



Wavelet-based video compression: A glimpse of the future?

by

Are Nor and Lina L. Kittilsen

**Masters Thesis in
Information and Communication Technology**

**Agder University College
Faculty of Engineering and Science**

**Dunedin, New Zealand
June 2004**

Abstract

Even though wavelet-based video compression has been an area of research for more than a decade, motion estimation and compensation has been considered complex and inefficient until recently. We have carried out a thorough investigation of existing research work in this field, and found that fundamental problem with wavelet-based temporal removal while obtaining highly scalability (the wavelet-properties of multiresolution structure in combination with embedded coding), has been solved by performing motion compensated temporal filtering within the wavelet domain of a overcomplete three-dimensional lifting-based wavelet transform, and that wavelet-based video CODECs now can compete with DCT-based video CODERS. In addition we have designed and implemented a video compression prototype founded on the Java Media Framework API and a DCT- and DWT-based still-image compression application developed by four engineer students in a previous project. We carefully planned a stepwise implementation progress, to help us making the prototype extendable. The entire plan has not been implemented. However, this was never the intention. We wanted to make a foundation that could be utilized in future studies, both for our one interest and others. Only a so-called intra-frame CODEC has been implemented, and has a lot of potential for further development.

Preface

In December 2003 we decided to write a master thesis on use of wavelets in video compression. Several wavelet theses regarding medical image analysis was presented by Per Henrik Hogstad (HiA), but because of the importance of video compression we decided on this. Per Henrik Hogstad have given us several lectures in Linear algebra, Fourier and Wavelet theory, the first was held in Grimstad, but because we have been writing our thesis in New Zealand, he has provided us with wavelet lectures via MSN messenger.

We would like to thank the library manager at Agder University College, Birger Kvamme for providing us with books and introducing the invaluable IEEE database to us. We also want to thank our mentor, Per Henrik Hogstad for providing us with wavelet lectures via MSN messenger at all times of the day. We have appreciated his educational expertise, and found his knowledge of wavelets essential for our understanding of wavelets. Special thanks is given to foreign student advisor Dr. Willem Labuschagne, head of IT-department Dr. Ian McDonald, and senior technician Allan Hayes at University of Otago, Dunedin, New Zealand, for making us feel welcome, and by giving us the chance to write our master thesis at the venerable southernmost university in the world. Dr. Zhiyi Huang has functioned as our informal mentor, and has been most helpful in several phases of our thesis.

Contents

Part 1 Fundamentals – Background material	8
1 Introduction	9
1.1 Problem Definition and Motivation	10
1.2 Preliminary Survey of Previous Work	10
1.3 Prototype.....	11
2 Methodology	11
2.1 Literature References	11
2.2 Test Methods.....	12
2.3 Other issues.....	12
3 Fundamentals of Video Compression	12
3.1 Introduction	12
3.2 Digital Video	13
3.3 What is a Video CODEC?	14
3.4 Removal of Spectral Redundancy	15
3.5 Removal of Spatial Redundancy	15
3.5.1 Transform Coding	15
3.5.2 The Discrete Cosine Transform (DCT)	16
3.5.3 Quantization	16
3.6 Removal of Temporal Redundancy	17
3.6.1 Three-dimensional transform coding	18
3.6.2 Motion Estimation (ME) and Compensation (MC)	18
3.6.3 Block matching algorithm (BMA)	20
3.6.4 Phase Correlation	21
3.6.5 Enhancements to the motion model.....	22
3.7 Removal of Statistical Redundancy	23
3.7.1 Entropy Encoding	23
3.7.2 Run-Length Encoding (RLE).....	23
3.7.3 Huffman Coding	24
3.8 Video Coding Standards	24
4 Fundamentals of Wavelets	25
4.1 Introduction	25
4.2 Wavelets from a Historical Perspective	25
4.3 Fourier vs. Wavelet	27
4.4 Wavelets Concept	28
4.5 Wavelets as Filters	29
4.6 The Wavelet Transform in Two Dimensions	31
4.7 Inverting the Wavelet Transform.....	34
Part 2 Trends in Wavelet-Based Video Compression Research	35
5 Introduction	35
5.1 Existing Wavelet-Based Video CODECs.....	35
5.2 Who Does What with Wavelet	35
5.3 What to Look for.....	36
6 Discrete Wavelet-based Transforms	36
6.1 The Wavelet Domain	36
6.2 The Fast Lifting Wavelet Transform (The Lifting Scheme)	38
6.2.1 Constructing wavelets with the lifting scheme	39
6.3 Three-Dimensional Wavelet Transform (3-D DWT).....	40
6.3.1 3-D SPIHT (Set Partitioning In Hierarchical Trees)	40
6.3.2 Lifting-based Invertible Motion Adaptive Transform (LIMAT)	41
6.4 The Dual-Tree Complex Wavelet Transform (DT CWT).....	42

6.4.1	The Dual-Tree Implementation	43
6.4.2	Key features	44
6.4.3	Applications	46
6.5	The Overcomplete Discrete Wavelet Transform (ODWT)	46
6.5.1	ODWT in video coders	46
7	Wavelet-based Spatial Compression	47
7.1	Prosperity of Wavelet-based Image Coders	47
7.2	Embedded Zero-tree Wavelet (EZW) Compression	48
7.3	Set Partitioning In Hierarchical Trees (SPIHT)	49
7.4	EBCOT Coder in JPEG2000	50
7.5	Adaptive Wavelet Coding of Multimedia Images	51
8	Wavelet-based Temporal Compression	51
8.1	Prosperity of Wavelet-based ME/MC algorithms in Video Coders	51
8.2	Wavelet-based Motion Estimation and Compensation in Spatial Domain ...	52
8.3	Motion Estimation and Compensation in Wavelet Domain	53
8.3.1	Shift Dependence	53
8.4	Motion-Compensated Temporal Filtering (MCTF)	56
Part 3 Java Prototype – MediaCODEC		57
9	Foundation	58
9.1	The Java Programming Language	58
9.2	Java Media Framework (JMF)	59
9.3	The Still-Image Compression Application	59
9.3.1	Converting colour space from RGB to YCbCr	60
9.3.2	Transforming and quantizing the image	65
9.3.3	Entropy encoding	66
10	Prototype Design	66
10.1	A coarse CODEC sketch	67
10.2	Layout and User Interaction	68
10.3	File Format	70
10.4	MDWT (Motion Discrete Wavelet Transform) CODEC	71
10.5	MEWT (Motion Estimation Wavelet Transform) CODEC	74
10.6	The Programming Progression Plan	74
11	Implementation and Development	75
12	User Guide	76
12.1	The File Menu	76
12.2	The Open Video Menu Item	78
12.3	The View Menu	80
12.4	The Window Menu	80
Part 4 Conclusion		81
13	Experimental Testing and Comparisons	82
13.1	Testing of Image- and Video CODEC algorithms in VcDemo	82
13.2	Comparison of Phase-Correlated and Block-based Motion Estimation	84
14	Conclusion	85
14.1	Further Development of the Prototype	85
References		87

List of Figures

Figure 3.1	Digital video	13
Figure 3.2	Projection of 3-D scene onto a video image	13
Figure 3.3	Spatial and temporal sampling	13
Figure 3.4	Architecture of a typical video encoder	14

Figure 3.5 'Roadmap' - A typical video coder.....	15
Figure 3.6 'Roadmap' - A typical video coder.....	15
Figure 3.7 Cosine Wave	16
Figure 3.8 Sine wave	16
Figure 3.9 Zig-zag scan of coefficients.....	17
Figure 3.10 'Roadmap' - A typical video coder	17
Figure 3.11 First four frames of the "Foreman" sequence.	18
Figure 3.12 First four frames of the "Football" sequence.	18
Figure 3.13 Predicted motion vectors for frame 2 of the 'Foreman' sequence	19
Figure 3.14 Residual frame examples from VCDemo.....	20
Figure 3.15 The block matching algorithm	20
Figure 3.16 Half-pixel interpolation	22
Figure 3.17 'Roadmap' - A typical video coder	23
Figure 4.1 Wavelet history: Timeline 1807 – 1984 [18]	26
Figure 4.2 Wavelet history: Timeline 1985 – 1999 [18]	27
Figure 4.3 Fourier basis functions	27
Figure 4.4 Wavelet basis functions.....	27
Figure 4.5 Wavelet can be viewed as a burst of energy with dominant frequency....	28
Figure 4.6 The discrete version of figure 1.3	28
Figure 4.7 (a) Convolution of a five-sample wavelet, W , with the samples of a signal, S . The wavelet is operating on S_7 . [7, slightly modified]	29
Figure 4.8 Additional samples must be created at the edges.....	29
Figure 4.9 The five-sample wavelet shown as a five-tap filter [7].	29
Figure 4.10 Some scaling functions from the biorthogonal Deslauriers-Dubuc family. a (2,2), b (4,2), c (6,2) and d (2,4). [19]	30
Figure 4.11 Some wavelets from the biorthogonal Deslauriers-Dubuc family. a (2,2), b (4,2), c (6,2) and d (2,4). [19].....	30
Figure 4.12 The first stage of the wavelet transform.....	30
Figure 4.13 The first level of the DWT, including downsampling by 2	31
Figure 4.14 A two level wavelet transform	31
Figure 4.15 Division of the image into four subimages [7, 8]	32
Figure 4.16 (a) The original image (Lena) [15].....	33
Figure 4.17 The first two levels of the inverse transform.....	34
Figure 6.1 'Roadmap' – Architecture of a typical wavelet-based video coder	36
Figure 6.2 Synthesis scaling and wavelet functions, 5/3 and 9/7 subband filter sets	37
Figure 6.3 Lifting scheme forward WT	38
Figure 6.4 Lifting scheme inverse WT	38
Figure 6.5 Forward steps [15].....	39
Figure 6.6 Inverse steps [15].....	39
Figure 6.7 Predicting and updating coefficients in 3 levels [15]	39
Figure 6.8 Split, Predict, and Update in 2 levels [15, modified]	40
Figure 6.9 3-D Wavelet Decomposition [35].....	41
Figure 6.10 Temporal decomposition of a group of frames (GOF) [35]	41
Figure 6.11 a) Lifting representation for the Haar temporal transform, b) same, but with motion compensated lifting steps.....	42
Figure 6.12 Dual tree of real filters for the Q-shift CWT [49].....	44
Figure 6.13 Wavelet and scaling function components of 16 shifted step functions for the Q-shift DT CWT (a) and real DWT (b) [49]	45
Figure 7.1 'Roadmap' - Architecture of typical image- and video coders	47
Figure 7.2 Relationship between parent-child regions in (2-D) DWT sub bands.....	48
Figure 7.3 Non-embedded vs. embedded bit stream.....	49
Figure 7.4 Example of ROI coding at 0.125 b/pixel [24].....	50
Figure 8.1 'Roadmap' - Architecture of a typical wavelet-based video CODEC.....	51

Figure 8.2 ME/MC in spatial domain	52
Figure 8.3 ME/MC in Wavelet Domain.....	53
Figure 8.4 a) Four level binary wavelet tree. b) The filter bank, used to achieve perfect reconstruction from an inverse tree.....	53
Figure 8.5 Comparison of some common wavelets [20].	54
Figure 8.6 Original signal and shifted version	54
Figure 8.7 Wavelet domain representation	54
Figure 8.8 Step response of level-4 Antonini wavelet (wavelet domain representation).....	55
Figure 8.9 MCTF incorporated in a three-dimensional wavelet transform	56
Figure 9.1 DWT from file to file [11]	60
Figure 9.2 Conceptual Classdiagram of the still-image application	62
Figure 9.3 Conceptual Classdiagram of the still-image application	63
Figure 9.4 Sequence diagram: Compressing an image and writing to file.....	64
Figure 9.5 Mallat decomposition – from JPEG2000	65
Figure 10.1 Coarse sketch of how to handle different type of input and output	68
Figure 10.2 User interaction for compressing, displaying, and storing a movie clip ..	68
Figure 10.3 Layout sketch for compression options	69
Figure 10.4 Coarse overview of the processes in the MDWT CODEC.....	71
Figure 10.5 Overview of the intra-frame encoder and decoder	72
Figure 10.6 Sequence diagram: Preparing the Processor for playback	73
Figure 12.1 MDI window with menu	76
Figure 12.2 Menu item: File->Open Image.....	77
Figure 12.3 JMF Registry Editor: Detect Capture Devices	78
Figure 12.4 Open file dialog.....	79
Figure 12.5 Video panel with presentation controls	79
Figure 12.6 Video encoding options dialog	80
Figure 13.1 Original (256x256)	82
Figure 13.2 SPIHT	82
Figure 13.3 EZW	82
Figure 13.4 JPEG2000.....	82
Figure 13.5 JPEG.....	83

List of Tables

Table 3.1 Block-matching search algorithms. Based on [6].....	21
Table 3.2 Video compression standards	24
Table 7.1 Coefficients ordered by magnitude.....	49
Table 10.1 Compression options.....	69
Table 10.2 Menu Items	70
Table 13.1 PSNR Performance at 0.5 bpp and 48:1 compression ratio.....	83

List of Equations

Equation 4.1 Calculate wavelet coefficients with the Continuous Wavelet Transform	28
Equation 4.2 Mother wavelet	28
Equation 4.3 Calculate coefficients using convolution.....	29

Part 1 Fundamentals – Background material

1 Introduction

Television and telephones have had a huge impact on the life of people in the western world after being introduced about a century ago. The development has been amazing; from black-white poor-resolution TV to high-quality interlaced colour TV, from old-fashion telephones to tiny wireless (mobile) phones. And now the world goes digital, and that is changing our life again.

We have already seen signs of 'everyday digitalization'; video are stored on DVD's, The 'Star Wars' film shot in 2000 was the first motion picture to be shot digitally in 2000, and digital cameras has become public property. TV will be broadcasted digitally within few years [1] and the digitalization of cinema is on its way [2]. Imagine the crystal-clear TV screens, the increased interaction possibilities, the freedom to choose, and the foundation for new mind-blowing inventions.

Mobile operators have had visions of consumers watching video and TV-clips on mobile phones for years. Japan and Korea have already taken the first steps, by introducing third-generation (3G) mobile systems back in 2002. All 3G handsets sold by Japan's largest mobile operator (NTT DoCoMo Inc. (DCM)) are video enabled. Handsets with two cameras are being sold, which in addition to being used as a video-phone can be utilized as a video-camera that allows hours of video recording if combined with memory cards. More than 200 000 was sold in a couple of weeks. Everyday we see the development and improvement of technology. 10 years from now the whole world surrounding the 3G mobile videophone will be totally different (by then, we will be calling it 4G, though).

However, the mobile operators dream can not yet be fulfilled. Considering the limited maximum available bandwidth of 3G mobile systems, and that an uncompressed "television-quality" digital video signal requires a transmission capacity of 216 Mb for one second of video, it is evident that something has to be done. Even with the high quality video coders of today, this can not be done in a high-quality fashion. Even though JVT (Joint Video Team) recently released a new video coding standard which perform superior compared to its predecessors, new visions, new creative inventions and consumer-created request for enhanced quality will continue the request for improved compression algorithms.

So, high compression ratio of video is crucial for today's limited bandwidth, storage and processor capacity, especially in handheld wireless communication devices such as mobile phones and PDA's. Moving Pictures Experts Group (MPEG) and International Telecommunication Union (ITU) have standardized the most widely used video CODECs, like MPEG-4 (MPEG) and H263 (ITU). Their standards*¹ are based on Discrete Cosine Transform (DCT). DCT is the most popular transform for video coding applications, but there are problems which reduces visual quality at high compression ratio when using DCT for image and video transform coding. Preliminary surveys and existing research indicates that wavelet, a relatively new mathematical field, is a powerful tool in the world of digital image processing. It is shown that wavelet can reduce problems like blocking artifacts caused by DCT in still image compression, and in fact, the most recent still image compression standard from Joint Pictures Expert Group (JPEG), JPEG2000 is based on Discrete Wavelet Transform (DWT).

¹ All recent published standards, except MPEG-4 which provides an alternative set of wavelet-based tools to code static texture.

1.1 Problem Definition and Motivation

The goal of this master thesis is to do a literature survey and if time permits to develop a software video codec prototype based on wavelet. The prototype should consider advantages and disadvantages of the previous study. This thesis should focus on temporal redundancy removal (motion estimation and compensation) and transform coding. These are the main stages of a video compression process that may benefit from using wavelets. The literature survey should consider if recent research has found ways to compress video signals using wavelets which results in 'pleasing' visual quality at high compression ratio, and whether this outcome can match or even outperform existing video CODECs. In addition the survey should aim to give an impression whether wavelet based video compression is an active field of research. If time permits, real-time processing and computational complexity are important issues that should be discussed in the literature survey.

1.2 Preliminary Survey of Previous Work

In order to gain competence in the field of video compression and wavelet transform, we have performed a preliminary literature study. The most important has resulted in chapter 3 Fundamentals of Video compression and chapter 4.

"In our search, we discovered the following:

Wavelets aren't nearly so promising for use in inter-frame CODECs, because wavelets actually make temporal compression more difficult than DCT. Wavelet compression is processor intensive, and so is considered better suited to still images than video." [5]

"Because of its good performance in compressing images, the DWT is used in the new JPEG-2000 still image compression standard and is incorporated as a still image compression tool in MPEG-4. However, wavelet techniques have not yet gained widespread support for motion video compression because there is not an easy way to extend wavelet compression in the temporal domain. Block-based transforms such as the DCT work well with block-based motion estimation and compensation, whereas efficient, computationally motion-compensation methods suitable for wavelet-based compression have not yet been demonstrated. Hence, the DCT is still the most popular transform for video coding applications." [6]

"At the time of writing, wavelet compression has made very little impact compared to DCT-based compression. There are numbers of reasons for this. Wavelets have been less successful than DCT-based system in achieving good efficiency at the near-transparent compression ratios. Also, once DCT was adopted by MPEG, most development effort went into producing integrated circuits for MPEG – that is, DCT. Until recently, little specialist silicon was available for wavelet compression. However, this is changing, and now that wavelet compression has been adopted in MPEG-4 (for static textures) and in JPEG2000, wavelet implementations are likely to become much more common." [7].

This triggered us to investigate what research work has been done in this field, and has resulted in part 2. In addition we wanted to investigate use of phase correlated ME/MC algorithms.

1.3 Prototype

We have implemented a video compression prototype founded on a DCT- and DWT-based still-image compression application developed by four engineer students in a previous project. We have incorporated the DWT-based compression algorithms in our prototype, by tearing application apart, and reconstructing the interesting parts inside an adapted Java Media Framework API (JMF API 2.1.1). We have designed a highly extendable prototype considering the advantages and disadvantages of the research study presented in Part 2. We carefully planned a stepwise implementation progress, to help us make sure making the prototype extendable. Having mainly focused on doing a thorough design work and a literature study, the entire plan has not been implemented. However, this was never the intention. We wanted to make a foundation that could be utilized in future studies, both for our one interest and others.

The current implementation contains a framework with support for opening and playing video sequences of those content types supported by the JMF API, including quicktime (*.mov) and msvideo (*.avi). Capturing video clips from a web camera is also supported, the captured sequence can either be rendered direct to screen, or stored to a 'quicktime' or 'avi' file. Our implementation does not currently support encoding the captured video directly with our encoder; it has to be intermediately stored to a file. We have implemented a so-called intra-frame CODEC (no attempt is done to remove temporal redundancy), the MDWT (Motion DWT) CODEC. It includes a MDWT encoder and decoder; this is where the DWT compression algorithms have been exploited. In addition, we have implemented a MDWT file format and a multiplexer/demultiplexer to handle writing and reading this format. Initially we implemented support for sound tracks in this multiplexer/demultiplexer, but this was removed in order to more correctly estimate the real compression performance of our CODEC.

