX Jornadas de Paralelismo, A Coruña (Spain), September 2009.

In this paper we present an improved version of the LTW encoder called E-LTW which includes an exact rate control and also a sign coding stage. This E-LTW encoder is presented in Chapter 4.

→ From Chapter 5:

- **Motion-LTW: a fast and efficient intra video coding system with low memory consumption**
  Otoniel López, Miguel Martínez, Pablo Piñol, Manuel P. Malumbres, J. Oliver
  XVIII Jornadas de Paralelismo, 2007, Zaragoza (Spain), *ISBN:-*978-84-9732-672-8.
  In this paper we present a new INTRA video encoder that it is able to encode an ITU D1 size image in real time with good R/D behavior. In this thesis this new encoder is presented in Chapter 5

- **M-LTW: A Fast and Efficient Non-Embedded Intra Video Codec**
  O. López, M. Martinez-Rach, P. Piñol, J. Oliver, M. Malumbres
  Pacific-Rim Conference on Multimedia, 2007, Hong Kong (China), *ISBN:*978-3-540-77254-5, pp. 830-838.
  In this paper we explain in depth the M-LTW encoder presented in Chapter 5.

- **M-LTW: A fast and efficient intra video codec**
  Otoniel M. López; Miguel O. Martínez-Rach, Pablo Piñol, Manuel Perez Malumbres, José Oliver
  Signal Processing: Image Communication Ed. ELSEVIER, *doi:10.1016/j.image.2008.07.001*, pp. 637-648.
  In this paper we extended the evaluation performed in previous papers.

→ From Chapter 6:

- **A General Frame-by-Frame Wavelet Transform Algorithm for a Three-Dimensional Analysis with Reduced Memory Usage**
  Oliver, J.; Lopez, O.; Martinez-Rach, M.; Malumbres, M.P.
  IEEE International conference on image processing (ICIP), *doi:10.1109/ICIP.2007.4378993*, Vol 1. pp. 469-472.
  In this paper, a novel way of computing the 3D-DWT is presented.

→ Others:

- **International Standards for Image Compression**
  J. Oliver, O. López, M. Martínez-Rach, P. Piñol, C.T. Calafate, M.P. Malumbres
  Ed. IGI-Global, *ISBN:978-1-60566-026-4*, pp. 5270.

- **An Study of Objective Quality Assessment Metrics for Video Codec Design and Evaluation**
  M. Martinez-Rach, O. López, P. Piñol, M.P. Malumbres and J. Oliver
  IEEE Int. Symp. on Multimedia (ISM 2006), San Diego (USA), *0-7695-2746-9*, pp. 517-524.

- **PSNR vs. quality assessment metrics for image and video codec performance evaluation**
  M. Martinez-Rach, O. López, P. Piñol, M.P. Malumbres, J. Oliver
  XVIII Jornadas de Paralelismo, Zaragoza (Spain), 2007 *ISBN:978-84-9732-672-8*.

- **A practical study of video streaming over IEEE 802.11 wireless networks using DirectShow video encoders**
  Pablo Piñol, Miguel Martínez, Otoniel López, M.P. Malumbres, J. Oliver
  XVIII Jornadas de Paralelismo, Zaragoza (Spain), 2007 *ISBN:978-84-9732-672-8*.

- **Low-Complexity TTCM Based Distributed Video Coding Architecture**
  J.L. Martínez, W.A.C. Fernando, W.A.R.J. Weerakkody, J. Oliver, O. López, M. Martinez-Rach, M.P. Malumbres, P. Cuenca and F. Quiles
  Second Pacific Rim Symposium (PSIVT) 2007, Santiago (Chile), *ISBN:*978-3-540-77128-9, Vol. 4872.

- **Quality Assessment Metrics vs. PSNR under Packet Loss Scenarios in MANET Wireless Networks**
  M. Martinez-Rach, O. Lopez, P. Piñol, M.P. Malumbres, J. Oliver, Carlos T. Calafate
  ACM MM 2007 Augsburg, Bavaria (Germany), *ISBN:*978-1-59593-702-5, pp. 31-36.

Appendix I

Acronyms

bpp bits per pixel

DCT Discrete Cosine Transform

DWT Discrete Wavelet Transform

EBCOT Embedded Block Coding with Optimized Truncation

ECECOW Embedded Conditional Entropy Coding of Wavelet Coefficients

EZW Embedded Zero-tree Wavelet

EOF End of Frame

FGS Fine Grain Scalability

GIS Geographic Information System

GOP Group of Pictures

HQ High Quality

ITU International Telecommunication Union

LSB Least Significant Bit

LSP List of Significant Pixels

LIP List of Insignificant Pixels

LIS List of Insignificant Sets

LTW Lower Tree Wavelet

LZC Layered Zero Coding

MSB Most Significant Bit

MSE Mean Squared Error

MC Motion Compensation

MCTF Motion Compensation Temporal Filtering

ME Motion Estimation

MV Motion Vector

PDA Personal Digital Assistant

PDF Portable Document Format

PCRD Post-Compression Rate Distortion

SBHP Subband-Block Hierarchical Partitioning

SFQ Space-Frequency Quantization

SNR Signal to Noise Ratio

SPIHT Set Partitioning In Hierarchical Trees

SSD Solid State Drives

VLC Variable Length Coding

VIF Visual Information Fidelity

WT Wavelet Transform

Appendix II

Kodak images set

Img01 Img02 Img03

Img04 Img05 Img06

Img07 Img08 Img09

Img10 Img11 Img12

Figure II.1: Kodak image set (768x512)


Img13


Img14


Img15


Img16


Img17


Img18


Img19


Img20


Img21


Img22


Img23

Figure II.2: Kodak image set (768x512)

Appendix III

Test images

Barbara(512x512)

Lena(512x512)

Boat(512x512)

GoldHill(512x512)

Mandrill(512x512)

Peppers(512x512)

Figure III.1: Test image set

Zelda(512x512)

Bike(2048x2560)

Cafe(2048x2560)

Woman(2048x2560)

Figure III.2: Test image set

Appendix IV

Test Videos

Foreman(QCIF and CIF)

News(QCIF and CIF)

Container(QCIF and CIF)

Coastguard(QCIF and CIF)

Mobile(ITU)

Station2(HD)

Figure IV.1: Test video sequences set

Bibliography

[1] ISO/IEC 14496-10 and ITU Rec. H.264. Advanced video coding, 2003.

[2] A. Aaron, R. Zhang, and B. Girod. Wyner-ziv coding of motion video. In *Signals, Systems and Computers Conference*, volume 1, pages 240–244, November 2002.

[3] T. Acharya and P. Tsai. *JPEG 2000 Standard for Image Compression: Concepts, Algorithms and VLSI Arquitectures*, chapter 5. Wiley, October 2005.

[4] M.D. Adams and F. Kossentini. Reversible integer-to-integer wavelet transforms for image compression: performance evaluation and analysis. *IEEE Transaction on Image Processing*, 9(6), June 2000.

[5] Adobe. Tag based image file format, revision 6.0, http://www.adobe.com/support/technotes.html/, June 1992.

[6] M. Albanesi and S. Bertoluzza. Human vision model and wavelets for high-quality image compression. In *International Conference in Image Processing and its Applications*, July 1995.

[7] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Transaction on Image Processing*, 1(2):205–220, 1992.

[8] M. Aviles, F. Moran, and N. Garcia. Progressive lower trees of wavelet coefficients: Efficient spatial and SNR scalable coding of 3D models. *Lecture Notes in Computer Science*, 3767:61–72, 2005.

[9] G. Bernabe, J. Gonzalez, and J.M. Garcia. Memory conscious 3D wavelet transform. In *Euromicro Conference*, September 2002.

[10] G. Björntegaard and K. Lillevold. Content-adaptative VLC coding and coefficients. Technical Report JVT-C028, Joint Video Team (JVT), May 2002.

[11] C. Cai and R. Ding. Rate control using significant coefficient percentage. In *IEEE International Conference on Signal Processing*, volume 1, pages 873–876, Beijing, China, 2002.

[12] A.R. Calderbank, I. Daubechies, W. Sweldens, and B-L. Yeo. Wavelet transforms that map integers to integers. *Applied and Computational Harmonic Analysis*, 5(3):332–369, July 1998.

[13] P. Campisi and A. Neri. Video watermarking in the 3D-DWT domain using perceptual masking. In *IEEE International Conference on Image Processing*, pages 997–1000, September 2005.

[14] D.M. Chandler and S. Hemami. Contrast-based quantization and rate control for wavelet-coded images. In *IEEE International Conference on Image Processing*, volume 3, pages 233–236, Rochester, New York, 2002.