Experimental testing has showed that our MDWT CODEC can yield a compression rate of 13:1 in a small resolution video clip, with good visual quality on single frames. The major problem of our CODEC is as initially concerned, the computationally complexity. Small-resolution video becomes very jerky, while larger-resolution video clips are considerably worse, maybe able to show one frames pr second. This is due to the combination of the computationally complexity of the implemented DWT and the CODEC, the 'slow nature' of the Java programming language, and most-likely and inefficient method used to draw and represent the frames.

2 Methodology

2.1 Literature References

Agder University College have access to the IEEE database, and we have used it comprehensively to find research papers.

We have tried to avoid using web sites as reference, in the cases we have used a web site as a reference, and this is sites which are commonly accepted and/or formal/official business sites.

2.2 Test Methods

There is no easy way to get an absolute objective measure of image or video quality. The best test method is still to look at the image or video sequence and determine by a subjective visual test whether the image looks good to you or not. It is easy to understand that this form of testing is highly subjective in that different people can have different opinion whether the image looks good or not. This has been a problematic challenge for researchers who of course want an absolute measure of their work, but no obvious solution has yet been presented.

There is one measure which has gained widespread popularity among image and video community, and that is Peak Signal to Noise Ratio (PSNR). This measures the difference between the original image and the compressed image. The problem with this quality measure is that it is not always given that a higher PSNR value is analogous with better visual quality.

2.3 Other issues

Unfortunately, we did not manage to complete all our intended work on time. The missing parts are mainly found in Part 4, we have not been able to document all the performed tests, comparisons and discussions. Another part that suffers from this is the references.

3 Fundamentals of Video Compression

The purpose of this chapter is to describe some basic concepts of image- and video compression to give an overview of the most common topics in modern video coding. Firstly we establish some concepts and terminology relating to digital video, then we take a look at the processes in a typical video CODEC architecture, before examining how to compress the different types of redundancies in a video sequence. The last section in this chapter gives a brief overview of video coding standards, and the standardization organisations these standards emerge from.

Note, in literature there are arguments about the semantics of compression, we have used 'compression' to cover all techniques that reduce data. We will introduce some basic concepts and some of the most popular algorithms used in modern video coders.

3.1 Introduction

An uncompressed 2-hour movie requires more than 194 GB of storage, which is equivalent to 42 DVDs or 304 CD-ROMs [2]. A "television-quality" digital video signal requires 216 Mb of storage or transmission capacity for one second of video [6]. Or consider another example, if you transmit a one minute uncompressed video clip (640 x 480, 30 frames/sec, 24 bpp) using a 28.8K modem, it would take you 5 days and 8 hours.

These examples illustrate the need for compression. The massive improvement of video compression techniques in early nineties lead to increased use of digital video, and introduction to several new areas. As it for example is introduced in communication devices like mobile phones, witch offers low bit rate and less computational power, the demand for even further improved compression techniques increases. Compression (coding) is the science of reducing the amount of data used to convey information [7]. It relies on the fact that information exhibits order and

patterning, and if this order and patterning can be extracted, the essence of the information can often be represented using less data than for the original. We can then reconstruct the original, or a close approximation of it. With good video compression algorithms, the 2-hour movie can be compressed and stored on 2-3 CD-ROMS (including audio), without noticeable difference in visual quality.

3.2 Digital Video

Digital video is probably the most important visual source nowadays. The numerous applications available have made it an integral part of the everyday life for many of us. It is implemented in many aspects of business, education and entertainment, from digital TV to web-based video news.



Figure 3.1 Digital video

A video image (frame) is a projection of a 3-D scene onto a 2-D plane, see Figure 3.2. A still-image can be seen a 'snapshot' of the 2-D representation at a particular instant in time, whereas a video sequence represents the scene over a period of time, containing a sequence of still-images (frames). The video sequence is represented as a signal in a discrete form. As illustrated in Figure 3.3, it is sampled both spatially (typically on a rectangular grid in the video image plane) and temporally (typically as a series of frames sampled at regular intervals in time).

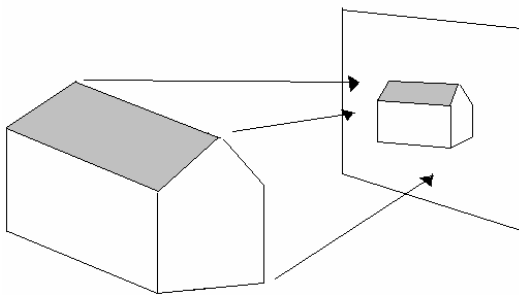


Figure 3.2 Projection of 3-D scene onto a video image

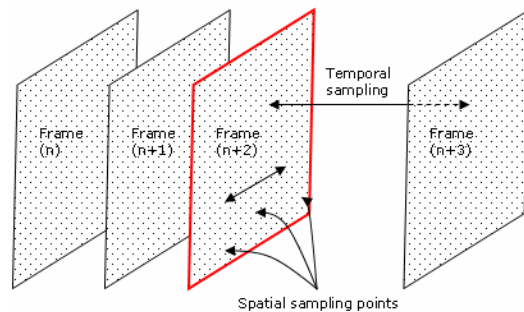


Figure 3.3 Spatial and temporal sampling

Each spatio-temporal sample, also known as a picture element or pixel (pel), is represented digitally as one or more numbers that describe the brightness (luminance) and colour of the sample. The visual quality of the image is influenced by the number of sampling points. More sampling points (a higher sampling resolution) give a 'finer' representation of the image, but requires higher storage capacity. A higher temporal sampling rate (frame rate) gives a 'smoother' appearance to motion in the video scene but requires more samples to be captured

and stored. The way the sampling is performed is crucial for correct representation and reconstruction of the video signal. A lot of research and theories is presented in this field, this is beyond the scope of this thesis, some fundamental sampling theory is presented in [7, 8].

A video usually comes in colours; while a monochrome ('grey scale') video image may be represented using just one number per pixel, indicating the brightness or luminance of each sample position, representing colour requires multiple numbers per sample. There are several alternative systems for representing colour, each of which is known as a colour space. The most common colour spaces for digital image and video representation are RGB (red/green/blue) and YCrCb (luminance/red chrominance/blue chrominance).

3.3 What is a Video CODEC?

"A program or a device that compresses a signal is an encoder and a device or program that decompresses a signal is a decoder. An enCOder/DECOder pair is a CODEC" [6, p. 28].



Figure 3.4 Architecture of a typical video encoder

The information in a digital video is highly correlated and redundant. As illustrated in Figure 3.4, the aim of a (lossy) video CODEC is to decrease this information by removing the information which cannot be perceived by our visual system. There are several solutions to reduce redundancies and irrelevant information, and in the following sections we will depict some of them. But first we want to establish some common terminology:

Lossless vs. Lossy compression

In lossless compression schemes, the reconstructed signal, after compression, is numerically identical to the original image. It is typical to remove statistical redundancy; an example of a typical lossless compression algorithm is implemented by the widely used winzip application. Lossless compression will rarely provide sufficient compression ratios for digital video, when used in systems or applications with limited bandwidth. Compression schemes that remove data from the original signal are generally referred to as lossy compression schemes. In this thesis we will mainly concentrate on lossy compression techniques.

Intra-frame vs. Inter-frame

In intra-frame compression each frame in a moving image sequence is processed without any consideration for previous or future frames, as opposed to inter-frame compression where sequences of frames are processed, typically encoding only the differences between frames. In an intra-frame video CODEC, like e.g. MotionJPEG, spatial redundancies are usually removed, by using an image compression algorithm on every frame in the video sequence. In an inter-frame video CODEC, like e.g. H.263+ or MPEG-4, it is typical to remove both spatial and temporal redundancies, by combining a spatial (image) compression technique with a motion estimation and compensation (ME/MC) algorithm to remove the temporal redundancies.

3.4 Removal of Spectral Redundancy

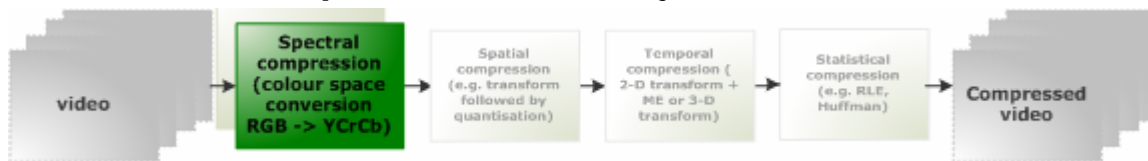


Figure 3.5 'Roadmap' - A typical video coder

The input for video compression schemes is usually a full colour video signal in RGB colour space. The pixels in the RGB space are represented by three numbers, indicating the relative proportions of red, green and blue (the three primary colours of light). RGB systems usually represent each of these three components with the same precision which means that all three colours are treated as equally important. The luminance is also present in all of these three colour components, and is therefore redundant. YCrCb on the other hand, is a more efficient colour space, because it has the property of separate luminance (Y) and colour (CrCb) parts. For every pixel in a colour image there is one luminance (Y), and two chrominance (Cr and Cb) components. Because the Human Visual System (HVS) is more sensitive to luminance than colour, some of the colour information can be removed without affecting visual quality. Since Cr and Cb are redundant, these signals can be downsampled. This implies that we can eliminate spectral redundancy in an image or video by converting the RGB colour space into YCrCb space. In 4:2:2 YCrCb systems, downsampling are applied only in horizontal direction, giving a 3:2 compression ratio. Whereas in 4:1:1 YCrCb conversion, Cr and Cb signals are downsampled in both horizontal and vertical directions and a 2:1 compression ratio is achieved.

3.5 Removal of Spatial Redundancy



Figure 3.6 'Roadmap' - A typical video coder

Besides spectral redundancy, image- and video signals also have spatial information we can not perceive. It is very hard to determine which pixels to remove in order to do a compression on the signal without affecting visual quality too much. If we think of the pixels as energy, this energy tends to be evenly distributed all over the image, and it is therefore hard to compact the information without doing some manipulation first. A solution is to transform the image into a different representation.

3.5.1 Transform Coding

A good transformation should be able to represent the signal with a few significant values, or compact the energy of the image. The most widely used transform method in modern image- and video coding algorithm is the Discrete Cosine Transform (DCT). Until recently, DCT was thought of as the best transform method in image coders. But when JPEG introduced their newest still-image compression standard, JPEG2000, it was based on another transform which outperformed the DCT [9], the Discrete Wavelet Transform (DWT). The DWT is presented in chapter 4 Fundamentals of Wavelets.

Both DCT and DWT transform the image to get decorrelated values. The aim is to localize signal energy at different frequencies. This is done by separating the lower spatial frequencies in the spatial domain from the higher ones, and representing them as coefficients. The lower frequencies are more visible to the human eye, and are the largest “building blocks” of an image, while higher frequencies add more details. Given the fact that the HVS is less sensitive to higher spatial frequencies, these high spatial frequency coefficients can be removed without considerably affecting image quality.

3.5.2 The Discrete Cosine Transform (DCT)

The Discrete Cosine Transform (DCT) is most widely used to perform transform coding in video compression algorithms. Even though other transforms as the Karhunen-Loeve Transform (KLT) has been known to outperform the DCT regarding compression performance, the simplicity and computational efficiency of the DCT has made it the first choice. The DCT has been around for well over a decade. The basic idea behind DCT was developed by Joseph Fourier a couple of centuries ago. (Some basics of Fourier are presented in section 4.3 `.) Fourier has to be applied to continuous, periodic signals while digital signals are discrete. The Discrete Fourier Transform (DFT) was developed to overcome this obstacle, and later a fast and more efficient implementation of this transform was developed, known as the Fast Fourier Transform (FFT). In

and Figure 3.8 we can see that cosine is symmetric and sine is asymmetric about their origin. This means that a signal containing both sine and cosine signals is asymmetric, but it can be made symmetric by using mirroring. Since the signal is symmetric, it can be represented only by cosines. So, as opposed to Fourier which needs both sines and cosines to represent a given signal, the DCT is only based on cosines which makes it less complex and very efficient.

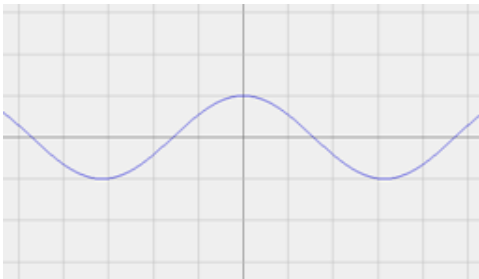


Figure 3.7 Cosine Wave

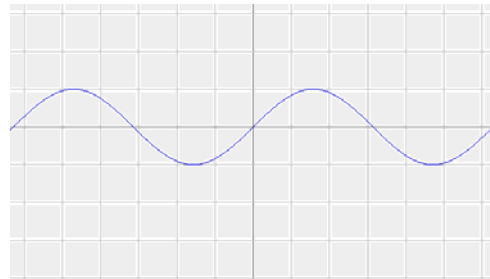


Figure 3.8 Sine wave

When an image or a frame is transformed by DCT, it is first divided into blocks, typically of size 8 x 8 pixels. These pixels are transformed separately, i.e. without any influence from the other surrounding blocks. The top left coefficient in each block is called the DC coefficient, and is the average value of the block. Since this is the most important coefficient in the block and has the lowest frequency, it should be coded without any loss. The rightmost coefficients in the block are the ones with highest horizontal frequency, while the coefficients at the bottom have the highest vertical frequency. This implies that the coefficient in the bottom right corner has the highest frequencies of all the coefficients.

3.5.3 Quantization

By performing a transform the image is not compressed in any way, actually it makes the file size larger [6, p. 43]. The image is only represented in a different domain where the image data is separated into components of varying importance. It

is during the quantisation process that the image is compressed, and the first step of this process is to divide each of the coefficients by an integer. The smallest or most insignificant coefficients, usually the high frequency values, are mapped to zero. By doing this, we remove information from the image, so the compression is irreversible (lossy).

Soft quantization only maps the smallest coefficients to zero, while coarse quantization maps larger, and therefore more coefficients to zero. This implies that coarse quantization usually gives higher compression at the expense of loss in image quality. In the coefficient matrix, the lower spatial frequencies are represented in the top left, and the higher frequencies in the low right. The quantized coefficients can be represented in different patterns. The aim of these patterns is to get long consecutive streams with zeroes (or other values), so the benefit of a statistical compression algorithm, known as Run-Length Encoding (RLE), can be as effective as possible. Since the output coefficients vary depending of type of source and the transform used, the optimal pattern also varies. The zigzag-pattern illustrated in Figure 3.9 is the most used for the DCT.

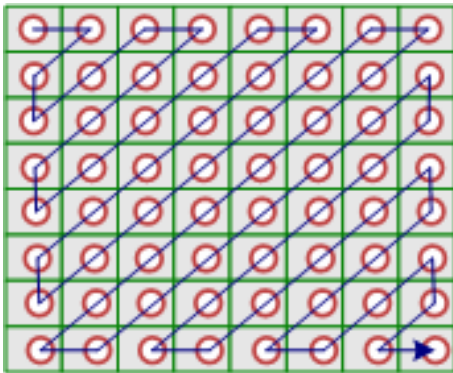


Figure 3.9 Zig-zag scan of coefficients

3.6 Removal of Temporal Redundancy

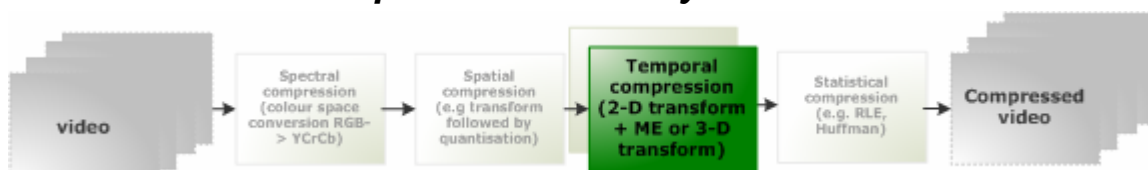


Figure 3.10 'Roadmap' - A typical video coder

Spatial compression is important, but spatial compression alone does not get us even close to common compression goals. A video signal is typically sampled with a rate of 15-30 frames second. This makes the subsequent frames in a video sequence usually appear almost identical. In most cases, only small portions of the scene change from frame to frame. For example, the four first frames of the sequence "Foreman" (Figure 3.11) contains minor movement, but even in the high-motion sequence "Football" (Figure 3.12), where the players are running and diving, the changes from frame to frame is minor.



Figure 3.11 First four frames of the "Foreman" sequence.

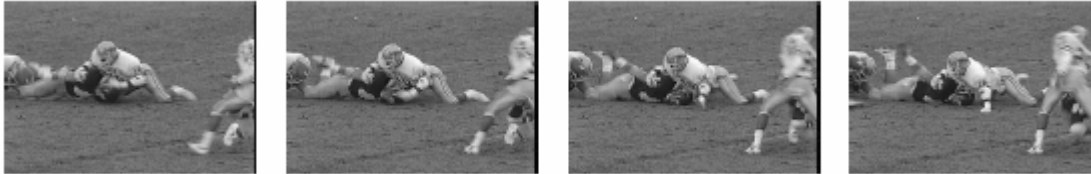


Figure 3.12 First four frames of the "Football" sequence.

Especially, the background information will be similar, if the camera movements and illumination changes are insignificant. This causes a temporal redundancy between frames. In order to remove this redundancy, modern video coders employ motion estimation and motion compensation (ME/MC). Another approach is to utilize a three-dimensional transform which is applied in both time and spatial domain, instead of a two-dimensional transform combined with a ME/MC technique [6].

3.6.1 Three-dimensional transform coding

Since a video sequence consists of a sequence of two-dimensional images, existing in three dimensions, our intuitive idea would be to extend the tools we been using from two-dimensions to three. Both the DCT and the DWT can be defined for three dimensions, just as they can for one or two. In fact, this approach works for video sequences that are soft and have only very slow motion [7], like for example in a video conferencing sequence. Unfortunately, this is not how motion in a video scene usually appears. Fast motion creates temporal aliasing and, if the spatial edge is sharp, temporal aliasing is created even by slow motion. The fundamental problem is, of course, that the temporal sampling rate of today's video systems is too slow. If an object moves significantly between two samples, we have no information about what happened between the two samples. In fact, we do not even know for sure that it is the same object... (Nyquist's sampling theorem).

Hence, the favoured approach in modern video coders for removing temporal redundancy is using a ME/MC algorithm combined with a 2-D transform.

3.6.2 Motion Estimation (ME) and Compensation (MC)

"The motion estimation and compensation functions have many implications for CODEC performance. Key performance issues include:

- Coding performance (how efficient is the algorithm at minimising the residual frame?)
- Complexity (does the algorithm make effective use of computation resources, how easy is it to implement in software or hardware?)
- Storage and/or delay (does the algorithm introduce extra delay and/or require storage of multiple frames?)
- 'Side' information (how much extra information, e.g. motion vectors, needs to be transmitted to the decoder?)

- Error resilience (how does the decoder perform when errors occur during transmission?)

These issues are interrelated and potentially contradictory (e.g. better coding performance may lead to increased complexity and delay and poor error resilience).“ [6, p. 93]

The basic idea of ME/MC is simple: Find out if and where parts of the previous frame have moved to in the new frame. If the CODEC knows this, it can use the previous information to predict the new frame, and so will need less information. In practice, doing this right is very complicated, and motion search algorithms are of a major focus of ongoing compression research.

We can split ME/MC into two main functions:

1. Motion estimation: create a prediction of the current frame based on one or more frames
2. Motion compensation: subtract the prediction from the current frame to produce a 'residual frame'.

The motion estimation is the key to this approach, the more accurate the estimation is the less data will the residual frame contain, hence the video can be compressed to a smaller size, and less data has to be transmitted. In addition to transmitting the residual frame, we also have to transmit a set of motion vectors which describe the scene motion as observed at the encoder (for example the location of the 'best' match). But this information is usually considerably less than an intra-frame coded frame. In order to decode the frame, the compensation process is 'reversed' and the prediction is added to the decoded residual frame (reconstruction).