[15] P. Cheng and J.W.Woods. Bidirectional MC-EZBC with lifting implementation. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1183–1194, October 2004.

[16] Yushin Cho, W.A. Pearlman, and A. Said. Low complexity resolution progressive image coding algorithm: PROGRESS (progressive resolution decompression). In *IEEE International Conference on Image Processing*, September 2005.

[17] S.J. Choi and J.W.Woods. Motion-compensated 3-d subband coding of video. *IEEE Transactions on Image Processing*, 8(2):155–167, February 1999.

[18] C. Chrysafis and A. Ortega. Line-based, reduced memory, wavelet image compression. *IEEE Transactions on Image Processing*, 9(3):378–389, March 2000.

[19] CIPR. http://www.cipr.rpi.edu/resource/stills/kodak.html. Center for image processing research.

[20] R.R. Coifman and M.V. Wickerhauser. Entropy-based algorithms for best basis selection. *IEEE Transactions on Information Theory*, 38:713–718, 1992.

[21] T.M. Cover and J.A. Thomas. *Elements of information theory*. Wiley Series in Communications, 1991.

[22] I. Daubechies and W. Sweldens. Factoring wavelet transforms into lifting steps. *Journal of Fourier analysis and applications*, 4(3):247–268, 1998.

[23] Philip J. Davis. *Interpolation and Approximation*. Dover Publications, June 1975.

[24] Aaron Deever and Sheila S. Hemami. What's your sign?: Efficient sign coding for embedded wavelet image coding. In *Proc. IEEE Data Compression Conf., Snowbird, UT*, pages 273–282, 2000.

[25] P.L. Dragotti and G. Poggi. Compression of multispectral images by three-dimensional SPITH algorithm. *IEEE Transactions on Geoscience and Remote Sensing*, 38(1):416–428, January 2000.

[26] H. Everett. Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, 11:399–417, 1963.

[27] M. Gaubatz and S.S. Hemami. Fast, accurate rate-control for low-rate wavelet-based image coding via bootstrapping with local probability models. In *IEEE International Conference on Signal Processing*, Genoa, Italy, 2005.

[28] M.J. Gormish and J. T. Gill. Computation-rate-distortion in transform coders for image compression. In *SPIE Visual Communications and Image Processing*, volume 1903, pages 146–152, 1993.

[29] M. Grangetto, E. Magli, M. Martina, and G. Olmo. Optimization and implementation of the integer wavelet transform for image coding. *IEEE Transaction on Image Processing*, 11(6):596–604, 2002.

[30] S. Grottke, T. Richter, and R. Seiler. Apriori rate allocation in wavelet-based image compression. *Second International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution, 2006. AXMEDIS '06.*, pages 329–336, Dec. 2006.

[31] A. Hallapuro, M. Karczewicz, and H. Malvar. Low complexity transform and quantization - part i: Basic implementation. Technical report, Tech. Report JVTB038, Joint Video Team (JVT), February 2002.

[32] C.R. Hauf and J.C. Houchin. The FlashPix(TM) image file format. In *Fourth Color Imaging Conference: Color Science, Systems and Applications*, pages 234–238, November 1996.

[33] Z. He and S. Mitra. A unified rate-distortion analysis framework for transform coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(12):1221–1236, December 2001.

[34] M.L. Hilton, B.D. Jawerth, and A. Sengupta. Compressing still and moving images with wavelets. *Multimedia Systems*, 2, 1994.

[35] E.S. Hong and R.E. Ladner. Group testing for image compression. *IEEE Transactions on Image Processing*, 11:901–911, August 2002.

[36] D.A. Huffman. A method for the construction of minimum redundancy codes. In *IRE 40*, pages 1098–1101, 1952.

[37] ISO/IEC 10918-1/ITU-T Recommendation T.81. Digital compression and coding of continuous-tone still image, 1992.

[38] ISO/IEC 15444-1. JPEG2000 image coding system, 2000.

[39] ISO/IEC JTC1. ISO/IEC 11172-2. Coding of moving pictures and associated audio for digital storage media at up to about 1.5 mbit/s, 1993.

[40] ISO/IEC JTC1. ISO/IEC 13818-2. Generic coding of moving pictures, 2000.

[41] ISO/IEC JTC1. ISO/IEC 14496-2. Coding of audio-visual objects, April 2001.

[42] ITU-T Recommendation H.261. Video codec for audiovisual services at p x 64 kbits/s, March 1999.