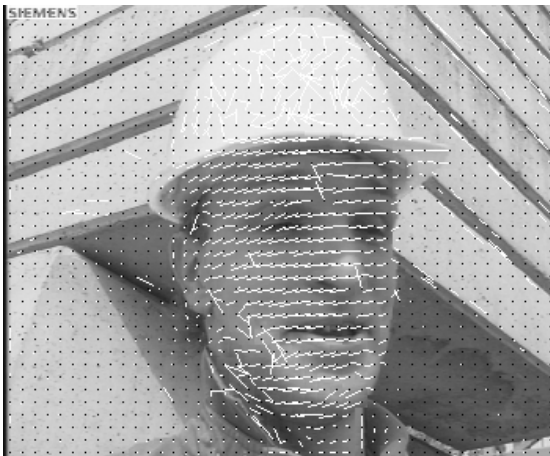


Figure 3.13 Predicted motion vectors for frame 2 of the 'Foreman' sequence

A very simple method of inter-frame encoding is to subtract the value of the previous frame (Figure 3.14 a). For video where the camera is not moving at all, this can yield substantial savings. However, only a little camera motion can throw everything off. Changes in one frame to the next are usually due to movement in the video scene, and if we can estimate this movement accurately, and then compensate for it, a significantly better prediction can be achieved (Figure 3.14 b).

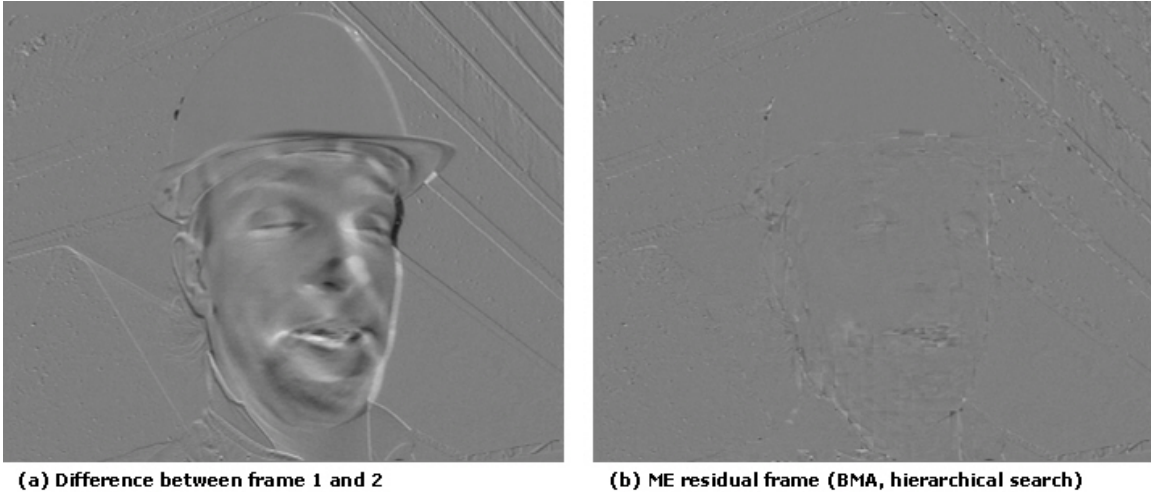


Figure 3.14 Residual frame examples from VCDemo

A number of ME algorithms have been developed in order to provide efficient prediction of scene motion between frames. ME schemes can generally be categorized as feature matching or region matching. Feature-matching ME is based on tracking specific image features (e.g., edges); however, the region-tracking methods are used almost exclusively in modern coders. We will briefly explain the main idea behind the most commonly ME/MC algorithm, block matching algorithm, and a ME/MC algorithm that is known to determine motion more accurately. Other commonly used ME/MC algorithms is optical flow methods (OFE) and pel-recursive methods.

3.6.3 Block matching algorithm (BMA)

The most widely used region-tracking technique is block matching, illustrated in Figure 3.15, in which the current image is divided into small blocks. The previous frame, called the reference frame, is searched for the best matching block for a given block in the current frame, and the resulting motion vector, indicates the position of the best-matching block.

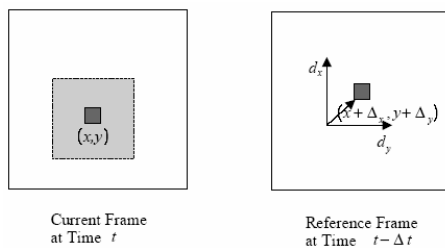


Figure 3.15 The block matching algorithm

The difficult bit is to find the best matching block. Before starting to search we have to establish some concepts:

Matching criteria

The ME scheme have to know what criteria to search for. Most ME schemes look for a minimum mean square error (MSE) between blocks. Other commonly used matching criterion for is the slightly less complex mean absolute error (MAE) or for even more simplified comparison, the sum of absolute errors (SAE) [6].

Optimal block size

It is also important to consider the effect of the block size that we attempt to match. For example, if the moving object is large and the block is large, it is harder to find a predictor block that matches reasonably well; we may find a wrong motion vector or maybe no acceptable match at all. If we on the other hand use a very small block, many more motion vectors must be coded. It is also very likely that many matches will be found; most of these will not be related to real motion within the image, only other areas that happen to be a similar patch of image. It has been shown that 16x16 is the first easy-to-use size for the DCT [7].

Search window

Theoretical it is necessary to compare the current block with every possible region of the reference frame, but this is usually very computationally intensive. In practice, a good match can usually be found in the immediate neighbourhood. Hence the search for a matching region is usually limited to a 'search window'.

Table 3.1 Block-matching search algorithms. Based on [6]

Search algorithm	Manner of operation	Performance
Full-search block matching	Every block within the search window is tested against the block it is desired to match.	Computationally intensive, particularly for large search windows, but usually finds the best match.
Fast-search	Usually search a few points within the search window to find minimum SAE, thereafter search within a local area for the best mach. Many alternative 'fast search' algorithms have been developed <ul style="list-style-type: none"> - n-step search - Logarithmic search - Cross search - One-at-a-Time search - Nearest Neighbours search 	Aims to reduce number of comparison operations, but gives usually poorer compression performance than full-search. Nearest-neighbours search is reported to perform almost as well as full search, with a very much reduced complexity.
Hierarchical search	Search the image in a coarse to fine fashion. Usually starts with a coarsely subsampled version of the image, followed by successively higher-resolution versions until the full image resolution is reached. Can perform a fast- or full-search at each level. Complexity by performing full-search at highest level is relatively low because groups of pixels are compared, not every pixel.	Can give a good compromise between performance and complexity and is well suited to hardware implementations

3.6.4 Phase Correlation

Motion estimation does not necessarily identify 'true' motion; it is rather an attempt to find a matching region in the reference frame that minimises the energy of the difference block. Unlike the BMA method, which searches the blocks from luminance matches, the phase correlation method measures the movement between the two fields directly from their phases. Phase Correlation is a means of determining very accurately the components of motion in an image sequence, by studying the amplitude and phase of each frequency in the image. This is done by transforming adjacent blocks into the frequency domain and then subtracts the transforms. The

result mainly consists of frequencies were there were a significant phase shift between fields (assuming similar frequency content). After normalizing these frequencies, they are passed to an inverse transform, which generates a new value to each pixel. These values represents a spatial surface (called a correlation surface), which displays amplitude peaks corresponding to the movement of each object. The position of the peak gives an accurate measure of the direction and speed of motion. Phase Correlation can be employed on the whole frame, but because of the computational complexity, a the frame is usually divided into blocks, as similar to the BMA. [7]

Unfortunately, when using the Fourier transform or the discrete cosine transform (DCT) in phase correlation, all precision in the spatial domain is lost. Phase correlation will very accurately find the amplitude and direction of each motion component in the image, but we do not longer no where in the image the motion occurs. There are ways to calculate this by an offsetting and subtracting technique, but it makes phase correlation a computational complex way to estimate motion. Recognising that wavelets as apposed to Fourier, gives both frequency and location information, we consider this a very interesting approach for wavelet-based ME.

3.6.5 Enhancements to the motion model

Sub-pixel motion estimation

Sub-pixel ME gives better block matching performance at the expense of increased computationally complexity. Most recent ME algorithms offers sub-pixel accuracy, this is done by interpolating the original pixels to form a higher-resolution interpolated region. Figure 3.16 shows an example of half-pixel. Black pixels are original integer pixel positions. Light grey pixels are formed by linear interpolation between pairs of integer pixels. Dark grey pixels are interpolated between four integer pixels.

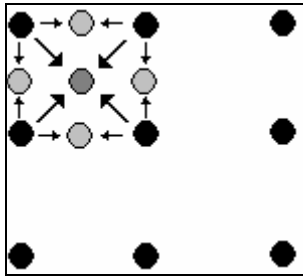


Figure 3.16 Half-pixel interpolation

Choice of reference frames

ME algorithms can predict the current frame by using 'older' encoded frames (forward prediction), 'future' frames (backward prediction), or by choosing the frame that gives best measure (e.g. SAE) of 'older' and 'future' frames (bidirectional frames). Some algorithms also have an option to use multiple reference frames to perform the prediction, e.g. MPEG-1 and MPEG-2. The choice of reference frame has influence on the compression performance of the ME algorithm.

There are a number of other advanced ways in which the motion model may be enhanced. Several of these methods are known as rather computational complex, but as technology performance is constantly improved, several of these techniques are focus areas in recent research work:

Overlapped Block Motion Compensation (OBMC)

OBMC is among others used in the H.263 standard. OBMC is a way to reduce blocking artifacts introduced by for example the BMA method.

Vectors that can point outside the reference picture

Variable block sizes

Complex Motion Models

Picture warping

Mesh-based motion compensation

Object-based coding>

3.7 Removal of Statistical Redundancy

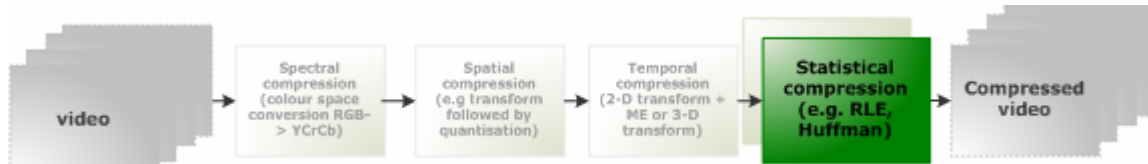


Figure 3.17 'Roadmap' - A typical video coder

Although the above redundancies are exploited, there is still redundancy in the compressed signal. This is due to the statistical properties of the video signals. In order to reduce these redundancies, some statistical lossless coding schemes are used. Run-length coding, Huffman coding and Arithmetic coding are examples of lossless coding techniques used to eliminate these statistical redundancies [12, 13].

3.7.1 Entropy Encoding

Entropy is a measure of disorder, or unpredictability, and the thought is that if a message is totally predictable, no information needs to be transmitted [7]. The concept of entropy encoding is to represent frequently occurring values using short symbols (hence using a small number of bits), and representing infrequently occurring values with longer symbols. The Morse alphabet is probably the most well known entropy coder, where the most frequently used letters in the English alphabet is assigned the shortest Morse codes, and the least frequent letters are assigned the longest codes. A variety of different entropy coders exists, the most popular used in video coding standards are Huffman coding and arithmetic coding (e.g. JPEG2000). We will only give a brief description of the popular Huffman coding scheme in this section, but arithmetic coding can more closely approach the theoretical maximum compression [13]. Entropy encoders are usually applied as the last step in the compression process.

In a typical transform-based video CODEC, the data to be entropy encoded falls into three main categories: transform coefficients, motion vectors and 'side' information (headers, synchronisation markers, etc). Frames, residual frames (after ME/MC) and motion vectors usually have different data structures, and can therefore benefit from being coded with differential quantization tables, zigzag-patterns and entropy coding tables (e.g. pre-generated Huffman tables).

3.7.2 Run-Length Encoding (RLE)

Data streams often contain long sequences with adjacent values. If such a stream contains, let's say one thousand consecutive zeros, this large section of similar bits could be represented in a much smaller binary sequence like '1000'. This is usually listed as pairs, for example like (0, 1000). Transform-based video CODECs have

usually high benefit of RLE, because the quantization process mostly produces zeros in the higher frequency coefficients, and when coding them with for example the zigzag-pattern, we get lots of consecutive zeros. RLE can then compress the stream without loss of information.

3.7.3 Huffman Coding

Huffman coding relies heavily upon accurate knowledge of signal statistics. The symbol with the highest probability is assigned the shortest code and the symbol with the lowest probability is represented with the longest code. To be able to decode this Huffman coded data file, the decoder must gain knowledge about which symbols are represented with which codes. This information is the code tree, or look-up table and it needs to be transmitted to the decoder prior to sending the encoded data. Huffman coding is very popular because of its effectivity and simplicity.

3.8 Video Coding Standards

ITU-T VCEG (International Telecommunications Union Video Coding Experts Group) and ISO/IEC MPEG (Moving Picture Experts Group) have been the leading standardization groups on video CODECs for well over a decade. Their main goal is to make good standards, which again helps to build a bridge between hardware and software manufacturers to allow compatibility between different systems. This enables a video format to be played in different types of video applications.

ITU's first standard, H.261 was developed to give good visual quality at low bit rates. MPEG-1 on the other hand was designed to give high quality at high bit rates. Table 3.2 shows the most important standards released by these groups.

Table 3.2 Video compression standards

Standard (ITU)	Bit rates	Usage	Standard (MPEG)	Bit rates	Usage
H.261 (1990)	64 Kbps	Video telephony over ISDN	MPEG-1 (1993)	1.4 Mbps	Audio video compression for CD-storage
H.263 (1995)	20-30 Kbps to several Mbps	Video telephony over packet and circuit switched networks	MPEG-2 (1995)	Typically more than 3 - 5 Mbps	Compression for storage and broadcast applications
H.263+ (1998) H.263++ (2001)	20-30 Kbps to several Mbps	Extensions to H.263, with improved compression performance	MPEG-4 (1998)	20-30 Kbps to high bit rates	Transport for multimedia terminals
Joint Video Team (JVT) (Merging of ITU and MPEG - December 2001)					
Standard (JVT)	Bit rates	Usage			
H264/MPEG4-AVC (2001)	From < 20 kbps to high bit rates	Video communication over low to high bit rates.			

Following the development, it is clear that the latest standards emerged from ITU and MPEG cover a broader range of video qualities and operating at a wider bit range. Both H.263 and MPEG-4 cover transmission speeds from the very low 20 Kbps to very high bit rates well above 1 Mbps.

In December 2001, the ITU-T VCEG and ISO/IEC MPEG formed the Joint Video Team (JVT) to establish a joint standard, H.264/MPEG4-AVC (similar to H.262/MPEG-2 video). This new standard is based on hybrid video coding and similar in spirit to earlier standards, but offers new key features as enhanced motion compensation, small blocks for transform coding, improved deblocking filter and enhanced entropy coding. Early indications give reasons to believe that this standard is superior to its predecessors, bit rate savings around 50 % against any other standard for the same perceptual quality has been reported. [6]

Most of the H.26x and MPEG standards are based on a hybrid-coding architecture, which features ME/MC followed by a DCT. Even if MPEG-4 has implemented a wavelet-based still image “texture” coder, we have yet to see a standardized video CODEC based on the DWT.

4 Fundamentals of Wavelets

The purpose of this chapter is to provide an easy introduction to the discrete wavelet transform (DWT), and some of the essential properties that seems to make wavelets appropriate for image- and video compression.

We have chosen to present the DWT as filter theory, because this is the most common way to represent wavelet in the field of video compression (and other signal processing communities). We will thank [6,7,8,14,15] for some good wavelet introductions, and credit especially [6,7,8] for some ideas and figures provided in this wavelet introduction. The mathematical derivation of wavelets is well beyond the scope of this thesis, but if you want a more fundamental understanding of wavelets, we strongly recommend you to read some essential mathematics. Many literature sources cover the subject in detail, we recommend [8], [15], which provides some fundamental mathematics of wavelets, and in addition introduces some basic mathematical structures that underlay the wavelet transform. If you like further information it is provide in for example [16] and [17].

4.1 Introduction

Wavelets are a mathematical procedure used in numerous types of applications, in a large variety of areas. Wavelets arises from a constellation of related concepts developed over a period of nearly two centuries, repeatedly rediscovered by scientists who wanted to solve technical problems in their various disciplines. Signal processors were seeking a way to transmit clear messages over telephone wires. Oil prospectors wanted a better way to interpret seismic traces. Yet “wavelets” did not become a household word among scientists until the theory was liberated from the diverse applications in which it arose and was synthesized into a purely mathematical theory. This synthesis, in turn, opened scientists’ eyes to new applications. Today, for example, wavelets are not only the workhorse in computer imaging and animation; they also are used by the FBI to encode its data base of 30 million fingerprints. In the future, scientists may put wavelet analysis to work diagnosing breast cancer; looking for heart abnormalities, or predicting the weather. [18]

4.2 Wavelets from a Historical Perspective

“Wavelets have had an unusual scientific history, marked by many independent discoveries and rediscoveries. The most rapid progress has come since the early 1980s, when a coherent mathematical theory of wavelets finally emerged.” [18]

4 Fundamentals of Wavelets

In an article [18] written by science writer Dana Mackenzie, with the assistance of several leading wavelets experts, including Drs. Ingrid Daubechies, Stéphane Mallat and Yves Meyer, we found a concise wavelet history timeline representation, which we really enjoyed. This is presented in Figure 4.1 and Figure 4.2. If you prefer a more detailed history, we can especially recommend [17, chapter 2], which provides a really thorough history guide and examination of the discovered wavelets.