[43] ITU-T Recommendation T.4. Standardization of group 3 fascimile apparatus for document transmission, 1993.

[44] R.L. Joshi, V.J. Crump, and T.R. Fischer. Image subband coding using arithmetic coded trellis coded quantization. *IEEE Transactions on Circuits and Systems for Video technology*, 5, December 1995.

[45] Software Kakadu. http://www.kakadusoftware.com.

[46] M. Karczewicz and R. Kurceren. A proposal for SP-frames. Technical report, VCEG-L27, ITU-T SG 16/6, January 2001.

[47] M. Karczewicz and R. Kurceren. The SP and SI frames design for H.264/AVC. *IEEE Transactions on Circuits and System for Video Technology*, 13:637–644, 2003.

[48] G. Karlsson and M. Vetterli. Three-dimensional subband coding of video. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 1100–1103, April 1988.

[49] J.H. Kasner, M.W. Marcellin, and B.R. Hunt. Universal trellis coded quantization. *IEEE Transactions on Image Processing*, 8(12):1677–1687, Dec 1999.

[50] B.J. Kim and W.A. Pearlman. An embedded wavelet video coder using three-dimensionnal set partitionning in hierarchical trees (SPIHT). In *Data Compression Conference*, pages 251–260, March 1997.

[51] B.J. Kim, Z. Xiong, and W.A. Pearlman. Low bit-rate scalable video coding with 3D set partitioning in hierarchical trees (3D SPIHT). *IEEE Transactions on Circuits and Systems for Video Technology*, 10:1374–1387, December 2000.

[52] P. Lancaster. *Curve and Surface Fitting: An Introduction*. Academic Press, September 1986.

[53] G.G. Langdon. An introduction to arithmetic coding. Technical report, IBM J. Res. Develop. 28:135-149, 1984.

[54] M. Liu. Overview of the p * 64 kbits/s video coding standard. *Communications ACM*, 34(4):60–63, April 1991.

[55] S. Mallat and F. Falzon. Analysis of low bit rate transform coding. *IEEE Transactions on Image Processing*, 46(4):1027–1042, April 1998.

[56] Stephane Mallat and Sifen Zhong. Characterization of signals from multiscale edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(7):710–732, July 1992.

[57] M.W. Marcellin, M.A. Lepley, A. Bilgin, T.J. Flohr, T.T. Chinen, and J.H. Kasner. An overview of quantization in JPEG2000. *Signal Processing: Image Communication*, 17, 2002.

[58] D. Marpe, G. Blättermann, and T. Wiegand. Adaptive codes for H.26L. Technical Report VCEG-L13, ITU-T SG16/6, Eibsee, Germany, January 2001.

[59] D. Marpe and H. Cycon. Very low bit-rate video coding using wavelet-based techniques. *IEEE Transactions on Circuits and Systems for Video Technology*, 9:85–94, February 1999.

[60] J.L. Martinez, W. Fernando, W. Weerakkody, J. Oliver, O. Lopez, M. Martinez-Rach, P. Cuenca, and F.J. Quiles. Transform domain wyner-ziv codec based on turbo trellis codes modulation. In *Lecture Notes in Computer Science (LNCS)*, volume 4872, pages 841–852, 2007.

[61] J.L. Martinez, W. Weerakkody, P. Cuenca, F.J. Quiles, and W. Fernando. Transform domain Wyner-Ziv codec based on turbo trellis codes modulation. In *Lecture Notes in Computer Science (LNCS)*, volume 4903, pages 413–424, 2007.

[62] M. Martinez-Rach, O. Lopez, P. Piñol, J. Oliver, and M.P. Malumbres. A study of objective quality assessment metrics for video codec design and evaluation. *IEEE International Symposium on Multimedia*, pages 517–524, 2006.

[63] F.G. Meyer, A.Z. Averbuch, and J.O. Strömberg. Fast adaptive wavelet packet image compression. *IEEE Transactions on Image Processing*, 9:792–800, May 2000.

[64] G. M. Morton. A computer oriented geodetic data base and a new technique in file sequencing. Technical report, IBM Ltd., 1966.

[65] E. Moyano, F.J. Quiles, A. Garrido, L. Orozco-Barbosa, and J. Duato. Efficient 3D wavelet transform decomposition for video compression. In *Int. Work. Digital and Computational Video*, 2001.

[66] D. Mukherjee and S.K. Mitra. Successive refinement lattice vector quantization. *IEEE Transactions on Image Processing*, 11:1337–1348, December 2002.

[67] Y. Nian, L. Wu, S. He, and Y. Gu. A new video coding based on 3D wavelet transform. In *IEEE International Conference on Intelligent Systems Design and Applications*, October 2006.

[68] J.R. Ohm. Advanced packet-video coding based on layered VQ and SBC techniques. *IEEE Transactions on Circuits and Systems for Video Technology*, 3(3):208–221, June 1994.

[69] J.R. Ohm. Three dimensional subband coding with motion compensation. *IEEE Transactions on Image Processing*, 3(5):559–571, September 1994.

[70] J. Oliver and M. P. Malumbres. Fast tree-based wavelet image coding with efficient use of memory. *Visual communications and image processing*, 5960, 2005.

[71] J. Oliver and M. P. Malumbres. Low-complexity multiresolution image compression using wavelet lower trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(11):1437–1444, 2006.

[72] J. Oliver, E. Oliver, and M.P.Malumbres. On the efficient memory usage in the lifting scheme for the two-dimensional wavelet transform computation. In *IEEE International Conference on Image Processing*, pages 485–488, September 2005.

[73] C. Parisot, M. Antonini, and M. Barlaud. Stripe-based MSE control in image coding. In *IEEE International Conference on Image Processing*, volume 1, pages 649–652, Rochester, New York, 2002.

[74] W.A. Pearlman. Trends of tree-based, set partitioning compression techniques in still and moving image systems. In *Picture Coding Symposium*, pages 1–8, April 2001.

[75] W.A. Pearlman, A. Islam, N. Nagaraj, and A. Said. low-complexity image coding with a set-partitioning embedded block coder. *IEEE Transactions on Circuits and Systems for Video technology*, 14:1219–1235, November 2004.

[76] W. Pennebaker and J. Mitchell. *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, 1994.

[77] B. Pesquet-Popescu and V. Bottreau. Three-dimensional lifting schemes for motion compensated video compression. *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1793–1796, 2001.

[78] R. Puri and K. Ramchandran. A new robust video coding architecture based on distributed compression principles. In *Allerton Conference*, October 2002.

[79] M. Rabbani and R. Joshi. An overview of the JPEG2000 still image compression standard. *Signal Processing: Image Communication*, 17, 2002.

[80] N.M. Rajpoot, R.G. Wilson, F.G. Meyer, and R.R. Coifman. Adaptive wavelet packet basis selection for zerotree image coding. *IEEE Transactions on Image Processing*, 12:1460–1472, December 2003.

[81] K. Ramchandran and M. Vetterli. Best wavelet packet bases in a rate-distortion sense. *IEEE Transactions on Image Processing*, 2:160–175, 1993.

[82] I.E.G. Richardson. *H.264 and MPEG-4 Video Compression*. John Wiley and Sons Ltd, 2003.

[83] J. Rissanen and G.G. Langdon. Arithmetic coding. Technical report, IBM J. Res. Develop. 23:149-162, 1979.

[84] A. Said and A. Pearlman. 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), June 1996.

[85] Amir Said. Comparative analysis of arithmetic coding computational complexity. Technical report, Hewlett-Packard Laboratories HPL-2004-75, 2004.

[86] P. Schelkens, A. Munteanu, J. Barbariend, M. Galca, X. Giro-Nieto, and J. Cornelis. Wavelet coding of volumetric medical datasets. *IEEE Transactions on Medical Imaging*, 22(3):441–458, March 2003.

[87] M. Schindler. Huffman coding. http:// www.compressconsult.com/, October 1998.

[88] Edward L. Schwartz, Ahmad Z, and Martin Boliek. CREW: Compression with reversible embedded wavelets. In *In Proc SPIE*, pages 212–221, 1995.

[89] A. Secker and D. Taubman. Motion-compensated highly scalable video compression using an adaptive 3D wavelet transform based on lifting. *IEEE Internantional Conference on Image Processing*, pages 1029–1032, October 2001.

[90] A. Secker and D. Taubman. Highly scalable video compression using a lifting-based 3D wavelet transform with deformable mesh motion compensation. *IEEE Internantional Conference on Image Processing*, pages 749–752, September 2002.

[91] A. Secker and D. Taubman. Highly scalable video compression with scalable motion coding. *IEEE Internantional Conference on Image Processing*, September 2003.