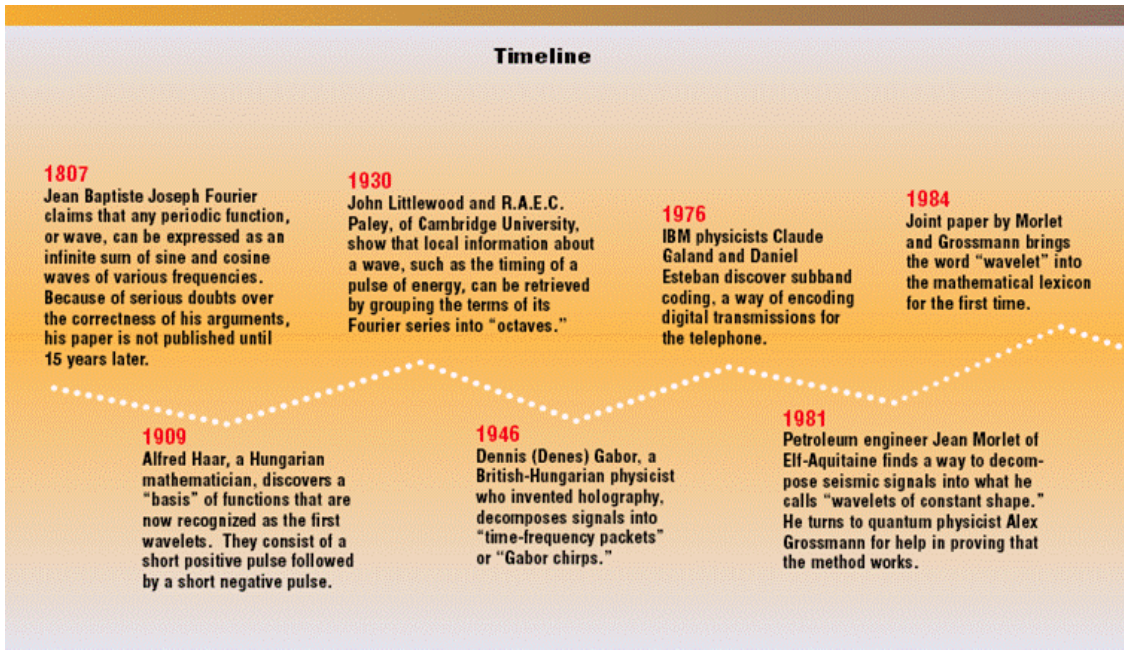


Figure 4.1 Wavelet history: Timeline 1807 – 1984 [18]

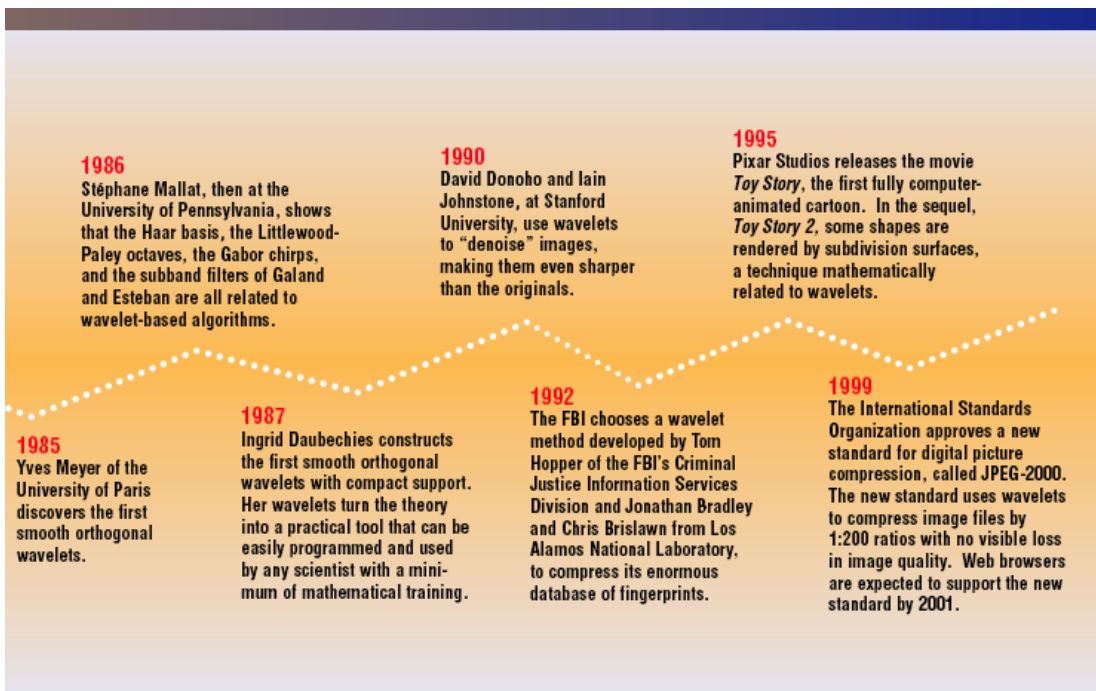


Figure 4.2 Wavelet history: Timeline 1985 – 1999 [18]

4.3 ‘Fourier vs. Wavelet

The most successful compression systems are based on transforms that move between the time or space domain and the frequency or spatial frequency domain. The fundamental building block for all such transform is the Fourier theorem, which states that any periodic function may be represented by an infinite sum of cosine and sine waves with different frequencies. This was revealed in 1807 (published 16 years later) by Joseph Fourier, who without knowing it discovered a new functional universe. [15]

Even though the Fourier transformation probably is the most commonly used, it has some severe limitations which makes it inadequate for some purposes. When transforming a signal with the Fourier transformation, the time information will become unavailable. Because the Fourier transformation is defined as an integral from -8 to 8 , a frequency component will affect the transformation equally regardless of where it occurs in the signal being transformed. Since most signals in practice are non-stationary, it is often more interesting to find where a frequency occurs, than finding the value of it. Several solutions have been developed to overcome this problem, the most known is maybe the modified version of the Fourier transformation, known as the Short Term Fourier transformation (STFT). The basic idea of the STFT is that it assumes that for some interval, the frequency of the signal does not change. It then applies the ordinary Fourier transformation to that interval. This interval is shifted to the right and the same procedure is repeated. In order to find the accurate location of a frequency, the interval has to be quite small, but the smaller it gets, the less accurate can we distinguish the different frequencies. This is a result of Heisenberg’s uncertainty principle, and is impossible to solve. What we can do is to improve this technique by varying the interval. This is also basically the same way as the wavelet transformation works.

and Figure 4.4 illustrates that wavelets basis-functions possess the ability of localizing frequencies, as opposed to the Fourier transform.

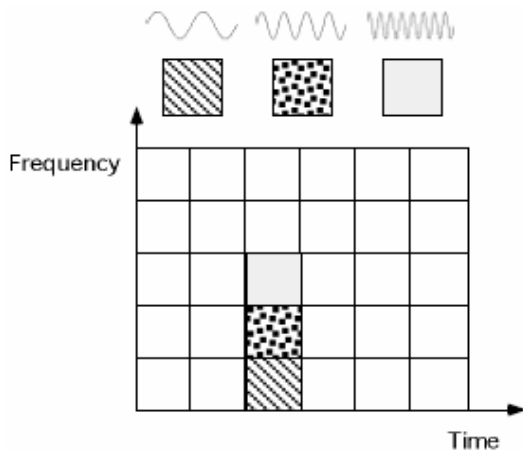


Figure 4.3 Fourier basis functions

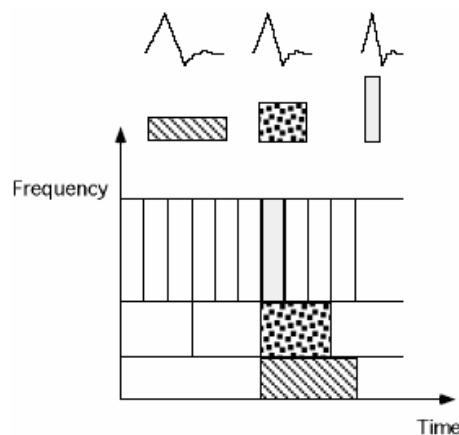


Figure 4.4 Wavelet basis functions

The other fundamental difference between wavelets and conventional transform techniques such as the Fourier transform and the DCT is that they search for similarities between a fixed basis function (sine or cosine) and the original signal,

while the wavelet transform have a huge number of basis functions that can be used, as long as they satisfy certain mathematical criteria.

4.4 Wavelets Concept

There are two ways to determine the wavelet transform. Most transforms have only one way, a formula. For example, with the following formula, we can calculate wavelet coefficients, $W(a,b)$, which gives us information about details from a signal [15]. The signal is represented by the function $f(x)$ in Equation 4.1 and Equation 4.2.

$$W(a,b) = (W_\psi f)(a,b) = \left\langle f \mid \psi^*_{a,b} \right\rangle = \int_{-\infty}^{\infty} f(x)\psi^*_{a,b}(x)dx$$

Equation 4.1 Calculate wavelet coefficients with the Continuous Wavelet Transform

The basic function ψ is usually called the mother wavelet:

$$\psi_{a,b}(x) = \frac{1}{\sqrt{a}}\psi\left(\frac{x-b}{a}\right)$$

Equation 4.2 Mother wavelet

In this chapter we will not concentrate on this formula, we will rather try to understand wavelet as the concept of filters, which is the way the wavelet transform is calculated in practice [8].

A wavelet can be used to 'pick out' details from a signal by multiplying it with the signal. The product would contain asymmetric information where the original signal had frequency content similar to that of the wavelet. Just as with the Fourier transform, integrating the result we get would give us a nonzero coefficient. (These are the same coefficients we find with Equation 4.1 for continuous wavelet transforms.) It is obvious that this result is depended on where on the signal the wavelet is placed. To get information on the whole signal, we have to "walk" the wavelet across the signal being transformed, this process is known as convolution [7].

In compression we are not dealing with analogue signals, but with a sequence of digital samples. The wavelet shown in would look more like Figure 4.6, where it is represented as a sequence of 13 sample values, six on each side of the origin.

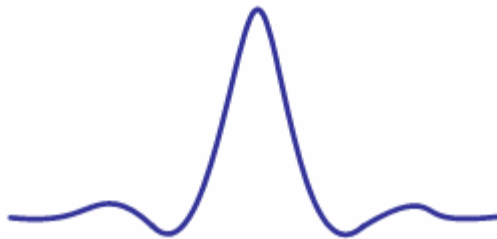


Figure 4.5 Wavelet can be viewed as a burst of energy with dominant frequency

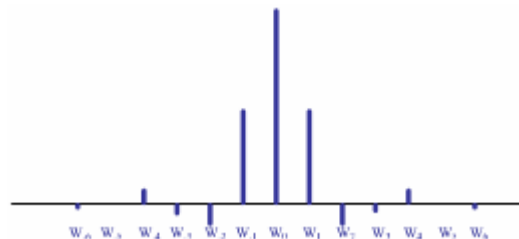


Figure 4.6 The discrete version of figure 1.3

In Figure 4.7 the wavelet is moved along, one sample at the time, and the convolution product is evaluated for each position. For simplicity, the wavelet is only represented by the five samples, W_{-2} to W_2 .

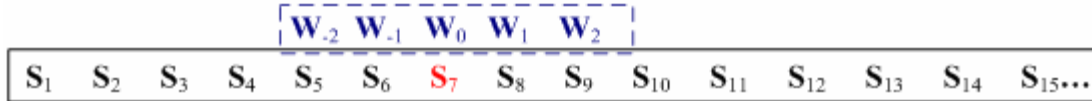


Figure 4.7 (a) Convolving a five-sample wavelet, W , with the samples of a signal, S . The wavelet is operating on S_7 . [7, slightly modified]

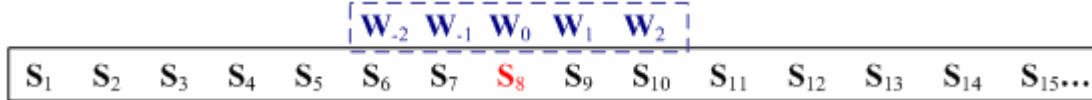


Figure 4.6 (b) The wavelet is moved along to next signal sample, S_8 . [7, slightly modified]

In Figure 4.7 (a) the wavelet is operating on sample S_7 , the output of the convolution is a new value S_7' , calculated in Equation 4.3

$$S_7' = (W_{-2} \cdot S_5) + (W_{-1} \cdot S_6) + (W_0 \cdot S_7) + (W_1 \cdot S_8) + (W_2 \cdot S_9)$$

Equation 4.3 Calculate coefficients using convolution

As shown in Figure 4.8 we have to create additional samples at the edges to convolve the wavelet over the whole sample set. In the example, only two additional samples are needed at each end. The signal is typically extended by 'reflecting' the edge samples far enough to accommodate the wavelet [7]. At the left-hand side we create $S_{-1} = S_0$ and $S_{-2} = S_1$. The idea of moving a wavelet over the image and picking out details shows us how wavelets can give both frequency and location information.

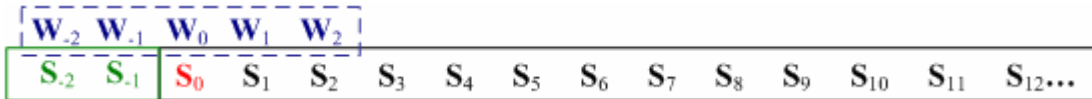


Figure 4.8 Additional samples must be created at the edges.

4.5 Wavelets as Filters

The convolution process just described can be represented as a digital filter, as shown in Figure 4.9.

In this classical representation of a filter, the signal samples, S , are input sequentially to the left-hand side and pass through four delays ('T') equal to the sample interval. When sample S_7 is at the centre of the filter, the output, ' S_7 ', is the same as shown in Equation 4.3. The impulse response of the filter is the wavelet shape shown in [7].

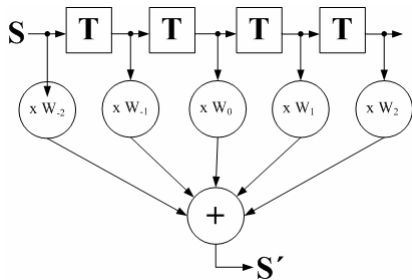


Figure 4.9 The five-sample wavelet shown as a five-tap filter [7].

4 Fundamentals of Wavelets

Viewing wavelet as filters, introduces another point – a wavelet is actually two complementary functions, or two complementary filters. Although there are a great number of complexities in the design and implementation of wavelet and scaling functions, the concept of fractional compression is quite simple. We filter the signal by convolving it with the wavelet, which extracts the high-frequency detail of the signal, behaving like a high-pass filter. The signal is also convolved with the complementary scaling function that removes the high frequencies [7]. One is known as the wavelet (or as the wavelet filter/band-pass filter), the other as the scaling function. and Figure 4.10 shows some examples of scaling functions and wavelets from the biorthogonal Deslauriers-Dubuc family. From left to right: (2,2), (4,2), (6,2) and (2,4). The first number is the number of vanishing moments of the analyzing wavelet (the wavelet that decomposes a signal) and the second number is the number of vanishing moments of the synthesizing wavelet (the wavelet that reconstructs the signal). Note that with increasing number of vanishing moments the wavelet becomes smoother or more regular. The same is true for the scaling function. Note also that the shape of these wavelets is not the rule. Although there are many wavelets that look like this, there are also many wavelets that look completely different. [19]

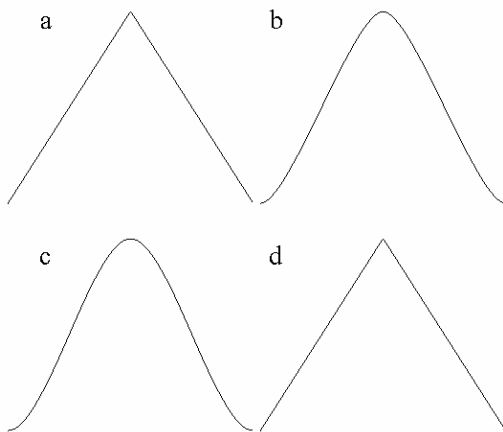


Figure 4.10 Some scaling functions from the biorthogonal Deslauriers-Dubuc family. a (2,2), b (4,2), c (6,2) and d (2,4). [19]

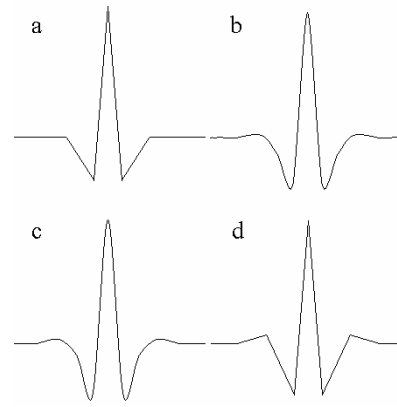


Figure 4.11 Some wavelets from the biorthogonal Deslauriers-Dubuc family. a (2,2), b (4,2), c (6,2) and d (2,4). [19]

The signal is split into two parts, one resulting sub-band which contains high frequency information and one which contains low frequency information. Figure 4.12 shows this decomposition process. So, say the signal represents an image, then as the high frequencies represents the fine details in an image, we now have a set of wavelet coefficients representing the fine detail of the image, H , and an image from which the fine detail has been removed, L .

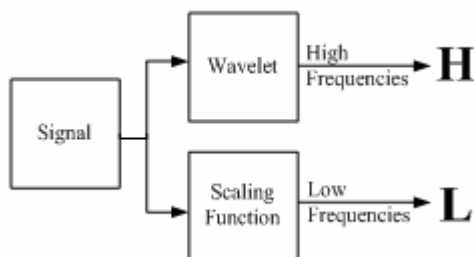


Figure 4.12 The first stage of the wavelet transform

Because of the bandwidth restrictions, the sampling density may be halved [7]. Figure 4.13 shows the same first stage of the wavelet transform as Figure 4.12, but here the downsampling operation is visible. It is represented by the symbol ($\downarrow 2$). This means that we downsample by two, we throw away every other sample². We keep the 1st, 3rd, 5th... samples and throw away the 2nd, 4th, 6th... samples, say. [8] Therefore if our signal includes 1024 samples, then the sub-band H contains 512 samples, as do L . Because of this downsampling operation, the DWT is called a (maximally) decimated or complete transform.

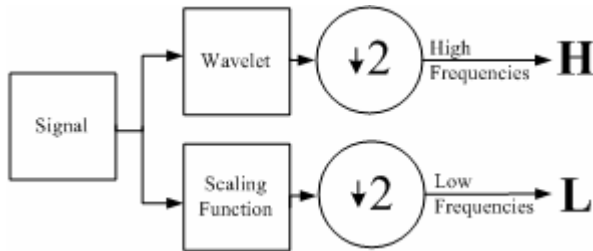


Figure 4.13 The first level of the DWT, including downsampling by 2

The decomposition process may be repeated for the low frequency band, L . The high-frequency part, H , represent half of the transform, it contains the smallest details we are interested in and we could stop here. However, the low-frequency band, L , still contains some details and therefore we can split it again by repeating the filter-downsample operation. Figure 4.14 shows what this means. In this figure we have labelled the first-stage sub-band H_9 and L_9 . This is simply because in our example they contain 512 samples and $2^9 = 512$. Similarly, the second filter-downsample sub-bands are labelled H_8 and L_8 because it contains $2^8 = 256$ samples. The 256 H_8 samples represent an additional one-fourth of the transform. Repeating the filter-downsample operation once again on the sub-band L_8 , generates H_7 and L_7 , each containing $2^7 = 128$ terms.

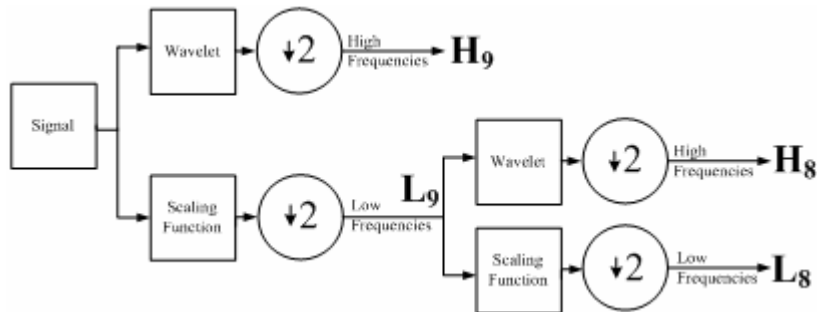


Figure 4.14 A two level wavelet transform

This process may be repeated further if desired, until, in the limit, the sub-band that contains the low-frequencies, contains only 1 sample (L_0 , which is equivalent to the 'DC'-coefficient (section 3.5.2) or the average value of the entire image).

4.6 The Wavelet Transform in Two Dimensions

An image is a two-dimensional signal. So how do we apply a wavelet transform to a two-dimensional image? It is possible to construct two-dimensional wavelets, but the transform is separable, so generally the two-dimensional image is handled by

² Downsampling will not be explained in further detail in this thesis. A more thorough explanation can be found in [3], chapter 17 (about Quadrature Mirror Filters, page 280-281).

4 Fundamentals of Wavelets

applying the wavelet and scaling functions in both the horizontal and vertical directions. The image is split in two parts, high frequency and low frequency operating, say in the horizontal direction first. The two resulting subimages contain both high- and low-frequency vertical information. Each of the subimages is now convolved with the wavelet and the scaling function vertically, each producing two new separations. The process is shown in Figure 4.15.

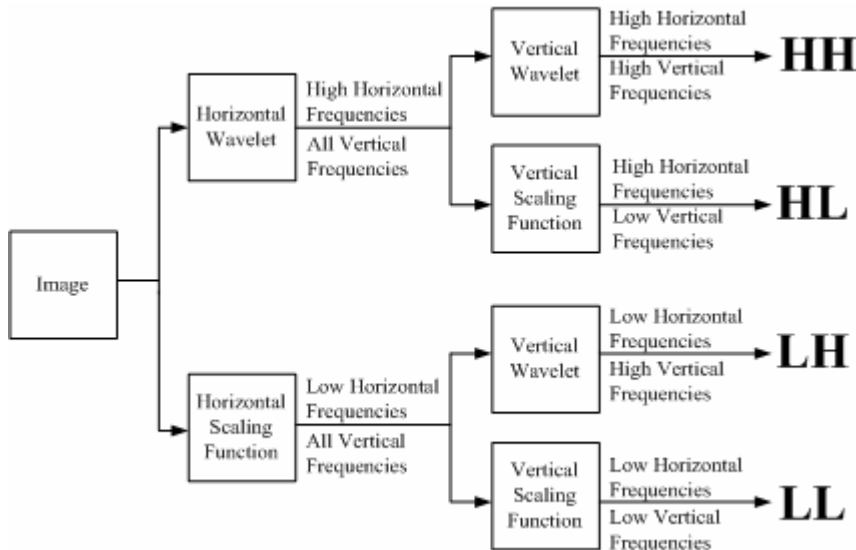


Figure 4.15 Division of the image into four subimages [7, 8]

After this process, we have four subimages. The downsampling process is not visible in this figure, but it has been applied, both horizontally and vertically in the same way as in Figure 4.13. Each subimage will therefore have one quarter of the samples of pixels of the original.