[92] A. Secker and D. Taubman. Lifting-based invertible motion adaptive transform (LIMAT) framework for highly scalable video compression. *IEEE Transactions on Image Processing*, 12(12):1530–1542, December 2003.

[93] J.M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12), December 1993.

[94] J.M. Shapiro. A fast technique for identifying zerotrees in the EZW algorithm. *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 3:1455–1458, 1996.

[95] H.R. Sheikh and A.C. Bovik. Image information and visual quality. *IEEE Transaction on Image Processing*, 15(2):430–444, February 2006.

[96] Y. Shoham and A. Gersho. Efficient bit allocation for an arbitrary set of quantizers. *IEEE Transactions on Acoustic, Speech, Signal Processing*, 36:1445–1453, September 1984.

[97] T. Sikora. MPEG digital video-coding standard. *IEEE Signal Processing Magazine*, September 1997.

[98] E.A.B. Da Silva, D.G. Sampson, and M. Ghanbari. A successive approximation vector quantizer for wavelet transform image coding. *IEEE Transactions on Image Processing*, 5:299–310, February 1996.

[99] M.J. Slattery and J.L. Mitchell. The qx-coder. *IBM Journal of Research and Development*, 42:767–784, November 1998.

[100] N. Sprljan, S. Grgic, M. Mrak, and M. Grgic. Modified SPIHT algorithm for wavelet packet image coding. In *International Symposium on Video/Image Processing and Multimedia Communications (VIPromCom)*, 2002.

[101] W. Sweldens. The lifting scheme: a custom-design construction of biorthogonal wavelets. *Applied and Computational Harmonic Analysis*, 3(2):186–200, April 1996.

[102] Table Curve 3D 3.0. http://www.systat.com. Systat Software Inc.

[103] X. Tang, S. Cho, and W.A. Pearlman. Comparison of 3D set partitioning methods in hyperspectral image compression featuring an improved 3D-SPIHT. In *Data Compression Conference*, March 2003.

[104] D. Taubman. High performance scalable image compression with EBCOT. *IEEE Transactions on Image Processing*, 9(7):1158–1170, July 2000.

[105] D. Taubman and A. Zakhor. Multirate 3-D subband coding of video. *IEEE Transactions on Image Processing*, 3(5):572–588, September 1994.

[106] D.S. Taubman and M.W. Marcellin. *JPEG 2000: Image Compression Fundamentals, Standards and Practice*, pages 262–281. Kluwer Academic Publishers, 2002.

[107] Information technology. coded representation of picture and audio information - progressive bi-level image compression. Technical report, Tech. Report T.82 (JBIG), ITU-T Recommendation.

[108] J. Tham, S. Ranganath, and A. Kassim. Highly scalable wavelet-based video codec for very low bit-rate environment. *IEEE Journal on Selected Areas in Communications*, 16:12–27, January 1998.

[109] Y. Wang and C. Wu. Low complexity multiple description coding method for wireless video. In *Advanced Information Networking and Applications Conference*, volume 2, pages 95–98, March 2005.

[110] T. Wiegand, G.J. Sullivan, G. Bjntegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, July 2003.

[111] I.H. Witten, R.M. Neal, and J.G. Cleary. Arithmetic coding for data compression. *Commun. ACM*, 30(6):520–540, 1987.

[112] J.W. Woods and G. Lilienfield. A resolution and frame-rate scalable subband/wavelet video coder. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1035–1044, September 2001.

[113] J.W. Woods and S. O'Neil. Subband coding of images. *IEEE Transactions on Acoustic, Speech, Signal Processing*, 34:1278–1288, October 1986.

[114] X. Wu. High-order context modeling and embedded conditional entropy coding of wavelet coefficients for image compression. In *Proc. of 31st Asilomar Conf. on Signals, Systems, and Computers*, pages 1378–1382, 1997.

[115] Z. Xiong, K. Ramchandran, and M. Orchard. Space-frequency quantization for wavelet image coding. *IEEE Transactions on Image Processing*, 6(5):677–693, May 1997.

[116] Z. Xiong, K. Ramchandran, and M.T. Orchart. Wavelet packet image coding using space-frequency quantization. *IEEE Transactions on Image Processing*, 7:892–898, June 1998.

[117] W. Zeng, S. Daly, and S. Lei. An overview of the visual optimization tools in JPEG2000. *Signal Processing: Image Communication*, 17, 2002.

[118] N.P. Zhidkov and G.M. Kobelkov. *Numerical methods*. Nauka Moscow, 1987.