So, we have seen that a single-stage wavelet transformation consists of a filtering operation that can decompose a two-dimensional signal into four frequency bands.

Figure 4.16 show the result of a transformation on an image, (a) is the original image, and (b) shows the result of a single-stage wavelet transform. The top-left corner (*LL*) is the original image, low-pass filtered with the scaling function and subsampled in the horizontal and vertical dimensions. The bottom-left corner, *LH*, contains residual horizontal frequencies. The top-right corner consists of the residual vertical frequencies, *HL*, (for example the vertical bars in the background are visible here) whilst the bottom-right corner contains residual diagonal frequencies, *HH*.

As shown in Figure 4.14 we can repeat the decomposition process for the low frequency band. For a two-dimensional signal we decompose the *LL* component to produce another set of four sub-bands: a new *LL* band that is a further subsampled version of the original image, plus three more residual frequency bands. Repeating the decomposition three times gives the wavelet representation shown in Figure 4.16 (c). The small image in the top left is the low-pass filtered original and the remaining squares contain progressively higher-frequency residual components. Each sample in (b) and (c) represents a wavelet transform coefficient.



Figure 4.16 (a) The original image (Lena) [15]

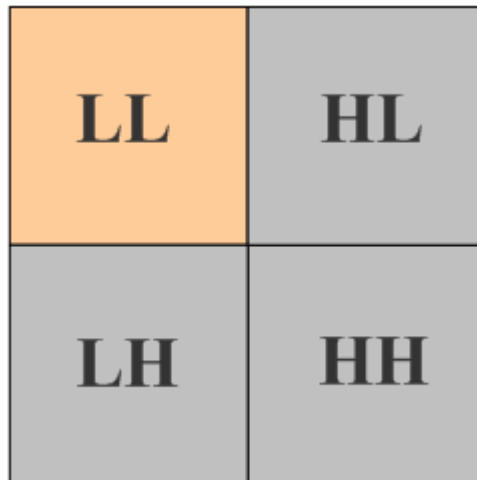


Figure 4.15 (b) A single pass of the transform, showing the three sets of coefficients, and the residual filtered image. [7, 15]

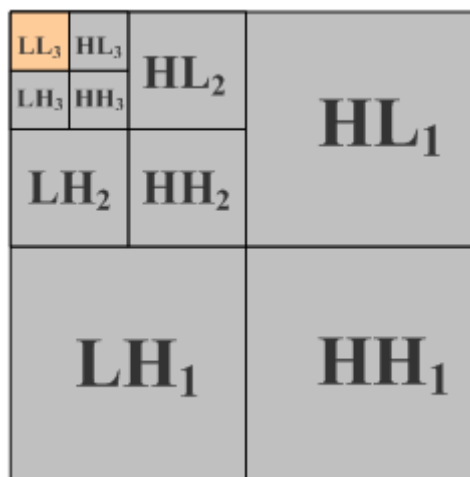
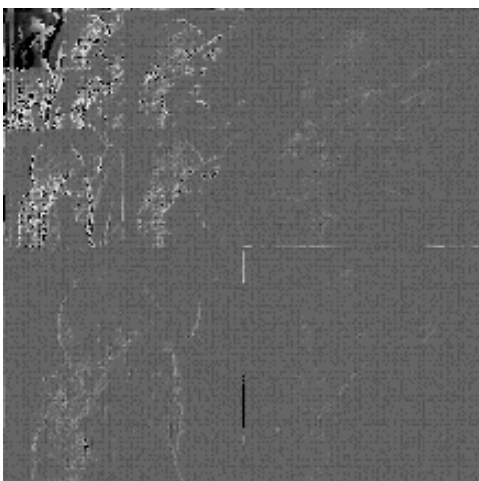


Figure 4.15 (c) The example shows three iterations and the resulting ten subbands. [7, 15]

4.7 Inverting the Wavelet Transform

A critical feature in compression is the possibility to reconstruct the signal perfectly. A wavelet algorithm has perfect reconstruction when the transformed signal yields exactly the original signal after transforming it back to the original domain (usually after performing some operation on it) by applying an inverse wavelet transform. This process is also known as a synthesis, which is the inversion of an analysis (the forward transform process). This means that the transform has to be invertible. The construction of high-pass and low-pass reconstruction filters is highly complex, and is only comprehensible in the context of the Fourier transform. We will settle with stating that the DWT is invertible when certain mathematical conditions are fulfilled. Haar is the simplest wavelet filter that shows perfect reconstruction.

To find the inverse transform, the decomposition process is reversed, as represented in Figure 4.17. This shows a one-dimensional signal, but this process can easily be generalized into the case of two-dimensional signals. This process continues until the last output is the original signal. Note that if no data processing is done, the transform – inverse transform process is lossless

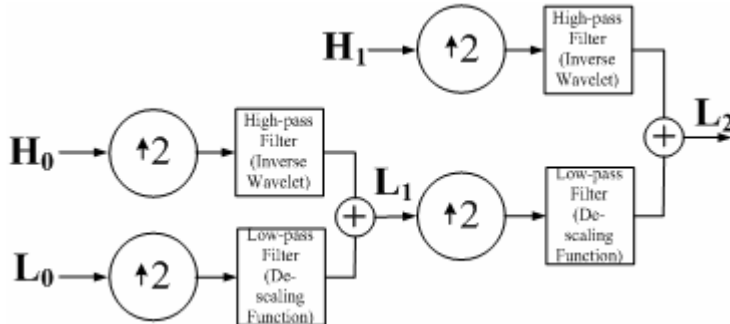


Figure 4.17 The first two levels of the inverse transform.

Part 2 Trends in Wavelet-Based Video Compression Research

5 Introduction

Until recently, the trend in wavelet video compression research have been to use wavelet to the transform coding but not to use wavelets qualities in Motion Compensated Prediction, but researchers have started to focus more on the latter approach.

Even if wavelet in video compression lately have been a common research topic, most video compression tools available is based on DCT. If you do a quick search on the internet, some wavelet based video compression tools are available though, but the tools publicly available are only intra-frame based. Very few, if none, present a commercially available product based on inter-frame video compression, even if this has been a research topic for a decade now.

The motivation for this work is therefore to establish whether wavelet based video coding have the potential of producing the same good results as wavelet in still image coding has proven to do.

5.1 Existing Wavelet-Based Video CODECs

There are no standardized video CODECs today that use wavelet based transform coding, but some implementations with wavelets exist. Some companies have developed wavelet based video compression products. The initial use of wavelet in video was with the old IMMIX Cube NLE systems, which used wavelet to provide better video quality at lower bit rates than the competing Motion-JPEG, but due to some other problems with the IMMIX, it lost the battle against M-JPEG. But wavelets did not loose though, and it is implemented as the chosen transform coder in Motion-JPEG2000, the video extension of JPEG-2000. Some other video CODECs that have implemented wavelets are Indeo v4 and v5. Aware has also made a software video codec, and as for the aforementioned CODECs, it only makes use of intra-frame compression, and a good inter-frame wavelet compression CODECs are still to be presented to the public.

5.2 Who Does What with Wavelet

Scientists from all parts of the world are researching on wavelets. Some are working independently, using their spare time and out of passion for this exciting field of research, whilst others are doing their research related to universities, corporate research labs or special interest organisations. The interest is huge, and the number of papers/research works is enormous.

MPEG and ITU-T decided to work together in JVT in 2001. They are currently working on the H26L standard, where the video compression algorithm will be founded on DCT. This does not mean that they have abandon wavelet totally. Both ITU-T and especially MPEG have done comprehensive research on wavelets in the field of video coding, but they have not yet chosen to implement it in their video coders. Bla Bla, say something about this...

When searching for papers in the field of wavelet and video compression, several other groups and research facilities have presented their work, especially in the IEEE database. Large companies like Sharp Labs, Philips, Sony, Digital Cinema and Microsoft are companies that we see have been very active in this field. This can give an indication that wavelet seems promising for commercial use, and that it is probable that support for wavelets based image and video coding will be incorporated in future hardware and software components.

Mobile phone companies like Nokia and Sony-Ericsson also seem to have taken interest in wavelets research, and assuming that this is a field that would benefit greatly of wavelets good compression properties, it will be exiting to follow the development of a possible wavelet CODEC inside a handheld device like a mobile camera phone.

5.3 What to Look for

There are a large amount of papers out there, so obviously just picking papers on luck will not be any good. We therefore had to start off by doing a wide search for wavelet and video compression, mainly this search was done by using the IEEE Explore database searching tool available from the IEEE web site. It is crucial that the gathered information is scientifically acknowledged in order to establish a well founded conclusion. We soon got an overview of what seemed to be the most respected researchers in our field of interest, and used this as our foundation for our future research.

6 Discrete Wavelet-based Transforms

In this chapter we first introduce the wavelet domain, and then depict a fast and efficient way to construct wavelets, called the lifting scheme. In the following we describe three different wavelet transforms; the three-dimensional DWT, the dual-tree complex DWT, and the overcomplete DWT.

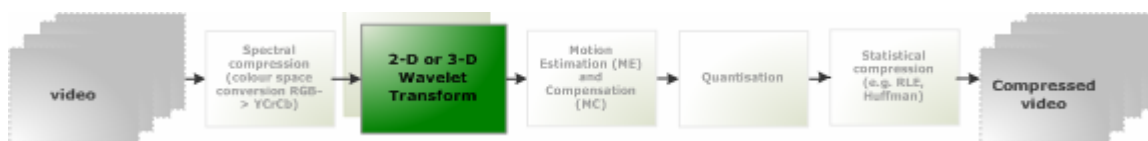


Figure 6.1 'Roadmap' – Architecture of a typical wavelet-based video coder

Note that when mentioning the classical, standard, or conventional DWT we refer to the convolution-based DWT as it is illustrated in chapter <2.x>.

We have been given the dual-tree complex DWT slightly more interest than the practical use in video coding justifies, because of its special nature, and the benefit it may have for other areas, like (medical) image processing. If interested in more information of the DT CWT and its applications, we recommend [20].

6.1 The Wavelet Domain

A huge amount of techniques takes place in the wavelet domain. Doing something in the wavelet domain, wavelet space, band-to-band, or in-band simply means that the process are performed after a wavelet transform is executed. As illustrated in Figure 6.1, state of the art wavelet-based video CODECs usually perform in-band ME/MC. When using a 3-D transform, doing something in the wavelet domain, means performing the algorithm after all three decomposition has been done.

Many techniques performed in wavelet domain do not depend on one specific wavelet transform, but rather of certain properties that the transform adds to the domain. Therefore, research papers presenting i.e. a new ME/MC algorithm or a new quantisation scheme, seldom mentions which wavelet transform is used, how it is constructed, or what type of wavelet filter the transform use. We will now briefly present some of the filters and transforms used in wavelet-based video coding.

One of the most favoured wavelet filters in image- and video coding when using the standard DWT is the biorthogonal Cohen–Daubechies–Feauveau 9/7 filter (also called the '9/7 filter' and 'Daubechies's 9-7 filter') [21], because of its good energy compaction properties. This filter is for example used in the JPEG2000 standard. Another popular filter in image compression is the Antonini (7,9) tap filter [22]. This is the filter that was used in the FBI fingerprint compression system. A third filter set, which is defined by the JPEG2000 standard, is the 5/3 integer transform [23], this is constructed by a lifting scheme, and was first proposed for image compression by Le Gall and Tabatai [24]. Figure 6.2 shows the wavelet and scaling functions associated with the 5/3 and 9/7 transforms defined by JPEG2000. As a digression, we will mention section 7.5 which briefly present a paper which discusses wavelet filters, and a way to choses wavelet filters adaptively depending on the statistical nature of the image being coded.

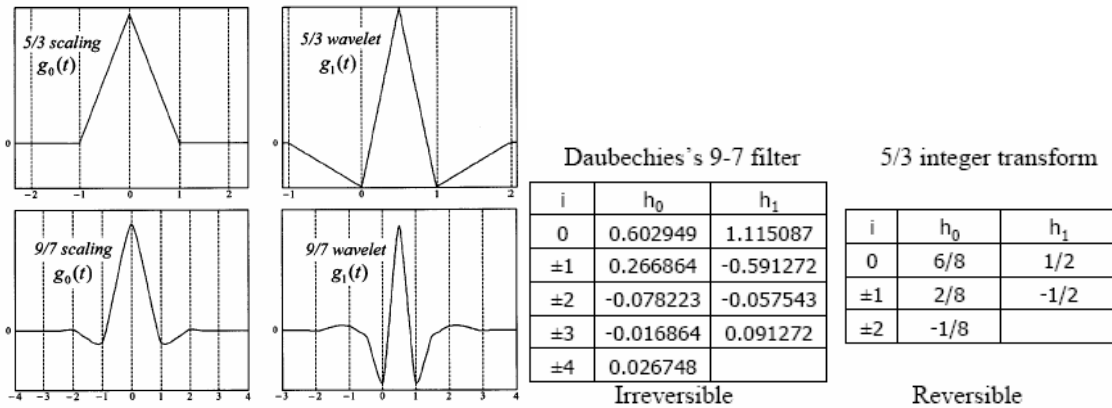


Figure 6.2 Synthesis scaling and wavelet functions, 5/3 and 9/7 subband filter sets

The classical DWT is, as we saw in <2.x Fundamentals of DWT>, based on convolution. However, there exist other ways to construct the same wavelets. In section 6.2 we depict a less computational complex way to calculate the wavelet transform, the lifting scheme. It seems that the use of lifting to perform the DWT has gained increased popularity in the image- and video compression community. As an example, lifting scheme wavelets form the basis of the JPEG2000 image compression standard.

A large number of recent developed ME/MC algorithms depend on a so-called shift invariant WT. The standard DWT is shift variant, therefore we have presented two different shift invariant wavelet transforms, a complex WT and an overcomplete WT, which in addition to shift invariance offers the important property of perfect reconstruction. As a digression, in the section depicting the overcomplete WT, we mention a transform called the Complete-to-Overcomplete DWT (CODWT) [25], this transform can be calculated in two ways; one using convolution and another using lifting.

All the transforms mentioned until now are two-dimensional (to be specific, they are separable, using one-dimensional WT in two directions). It is also possible to perform multiple-dimensional transform (also separable, using one-dimensional WT in multiple directions). The three-dimensional transform removes both spatial and temporal redundancies, and has been researched in a lot of papers; some are mentioned in section 6.3.

6.2 The Fast Lifting Wavelet Transform (The Lifting Scheme)

The original lifting scheme was introduced by Wim Sweldens in mid-nineties, and is a general framework to design biorthogonal wavelet filters which can be used in all types of discrete wavelet transforms. As opposed to classical constructions, the lifting scheme is not based on the Fourier Transform. This makes it possible to construct *second generation wavelets*; wavelets which are not necessarily translates and dilates of the mother wavelet. However, this scheme will never come up with wavelets that could not be found by [27]. These second generation wavelets can be used on places where no Fourier transform is available; for example on bounded domains in applications such as data segmentation, to analyze data that live on curves or surfaces, or for adapting to irregularly sampled data [33].

The main advantages lifting offers wavelet-based video CODECs is the reduction of computationally complexity. It allows a faster implementation and a fully in-place calculation of the wavelet transform. The latter removes the need for auxiliary memory in order to store temporary data during the calculation. In [34] it is proven that for long filters, the lifting scheme cuts computation complexity in half, compared to the standard iterated FIR filter bank algorithm (the classical DWT). This type of wavelet transform is already much more efficient than the Fast Fourier Transform (FFT), and lifting speeds things up with another factor of two, due to the fact that the lifting scheme makes optimal use of similarities between the odd and even values, or high and low pass filter.

For classical wavelet transforms one have to use the Fourier transform to see that an inverse wavelet transform yields exactly the original signal [33]. But by using lifting, it is easy to see that a given wavelet filter could be perfectly reconstructed by undoing the operations of the forward transform. As seen in Figure 6.4 the inverse wavelet transform is just a mirror of the forward transform in Figure 6.3.

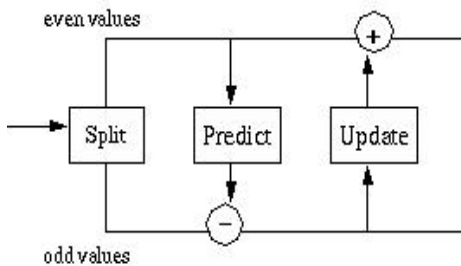


Figure 6.3 Lifting scheme forward WT

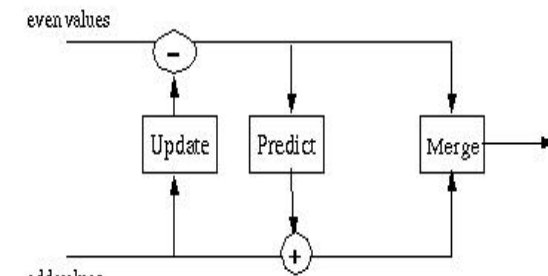


Figure 6.4 Lifting scheme inverse WT

The original lifting scheme has been further developed, and several different lifting schemes now exist. They will not be described here, but as an example, integer lifting [Cal96], [Uyt97b] and multidimensional lifting [Kov97], [Uyt97a] are quite recent and interesting developments.

6.2.1 Constructing wavelets with the lifting scheme

The name 'lifting' comes from the way the wavelet is constructed, it starts with a simple wavelet, often called the 'Lazy wavelet', which does not do anything, however it has the formal properties of a wavelet. Then the lifting scheme gradually builds a new wavelet by adding in new basis functions, improving the properties of the 'Lazy wavelet'. We will now briefly explain how the basic idea of the lifting scheme works in practise by using an example. Since the inverse transform can be found by reversing the forward transform, and simply change each '+' into a '-' and vice versa, we concentrate on explaining the forward process.

The lifting process starts with splitting the even (a) and odd (b) samples, followed by two steps, a predict and an update step as illustrated in Figure 6.5.

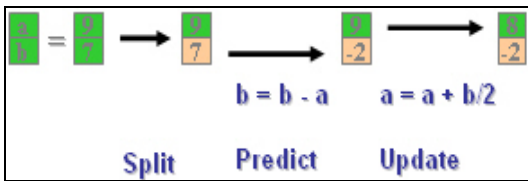


Figure 6.5 Forward steps [15]

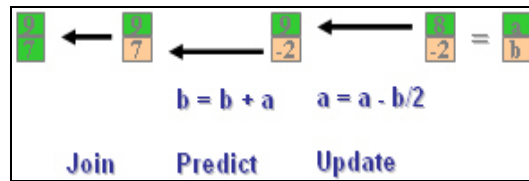


Figure 6.6 Inverse steps [15]

- We start by splitting the two samples in Figure 6.5, the odd sample is the green 9, and the even sample is the peach-coloured 7.
- Then we predict the new even sample by subtracting the original even from the original odd, $7 - 9 = -2$.
- We directly update the odd sample, by adding half the new predicted even sample to the original odd, $9 + (-2/2) = 8$.
- As illustrated in Figure 6.6, to get back to our original values, the process is simply inverted.

In Figure 6.7 and Figure 6.8 we see the lifting scheme working on 8 samples in several levels. The updated odd (green) samples are used as the input for the next level. These are split, predicted and updated as in the first pass. This process can continue until we in the end has one single odd element left.

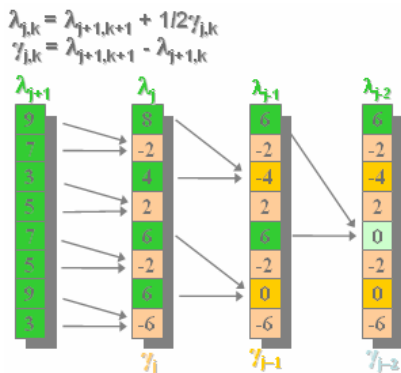


Figure 6.7 Predicting and updating coefficients in 3 levels [15]

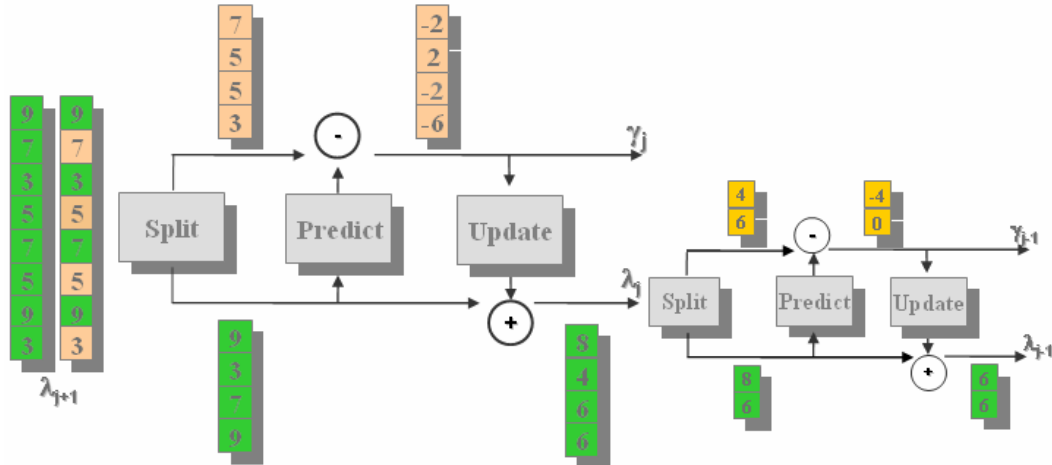


Figure 6.8 Split, Predict, and Update in 2 levels [15, modified]

6.3 Three-Dimensional Wavelet Transform (3-D DWT)

Karlsson and Vetterli first proposed the use of a separable three-dimensional (3-D) discrete wavelet transform (DWT) for video compression.

6.3.1 3-D SPIHT (Set Partitioning In Hierarchical Trees)

Pearlman and Kim have extended the popular 2-D SPIHT image coder (described in section 7.3) to a 3-D video coder [35]. 3-D SPIHT, among others [36][37], performs quite well because the energy between frames is compacted by a wavelet transform, which in turn can be quantized and entropy coded in order to obtain good compression ratios. This is a more natural way of removing temporal redundancy.

Typically a three level three-dimensional wavelet decomposition (like the one shown in i) with the 9/7 biorthogonal wavelet filters is performed on a Group of Frames (GOF) of size 16, as shown in Figure 6.10. First the GOF is temporally transformed (1-D) followed by a spatial domain transform (2-D). The temporal transform could be done by applying the wavelet filter to all the frames in the video sequence, but this obviously would result in large delays because all the frames have to be traversed before any transmission could begin. Also a lot of memory is needed when transforming large video sequences, possibly containing millions of frames. Therefore the frames are divided into group of typically 16 frames, in order to be able to do an effective temporal transformation in systems with limited amount of memory.

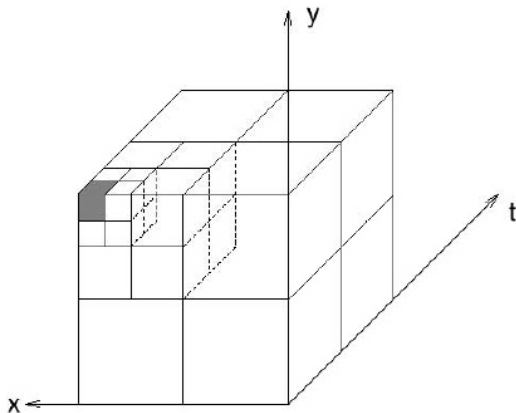


Figure 6.9 3-D Wavelet Decomposition [35]

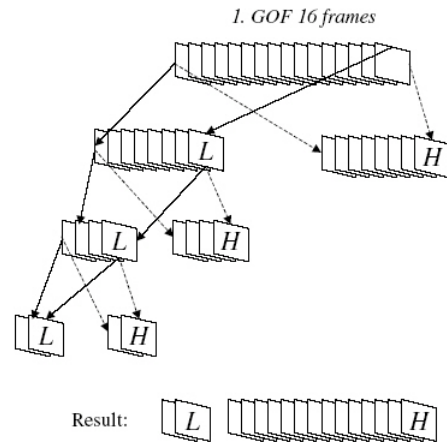


Figure 6.10 Temporal decomposition of a group of frames (GOF) [35]

As for 2-D SPIHT, 3-D SPIHT also produces an embedded bit stream, in order to provide a progressive video transmission, which makes it very scalable and also being able to choose an appropriate video quality. 3-D SPIHT supports separation of colour and luminance, which again stresses that the coding is very scalable in that both colour and luminance (monochrome) video can separately be extracted from the same bit stream. If a system has a limited bandwidth, it is possible to obtain a better resolution level, by using the luminance components only. As for 2-D SPIHT, the main feature of 3-D SPIHT is its speed and simplicity.

6.3.2 Lifting-based Invertible Motion Adaptive Transform (LIMAT)

LIMAT was introduced by Secker and Taubman in [38]. Taubman is an authority on use of the wavelet transform in highly scalable image- and video compression, and has presented several research papers in this field, including EBCOT [39] which is the core of the JPEG2000 image compression standard.

LIMAT is a framework for constructing three-dimensional transforms, based on any temporal wavelet kernel and motion model, while retaining perfect reconstruction. This invertibility property is inherited from a lifting realization of the three-dimensional DWT, in which each lifting step is compensated for the estimated scene motion. Incorporation of sophisticated motion models allows the transform to adapt to complex motion, which is demonstrated in [38] with a deformable mesh motion model. The LIMAT framework can be used with any temporal wavelet kernels, consistently superior performance is observed in [38] with the 5/3 wavelet as compared to Haar. This differs from evidence reported in the context of block-based and framewarping approaches [39, 40].

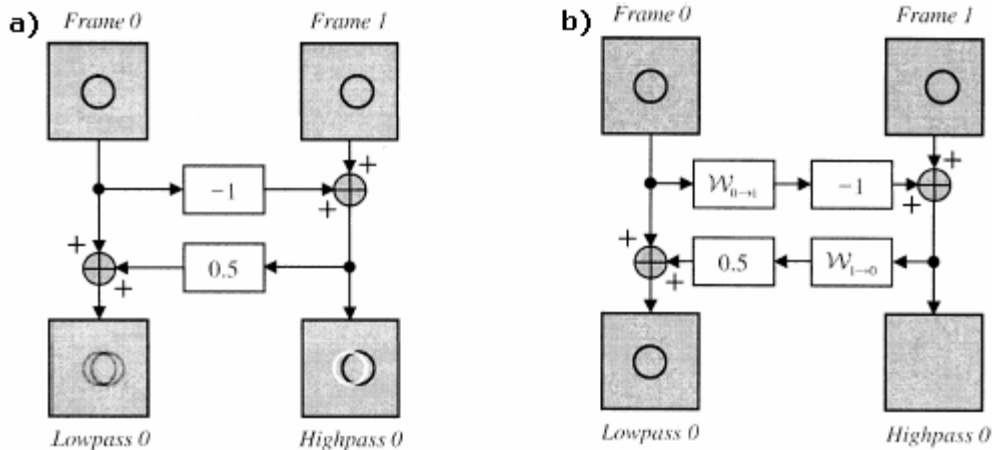


Figure 6.11 a) Lifting representation for the Haar temporal transform, b) same, but with motion compensated lifting steps

The 3-D-DWT is essentially applied in a separable fashion along the displaced blocks, but the effects of expansion and contraction in the motion field are observed by the appearance of “disconnected” pixels between the blocks. For the transform to remain invertible, these disconnected pixels must be treated differently, which seriously affects coding efficiency. In addition, perfect reconstruction is only possible with integer block displacements, although extensions to half-pixel accuracy have been demonstrated. These methods invariably involve block-based motion models, which cannot capture expansive or contractive motion.

Deformable meshes can improve motion compensation by tracking expansions and contractions, while maintaining a continuous motion field. More importantly, only continuous motion mappings allow us to understand the proposed transform as truly applying the temporal DWT along a set of motion trajectories. Experimental results in [38] reveal that this is particularly desirable for compression performance. However, without motion compensation, temporal filtering produces visually disturbing ghosting artefacts in the low-pass temporal subband. This is clearly undesirable where temporal scalability is of interest. The challenge therefore lies in finding a way to effectively exploit motion within the spatio-temporal transform

6.4 The Dual-Tree Complex Wavelet Transform (DT CWT)

Magarey and Kingsbury developed a motion estimation algorithm, using a 2-D DWT which is based on a complex-valued pair of 4-tap FIR filters with Gabor-like characteristics [41, 42, 43]. The aim of this work was to identify true motion as far as possible, using a hierarchical and phase-based approach. This ME algorithm is further depicted in a later section <>, but the complex-valued DWT they discovered are the subject of this section.

Magarey and Kingsbury was motivated by Fleet and Jepson [44] to use the efficiency of the DWT subband decomposition with complex-valued (Gabor-like) basis filters to provide the phase information required for their ME algorithm. “The desirable property of Gabor filters is that they are optimally localised in both spatial and spatial frequency domains [45].” [43, III section B.3] The 2-D CWT is implemented separably, so that only 1-d convolutions and downsampling are required. “The 2-D CWT may be thought of as producing a pyramid of complex subimages. At each level, the coefficients are formed by orientationally selective filtering of overlapping circular regions. There are six evenly spaced orientational subimages at each

pyramid level. Note that such directional filters are not obtainable by a separable DWT using a real filter pair, because its separable filters cannot distinguish between edge features on opposing diagonals (further explained in section 3.4.2 b)). Complex coefficients make this selectivity possible.” [43, III section B.2, slightly modified]

6.4.1 The Dual-Tree Implementation

The work with complex wavelets for motion estimation [41, 42, 43] showed that complex wavelets could provide approximate shift invariance and good directional selectivity. Unfortunately they were unable to obtain perfect reconstruction and good frequency characteristics when using short support complex FIR-filters in a single tree (e.g. Figure 6.12 Tree a). However, Kingsbury et. al. further develop the complex wavelet transform, and introduce the Dual-Tree implementation of a Complex Wavelet Transform (DT CWT) in [46-50].

The dual filter tree implementation is based on the idea that it is possible to achieve approximate shift invariance with a real DWT, by doubling the sampling rate at each level of the tree (but only if the samples are evenly spaced). The dual filter tree comprises two trees of real filters, a and b, which produce the real and imaginary parts of the complex coefficients. The key to successful operation of the DT CWT lies in the differences between the filters in the two trees. It is crucial that level 1 downsamplers in tree b picks the opposite samples to those in tree a, and at each level in both trees an additional delay difference is needed to obtain the correct total delay difference so that optimal shift invariance is achieved [48]. To invert the transform, perfect reconstruction filters are applied in the usual way to invert each tree separately and finally average the two results.

In the first form of the DT CWT [46, 47, 48] this was achieved by a simple delay of one sample between the filters at level 1, and then, for subsequent levels, alternate odd-length and even-length biorthogonal linear-phase filters were used to achieve the correct relative signal delays. But later Kingsbury et. al. discovered certain problems with the odd/even filter approach, and to overcome them, the Q-shift version of the DT CWT, with improved orthogonality and symmetry properties, was proposed in [49, 50]. This is seen in Figure 6.12. All the filters beyond level 1 are even length, but they are no longer strictly linear phase. Figures in brackets indicate the delay for each filter, where $q = \frac{1}{4}$ sample periods.

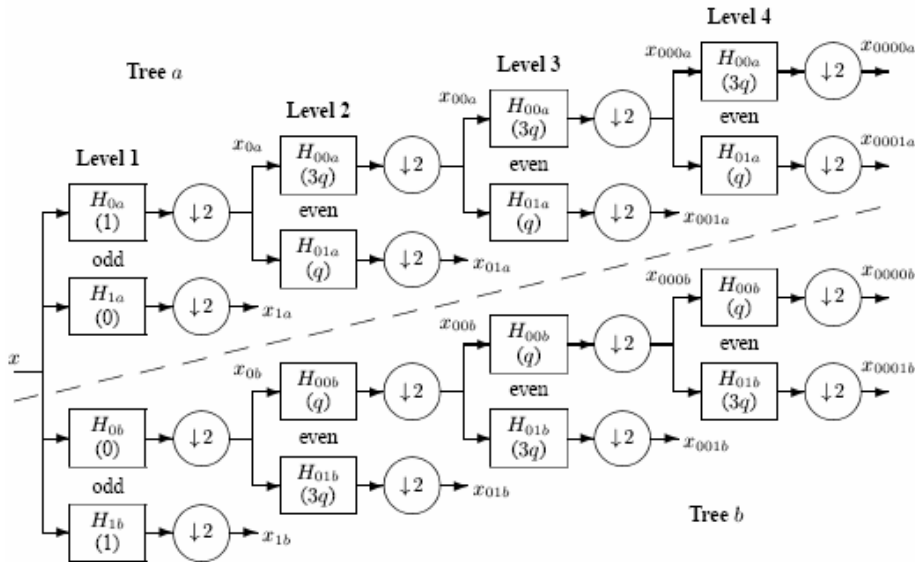


Figure 6.12 Dual tree of real filters for the Q-shift CWT [49]

In the Q-shift version, the filter coefficients are no longer symmetric, which makes it possible to design the perfect-reconstruction filter sets to be of even-length, shorter and orthonormal (like Daubechies filters) beyond level 1 so that filters in the two trees are just the time-reverse of each other, as are the analysis and reconstruction filters. They are designed with the additional constraint that the filter group delay should be approximately one quarter of the sample period. All this leads to a transform, where all filters beyond level 1 are derived from the same orthonormal prototype set, and in which the two trees are very closely matched and have a more symmetric sub-sampling structure.

6.4.2 Key features

The key features of the DT CWT [50] may be summarised as:

- a) Approximate shift invariance;
- b) Good directional selectivity in 2-dimensions (2-D) with Gabor-like filters (also true for higher dimensionality, m-D)
- c) Perfect reconstruction using short linear-phase filters;
- d) Limited redundancy, independent of the number of scales, 2:1 for 1-D ($2^m:1$ for m-D)
- e) Efficient order-N computation - only twice the simple DWT for 1-D (2^m times for m-D)

a) Approximate shift invariance

(The importance of shift invariance in wavelet-based video CODECs is explained in section 8.3.1 Shift Dependence.) The good shift invariance of the DT CWT is demonstrated in Figure 6.13, where it is compared with a standard DWT based on the (9,7) filters. Wavelet and scaling function components of 16 shifted step functions (top), for the Q-shift DT CWT (a) and real DWT (b), at levels 1 to 4 is seen. If there is good shift invariance, all components at a given level should be similar in shape, as in (a).

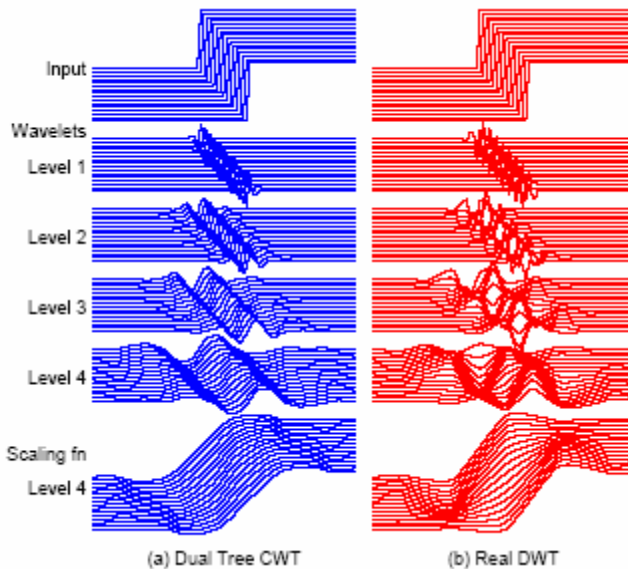


Figure 6.13 Wavelet and scaling function components of 16 shifted step functions for the Q-shift DT CWT (a) and real DWT (b) [49]

b) Good directional selectivity

Good directional selectivity is believed to be an important feature for many applications, including motion estimation and compensation, denoising, and edge enhancement. As an example, as shown in [47], the DT-CWT can outperform overcomplete real wavelet transforms because the improved directional selectivity provides better denoising near non-vertical and non-horizontal edges or lines.

In the standard DWT, separable filtering of the rows and columns of an image produces four subimages at each level, as we saw in <figure x.9, chapter x Wavelet fundamentals>. The *LH* and *HL* subimages can select mainly horizontal or vertical edges respectively, but the *HH* subimage contains components from diagonal features of either orientation. One way of explaining the poor selectivity of the standard DWT is that horizontal wavelet filters (real high-pass row filters) select both positive and negative horizontal high frequencies, while vertical wavelet filters (real high-pass column filters) select both positive and negative vertical high frequencies. Hence the combined *HH* filter must have passbands in all four quadrants of the 2-D frequency plane. On the other hand, a directionally selective filter for diagonal features with positive gradient must have passbands only in quadrants 2 and 4 of the frequency plane, while a filter for diagonals with negative gradient must have passbands only in quadrants 1 and 3. The poor directional properties of real separable filters make it difficult to generate passbands or directionally selective algorithms, based on the separable DWT. [51, section 4, slightly modified]

The DT CWT has much better directional resolution than a standard DWT, despite being implemented separably, as it possesses six directional subbands rather than three (for images) [52]. This is because the complex filters are able to separate positive and negative frequencies in 1-D, and hence separate adjacent quadrants in m-D frequency space.

Another approach both to shift invariance and to directional selectivity was pioneered by Simoncelli et al. [53], and was based on Laplacian pyramids and steerable filters, designed in the frequency domain.

c) Perfect reconstruction is critical in compression to reconstruct the signal, as explained in 4.7.

d) and e) leads to lower algorithm complexity and increased speed. [50]

6.4.3 Applications

"The shift invariant and directionally selective features of the DT CWT are believed to increase flexibility for spatially adaptive filtering of multidimensional signals without needing to worry about introduction of undesirable aliasing artifacts". [50, section 9, slightly modified]

The DT CWT is used in a variety of applications, such as motion estimation and compensation [41, 42, 43] (CWT), denoising and deconvolution [54, 55, 56], texture analysis and synthesis [57, 58, 59], segmentation and classification [60, 61, 62] and watermarking [63, 64]. Mostly, the main signals are images, but the DT CWT is also believed to be useful for video sequences and general 3-D datasets, such as medical scans and geological seismic data [50].

6.5 The Overcomplete Discrete Wavelet Transform (ODWT)

The Overcomplete DWT (ODWT) has a long history of development, and has been given several names, e.g the "undecimated DWT", the "redundant DWT" and the *algorithme à trous*.

The ODWT removes the downsampling operation from the traditional DWT presented in <fundamental of wavelet>, to produce an overcomplete representation. It can be considered as an approximation to the shift invariant continuous wavelet transform. (The importance of shift invariance in wavelet-based video CODECs is explained in section 8.3.1 Shift Dependence.) The downsampling and upsampling of coefficients is eliminated, and at each level, the number of output coefficients doubles that of the input. The wavelet and scaling filters are upsampled to fit the increasing data length.

The ODWT has been implemented in several ways. [65, section 5.5.2, ---23--- , 66] presents some classical and direct implementations of the *algorithme à trous*, where the filter-upsampling procedure inserts "holes" - "*trous*" in french - between the filter taps. The implementation in [66] results in subbands that are exactly the same size as the original signal. The advantage of this is that each coefficient is located within its subband in its spatially correct position. By appropriately downsampling each subband, it is possible to produce exactly the same coefficients as the conventional DWT. The output coefficients of the ODWT can also be represented in many ways. A popular scheme is to use a "coefficient tree". This tree representation is created by employing filtering and downsampling as in the usual critically sampled DWT; however, all phases of downsampled coefficients are retained and arranged as children of the decomposed signal. The process is repeated on the lowpass bands of all nodes to achieve multiple decomposition scales.

6.5.1 ODWT in video coders

The classical construction of the ODWT is trivial by using for example an "*à trous*" algorithm [55] or the LBS algorithm [68, 69]. However, in wavelet-based coding systems, the CODEC always processes the critically-sampled DWT subbands. Hence, the inverse DWT has to be performed first in order to reconstruct the input signal,

and then followed by the ODWT [26] [15]. For example, in the LBS approach [68, 69], phase shifting is implemented by an inverse transform, shifting in the spatial domain (i.e., with low band) and then performing a forward overcomplete transform.

Andreopoulos et. al., "complete-to-overcomplete DWT (CODWT)" [26] and Xin Li et. al., "performing phase shifting in the wavelet domain" [] have independently proposed similar direct solutions, which are much more computational effective than the conventional approach used of Kim and Park [68, 69]. The main idea behind these approaches is to produce the ODWT directly from the DWT. Specifically, the computation of the input signal is not required, and as a result, the minimum number of downsampling operations is performed and the use of upsampling is avoided. In addition, Andreopoulos et. al proposes in [26] an efficient scheme for the transform-calculation of the CODWT, testing both a convolution and a lifting based implementation.

The possible applications for the techniques briefly presented in this section are i.e. image- and video coding and compressed domain processing. The shift invariance and perfect reconstruction properties of the ODWT makes it very interesting for these applications, and it has recently been utilized by a large number of wavelet-based image- and video coding systems [1-16].

7 Wavelet-based Spatial Compression

In this chapter we will discuss how to remove spatial redundant data in digital image- and video signals using wavelet-based algorithms. Given that spatial compression can be performed by using the same approaches as in still image compression, we will therefore depict some state of the art wavelet-based still image algorithms and coders.

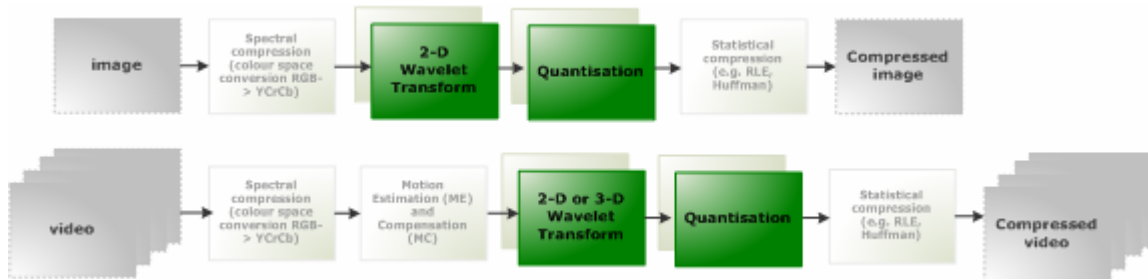


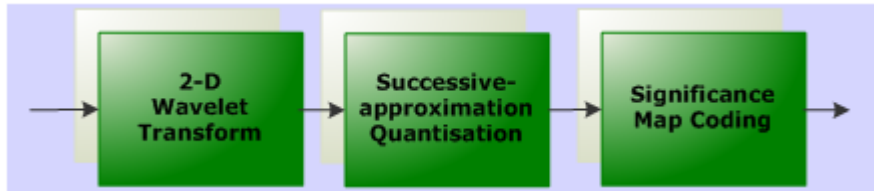
Figure 7.1 'Roadmap' - Architecture of typical image- and video coders

7.1 Prosperity of Wavelet-based Image Coders

Over the past few years, a variety of novel and sophisticated wavelet-based image coding schemes have been developed. These include EZW[23], SPIHT[22], SFQ[32], CREW[2], EPWIC[4], EBCOT[25], SR[26], Second Generation Image Coding[11], Image Coding using Wavelet Packets[8], Wavelet Image Coding using VQ[12], and Lossless Image Compression using Integer Lifting[5]. This list is by no means exhaustive and many more such innovative techniques are being developed as this article is written. We will briefly discuss a few of these interesting algorithms here.>

<Modern embedded image coders are essentially built upon three major components: a wavelet transform, successive-approximation quantization, and significance-map encoding. Below, we overview these components and describe how each are implemented within several prominent algorithms, including the recent

JPEG-2000 standard [1; 2]. Technique (SPIHT[4]) and a conditional-coding technique (JPEG-2000 [1]), as well as three other techniques based on other forms of significance map coding (WDR [7], SPECK [8; 9], and tarp [10]). All coders use a 5-stage wavelet decomposition with the popular 9-7 wavelet filters from [27].>



7.2 Embedded Zero-tree Wavelet (EZW) Compression

Lewis and Knowles [29] in 1992 were the first to introduce a tree-like data structure to represent the wavelet coefficients in the <octave-band wavelet decomposition> subbands. Later, in 1993 Shapiro [28] called this structure 'zerotree' of wavelet coefficients, and presented a elegant algorithm called 'Embedded Zerotree Wavelet' (EZW) algorithm.

The EZW is known to be a very simple and computationally effective algorithm [23]. As illustrated in Figure 7.2, every coefficient in the lower frequency band has several corresponding child coefficients in the higher frequency bands. The idea behind this approach is that if a parent coefficient is insignificant, then it is very likely that all its child coefficients are insignificant too. This makes it computationally effective, because it is not necessary to use a complex verification algorithm to check the significance of the child coefficients, if the parent coefficient is insignificant, we simply set it and its children to zero.

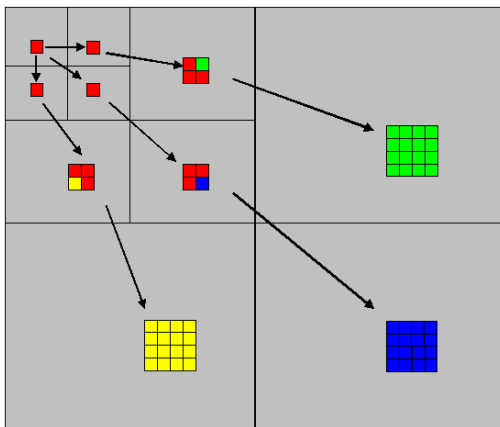


Figure 7.2 Relationship between parent-child regions in (2-D) DWT sub bands

Many insignificant coefficients at higher frequency subbands (finer resolutions) can be discarded, because the tree grows as powers of four. The EZW algorithm encodes the tree structure so obtained. This results in bits that are generated in order of importance, producing an embedded bit stream. The main advantage of this coding is that the encoder can terminate the encoding at any point, thereby allowing a target bit rate to be met exactly. Similarly, the decoder can also stop decoding at any point, because the image can be reconstructed before all the bits are transceived. However, the quality of the image increases as further bits are received, as illustrated in Figure 7.3.

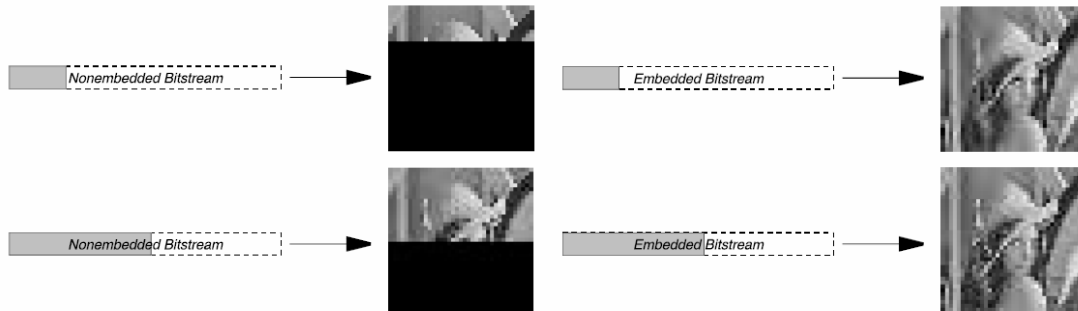


Figure 7.3 Non-embedded vs. embedded bit stream

7.3 Set Partitioning In Hierarchical Trees (SPIHT)

Many enhancements have been made to make the EZW algorithm more robust and efficient. One very popular and improved variation of the EZW is the SPIHT algorithm introduced by [35] in 1994.

In [35], the principles of EZW are explained in an alternative way, and can be summarized by the following three key concepts:

- partial ordering by magnitude of the transformed coefficients with a set partitioning sorting algorithm,
- ordered bitplane transmission of refinement bits,
- and exploitation of self-similarity of the image wavelet transform across different scales of an image

SPIHT offers a more effective implementation of the modified EZW algorithm based on set partitioning in hierarchical trees. The way coefficients are divided into subsets and thereafter partitioned has been improved, so it is even more likely that the most significant information will be conveyed first. For an image bit stream to be fully embedded, it is crucial that the most significant information is transferred first, i.e. the information that reduces the image distortion most, has to be assigned the first position in the transfer queue. The significance of the information is determined by a calculation of the mean square error (MSE) distortion measure.

SPIHT also includes a scheme for progressive transmission of the coefficient values that sorts the bits within each coefficient within each subband by magnitude, so the most significant bits in the most significant subband are transmitted first. This implies that even if a coefficient in a subband is more significant than another coefficient in the same subband, it is not given that all the bits in that coefficient is more important than every bit in the least significant. As illustrated in Table 7.1 the most significant bits are transferred first as indicated by the horizontal arrows. The most significant coefficient in this example is c_i , followed by c_j , c_k , c_l , c_m ... The bits in each coefficient are ordered after significance vertically. First all the most significant bits (msb) with significance 5 is transferred, then all the bits with significance 5, etc. Notice how the most significant bits (msb) in coefficient, c_k , are transferred before the least significant bits (lsb) in coefficient, c_i .

Table 7.1 Coefficients ordered by magnitude

Coefficients		C_i	C_j	C_k	C_l	C_m	...									
Sign		+/-	+/-	+/-	+/-	+/-	+/-	+/-	+/-	+/-	+/-	+/-	+/-	+/-	+/-	+/-
msb	5	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	4	→		1	1	0	0	0	0	0	0	0	0	0	0	0
	3	→			→	1	1	1	1	0	0	0	0	0	0	0
	2	→				→				1	1	1	1	1	1	1
	1	→														
lsb	0	→														

We will also mention another popular coding scheme Pearlman has been involved with, Set-Partitioning Embedded block Coder (SPECK), which uses quad-trees instead of zero-trees in the significance coding process [29][30]. Independent tests [31] have shown that this scheme offers approximately the same results as SPIHT.

7.4 EBCOT Coder in JPEG2000

JPEG2000 is the latest image compression standard to emerge from the Joint Photographic Experts Group (JPEG), and is the leading image compression standard for photographic pictures. JPEG2000 supports both lossy and lossless compression. Central to this standard is the concept of scalability, which enables image components to be accessed at the resolution, quality, and spatial region of interest. An example of ROI coding can be seen in Figure 7.4, notice how the rectangular ROI in the facial region is well preserved at the expense of the background. [24] All these features derive from a single wavelet-based algorithm, Embedded Block Coding with Optimized Truncation (EBCOT) [71]



Figure 7.4 Example of ROI coding at 0.125 b/pixel [24]

The EBCOT algorithm was introduced by Taubman in [71], and is related in various degrees to earlier work on scalable image compression; including the EZW algorithm, SPIHT algorithm, and Taubman and Zakhor's LZC (Layered Zero Coding) algorithm [71]. The key advantage of the EBCOT algorithm is as for SPITH, 'resolution' and 'distortion' scalability, provided by the multiresolutional wavelet transform and the embedded block coding. However, the EBCOT algorithm uses a more sophisticated approach for significance propagation of bits than the SPIHT algorithm. It takes

advantage of the fact that neighbouring pixels in images tend to be very similar to each other, by changing insignificant coefficients to be significant, if having at least one significant neighbouring coefficient.

The wavelet transform used in EBCOT is implemented both convolution-based and lifting-based. Lifting plays a central role, because it enables both state-of-the-art lossy compression and near state-of-the-art lossless compression. In EBCOT a 'reversible' transform is defined by exploiting an important property of the lifting structure; lifting filters may be modified in any desired manner without compromising invertibility. In this transform, the 5/3 integer transform, the output of each filter is rounded to an integer [75, 76]. In conjunction with the embedded quantization and coding strategies, this makes it possible to produce an efficient losslessly compressed representation of the image, which embeds any number of efficient lossy compressed representations.

It is shown that JPEG2000 perform fairly close to SPIHT and SPECK, with a slight advantage of rate-distortion performance [31]. However, these other implementations do not offer the same variety of features as JPEG2000. <regere enn baseline JPEG>

7.5 Adaptive Wavelet Coding of Multimedia Images

Contrary to the previous algorithms presented, the algorithms mentioned here concentrate on the wavelet filter, not quantization and context modeling of the transform coefficients.

[77] has experimented with a variety of wavelets to compress images of different types. It states that the performance of lossless coders is image dependent, and that the wavelet filter should be chosen adaptively depending on the statistical nature of the image being coded. They argue the importance of using *good* wavelet filters, since this may increase performance, even when using sophisticated quantization algorithms. Their results shows that while some wavelet filters perform better than others, no specific wavelet filter performs better than others on all images. Similar results have also been observed in the context of lossless compression using various integer-to-integer wavelet transforms.

8 Wavelet-based Temporal Compression

In this chapter we investigate research work done in the field of wavelet-based temporal redundancy removal. Initially we briefly explore the development in this field, and take a look at some reasons why wavelet-based representation has not achieved the same success in the area of video coding as for image coding.

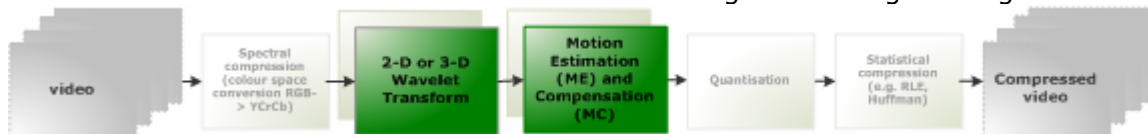


Figure 8.1 'Roadmap' - Architecture of a typical wavelet-based video CODEC

8.1 Prosperity of Wavelet-based ME/MC algorithms in Video Coders

MCP effectively exploits the redundancy in the temporal domain
 +
 WT effectively exploits the redundancy in the spatial domain

=

Consequently, a tantalizing question is how do we put ME/MC and WT together and make them work for videocoding?

Previous works of wavelet coding:

3D Wavelet-transform Coding (Ohm' 1994)

3D Subband Coding (Taubman and Zakhor '1994)

MC 3D Subband Coding (Choi and Woods '1999)

3D-SPIHT (Kim, Xiong and Pearlman '2000)

3D-ESCOT (Xu et al '2001)

Invertible MC 3D-WT (Hsiang and Woods '2001)

8.2 Wavelet-based Motion Estimation and Compensation in Spatial Domain

Most existing video coding standards (including MPEG and H.26x series) share a similar structure: motion compensated prediction (MCP) is first performed in the spatial domain and then motion-compensated residues are compressed by DCT-based coders. The most straightforward way to replace the DCT with a DWT in a typical video coder would be to perform ME/MC in the spatial domain and to calculate a DWT on the resulting residual image, as in [78-80].

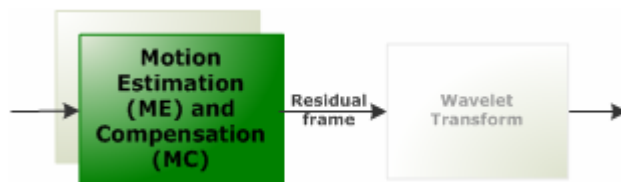


Figure 8.2 ME/MC in spatial domain

However, it has been shown (e.g. [73, 74]) that this simple approach has certain drawbacks. Conventional block-wise motion compensation does not fit to wavelet transform coding, due to blocking artifacts which are made worse when the DWT is deployed as is usual as a full-frame transform. It has been proposed [79, 80, 81] to use overlapped block motion compensation (OBMC) in the spatial domain before the DWT, to reduce these blocking artifacts.

Moreover, when trying to preserve the multi-resolutional structure of each frame and obtain the scalability that the wavelet transform provides, another issue become apparent. It seems that when the decoder operates at a lower resolution, drifting becomes a serious problem, since motion compensated prediction assumes the knowledge of the previous frame at the full resolution.

Another drawback with this approach is that the wavelet basis is not suitable for representing the predicted residual image [29]. The main benefit of good transform coding is that the image energy becomes highly compacted, with a large portion of information represented in the lower frequency areas. But if we have two almost identical frames, and the one is subtracted from the other, the residue image does not have the same properties as a normal image. Most of the low frequency information is gone, and it is therefore hard to say what information to remove in order to obtain good visual quality. At the same time, significant coefficients are most likely scattered around, and not concentrated around the top left corner of the matrix, which makes it hard to perform efficient entropy coding prior to transmission.

8.3 Motion Estimation and Compensation in Wavelet Domain

An alternative paradigm would be to perform ME/MC after a wavelet transform is applied to all frames.

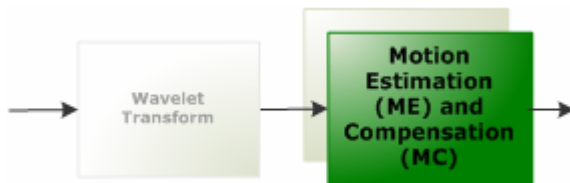


Figure 8.3 ME/MC in Wavelet Domain

This concept eliminates the inefficiency due to high-frequency blocking artifacts. More important, perhaps, is that resolution-scalable coding without drift becomes possible. However, it has been shown that this approach also leads to inefficient coding. Due to frequency aliasing effect introduced by the down-sampling operation³ ME/MC in the wavelet domain has been considered complex and inefficient until recently. This aliasing effect, also commonly known as shift variance, is explored in the following section.

8.3.1 Shift Dependence

We recall that in compression it is critical that the wavelet algorithm used can be reconstructed perfectly, as seen in Figure 8.4.

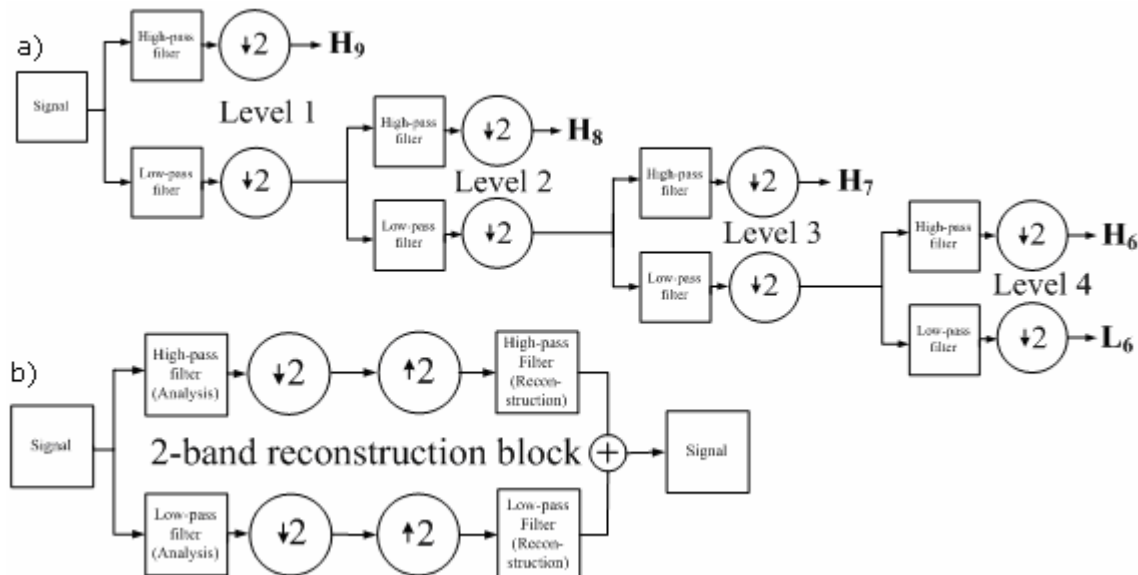


Figure 8.4 a) Four level binary wavelet tree. b) The filter bank, used to achieve perfect reconstruction from an inverse tree

In Figure 8.5, we see that there is a strong similarity between the shapes of the various wavelets; due to the fact that perfect reconstruction constrains each filter in Figure 8.4 to be approximately a half-band filter [31]. This causes aliasing and results in severe shift dependence of the wavelet transform [2, <Andreopolus>, 31].

³ See chapter x.x and evt. appendix hvis jeg lager det

8 Wavelet-based Temporal Compression

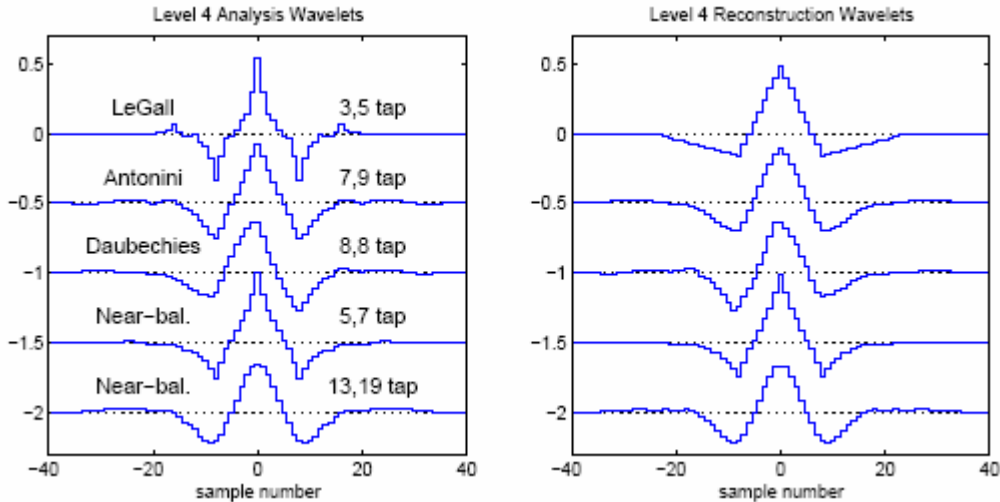


Figure 8.5 Comparison of some common wavelets [20].

Shift variance is a disadvantage of discrete wavelets: the resulting wavelet transform is no longer shift invariant, which means that the wavelet transforms of a signal and of a time-shifted version of the same signal are not simply shifted versions of each other, as demonstrated in Figure 8.6 and Figure 8.7. In the first figure we see a signal and a one sample shifted version of the signal. A 1-D DWT, using the 9/7 filter [14], is performed on both, and the resulting coefficients are displayed in Figure 8.7. It is easy to determine the “motion” of the signal waveform when comparing the original signal with its shifted version, but in the wavelet domain we can see that although there is still some correlation between low-band outputs, the high-band signals are completely dissimilar. Because the signals represented in wavelet domain suffer from the shift-variant characteristic of the DWT, it is not possible to obtain accurate motion vectors for ME using either the low-band or high-band signals in the DWT domain.

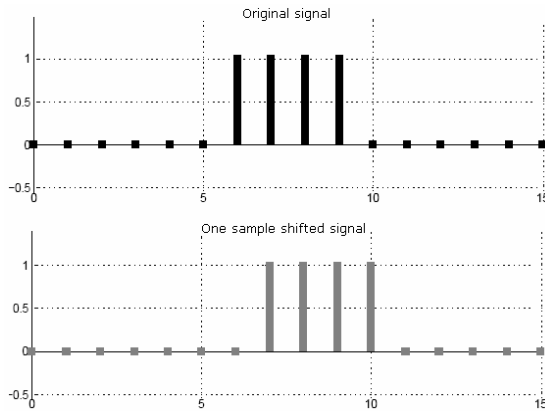


Figure 8.6 Original signal and shifted version

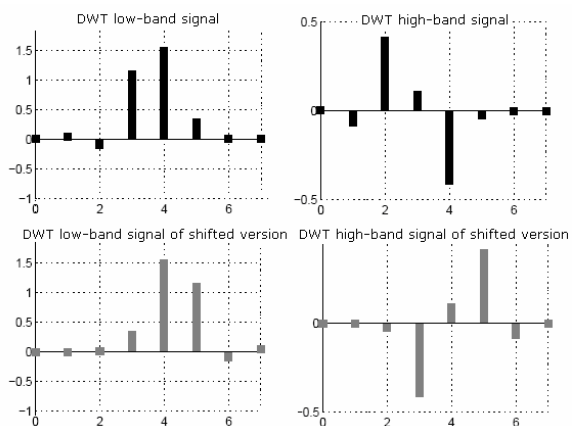


Figure 8.7 Wavelet domain representation

So why does this happen? “When we analyse the Fourier spectrum of a signal, it is expected that the energy in each frequency bin is invariant to any shifts of the input in time or space. But when using the DWT to analyse signals - even though they have perfect reconstruction properties, they do not provide energy shift invariance separately at each level. The energy distribution between the various wavelet levels depends critically on the position of key features of the signal. Ideally we would like it to depend on just the features themselves.” [31, section 2.3, some modifications]

Figure 8.8 illustrate this significant drawback with another example. In (a) we see the resulting signal, analysed with a 4-level DWT using Antonini (7,9)-tap filters assuming that the wavelet coefficients are computed at full input sampling rate. In practice, the samples are yielded as in (b) (1/16 of the sampling rate). If we time-shift the input step, we obtain the samples shown in (c), these are the ones represented as circles in (a). If we compare the total energy of the samples in Figure 8.8 (b) with the total energy of the samples in (c), we see that the energy variation is considerably larger.

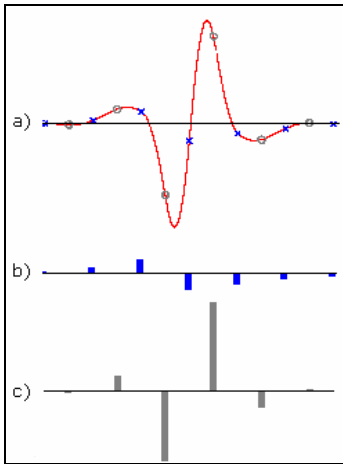


Figure 8.8 Step response of level-4 Antonini wavelet (wavelet domain representation)

Because of this shift dependence, caused by aliasing due to downsampling at each wavelet level, conventional DWTs are unlikely to give consistent results when used to detect key features in images, or to obtain accurate motion vectors for ME.

Recently, there have been several successful attempts to overcome the shift variance of DWT in motion estimation. In chapter 6.4 and 6.5 we took a glance at two shift invariant wavelet transforms, the DT CWT and the ODWT, which has been essential in this development.

8.4 Motion-Compensated Temporal Filtering (MCTF)

Girod identified that even if a three-dimensional transform is used to perform spatial and temporal compression in natural video sequences, motion compensation is an essential step for efficient decorrelation of the video information along the temporal axis [7 i Andreopoulos]. Moreover, without motion compensation, three-dimensional transforms introduce visually disturbing ghosting artefacts in the low-pass temporal subband [TAUBMAN-LIMAT]. The challenge therefore lies in finding a way to effectively exploit motion within the spatio-temporal transform.

A number of pioneering works effectively incorporated motion compensated steps in the temporal transform [8] [9], leading to a class of algorithms that perform motion-compensated temporal filtering (MCTF). MCTF is usually performed in the temporal domain prior to the spatial DWT, quantization and coding [8-11].

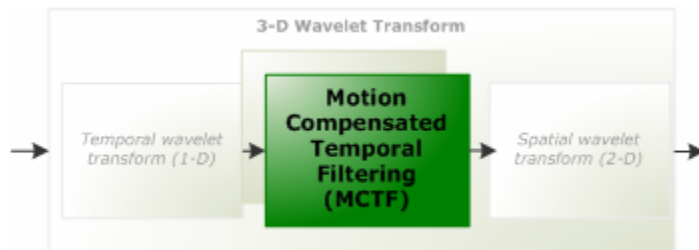


Figure 8.9 MCTF incorporated in a three-dimensional wavelet transform

Part 3 Java Prototype – MediaCODEC

9 Foundation

We have implemented a video compression prototype founded on a DCT- and DWT-based still-image compression application developed by four engineer students in a previous project. We have incorporated the DWT-based compression algorithms in our prototype, by tearing application apart, and reconstructing the interesting parts inside an adapted Java Media Framework API (JMF API 2.1.1). We have designed a highly extendable prototype considering the advantages and disadvantages of the research study presented in Part 2. We carefully planned a stepwise implementation progress, to help us make sure making the prototype extendable. Having mainly focused on doing a thorough design work and a literature study, the entire plan has not been implemented. However, this was never the intention. We wanted to make a foundation that could be utilized in future studies, both for our one interest and others.

9.1 *The Java Programming Language*

We have chosen to implement our prototype in the Java programming language. Java is not the obvious first choice when building a video CODEC, so in this part we try to explain our choice. The Java programming language has some evident strengths and drawbacks. One of its main features is platform independence [1], but the price is relatively slow execution [2]. Several projects at Agder University College using Java combined with wavelets for image processing experienced that this combination resulted in long execution time, including Geir Broms Fløystad, who worked with [15]. So when having to combine this with the high computational complexity of a video CODEC application, we were quite concerned about the result.

Our initial choice when we started to plan this CODEC was to use C or C++, but a few reasons made us change our minds; Firstly, having some experience with an early version of an additional API for Java, Java Media Framework (JMF) from Sun [4], made us believe that this API might give us a headstart when building the different parts of the CODEC. A brief study confirmed that this API had gone through major development the last couple of years [5, 6], and it seemed really interesting for our purpose. Secondly, our mentor, Per H. Hogstad, had access to several wavelet applications implemented in Java. Some of these applications were implemented by his students, and they could offer a variety of wavelet filters and useful functions. Among these applications was a still-image compression application using DCT and DWT, developed at Agder University College in 2001 by four engineer students [11]. This project achieved really good results, and we decided to take a closer look at it.

So, considering these advantages we started leaning towards using Java. But we were still concerned for the performance of the CODEC. We continued our investigation and found that a common solution for high-performance JVMs (Java Virtual Machine) is to use dynamic translation or just-in-time (JIT) compilers. These generate machine code on the fly for a method if it been called enough times, and future calls to the method execute the machine code directly. We also found compilers which can compile Java source code directly to native machine code. As an example, GNU (G) has an ongoing project, GCJ (the GNU Compiler for the Java language), to develop a compiler for the Java programming language. It makes it possible to compile Java source code directly to native machine code [8]. Unfortunately, the GCJ runtime, libgcj, which among other provides the core class libraries, does not support all the Java APIs we have need for. In addition we also

Part 3 Java Prototype – MediaCODEC

feel it is too experimental for our project; we would not risk having the kinds of problems it could give us.

Finally, we found that it is possible to make call to libraries written in another language inside Java, by declaring some Java methods to be native. This makes it possible to do 'low-level performance hacks' for more efficient coding. In 1997, Sun released the Java Native Interface (JNI), which is a standard for writing native methods in either C or C++. The main goal of JNI is portability in the sense that native methods written for one Java implementation should work with another Java implementation, without recompiling. [8] claim that JNI is very slow because everything is done indirectly using a table of functions to ensure JNI's portability. But studying JMF source code we found that encoding/decoding is done via native methods, which should indicate that it is fast enough for our purpose.

Finding that there exist several solutions to improve Java's 'speed performance'; we decided to start programming in Java.

9.2 Java Media Framework (JMF)

It is obvious that in order to implement and test a video CODEC it is necessary to be able to open and play movie clips. This would certainly be a time-consuming task to implement from the start. From our early experience with JMF 1.1 API, we knew it enabled us to present movie clips and other time-based media in an easy way. But we had no idea of how much processing it allowed us to do on the media. Seem to remember that back in 1999, Sun were working on an extension of the framework, launching a beta version just around the time when finishing that previous work. We took a dive into the Java Media Framework API.

The Java Media Framework (JMF) is an extension to the Java 2 APIs, and has to be installed separately <skriv om dette litt!> [6]. It was released by Sun in 199x, and made it possible to present multimedia in Java applications and applets. x years later, a major improvement was released, the JMF 2.0 API, which provides support for capturing and storing media data, controlling the type of processing that is performed during playback, and enables custom processing on media streams. [5]

JMF is design so it can decide at runtime what code to use to handle a given format, multiplex scheme, and encoding. This is done by carefully parcelling out responsibility for format handling, demultiplexing, decoding, and rendering into different classes, then using reflection to discover what kinds of handlers are available.

9.3 The Still-Image Compression Application

In 2001 four engineer students at Agder University College developed a still-image compression application⁴ as part of their finishing project to complete a 3-year college program [11]. The aim of the project was to explore different still-image compression techniques.

⁴ We have made an effort to explain the key aspects of the implementation of this application. This is because we for the time being re-use the DWT compression process (a future goal would be to enhance these according to our literature study), and therefore the concepts described here also holds for our prototype. In addition, since [11] is written in Norwegian and may be hard to acquire, we want to provide the necessary information for possible readers that do not speak Norwegian.

The image CODEC was based on two different transforms; DCT (Discrete Cosine Transform) and DWT (Discrete Wavelet Transform). In addition to the transform algorithms, several other compression algorithms, including Traverse coding, RLE (Run Length Encoding), and Huffman, was implemented. Some conceptual class-diagram constructed to get better understanding of the program was made and has been provided in Figure 9.2 and Figure 9.3.

In this section we will take a brief walk-through of the processes and algorithms which are essential to us. As we are concentrating on wavelets in this thesis, we will only implement the DWT transform. Figure 9.1 shows a schematic representation of the encoding process. To decode an image compressed by this application, this process is reversed. In the still-image compression application, all the separate processes, except the DWT, are optional.

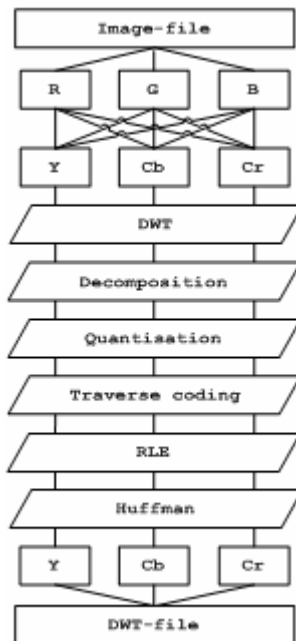


Figure 9.1 DWT from file to file [11]

Figure 9.4 is a conceptualized sequence diagram, showing the most important classes, methods and messages when the encoding process represented in Figure 9.1 is executed. This process is started when the user click the 'Kompress' button in the DWT-option dialog in the still-image application, making call to the `DWTCodec.Compress(...)` method.

9.3.1 Converting colour space from RGB to YCbCr

The conversion from RGB to YCbCr is done by going through every pixel in the image, and converting the three elements between R, G and B; and Y, Cb and Cr, by using the following equations:

<sjekk at formlene er riktige, s. 79>

From RGB to YcbCr:

$$\begin{aligned}
 Y &= 0.301 * R + 0.586 * G + 0.113 * B \\
 Cb &= -0.172 * R - 0.340 * G + 0.512 * B + 128 \\
 Cr &= 0.512 * R - 0.430 * G - 0.08 * B + 128
 \end{aligned}$$

From YcbCr to RGB:

Part 3 Java Prototype – MediaCODEC

$$R = Y + 1.371 * (Cr - 128)$$

$$G = Y - 0.698 * (Cr - 128) - 0.336 * (Cb - 128)$$

$$B = Y + 1.732 * (Cb - 128)$$

In the application, the colour space conversion is implemented in the static methods `Codec.RGB2YCbCr` and `Codec.YCbCr2RGB`.

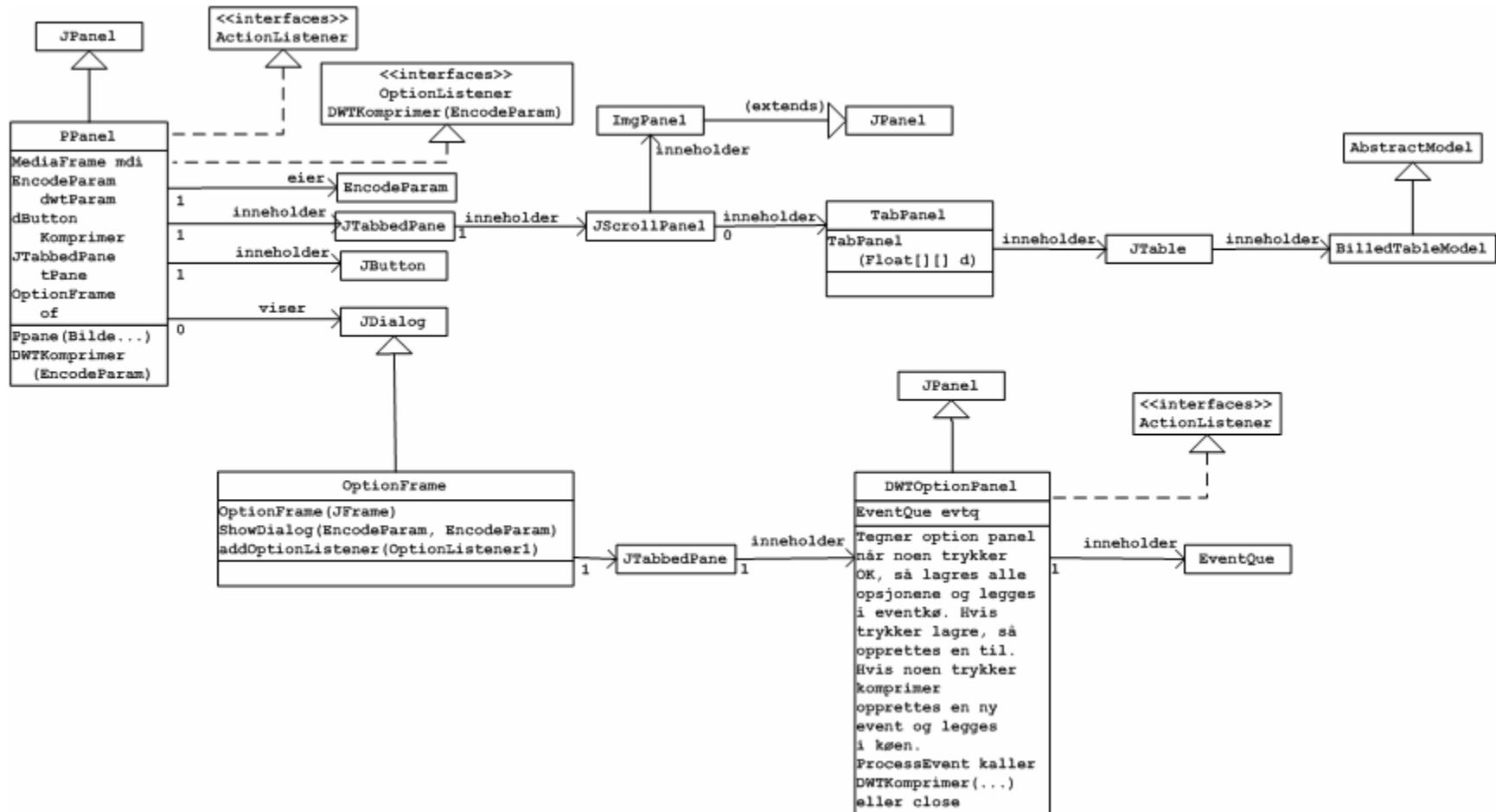


Figure 9.2 Conceptual Classdiagram of the still-image application

Part 3 Java Prototype – MediaCODEC

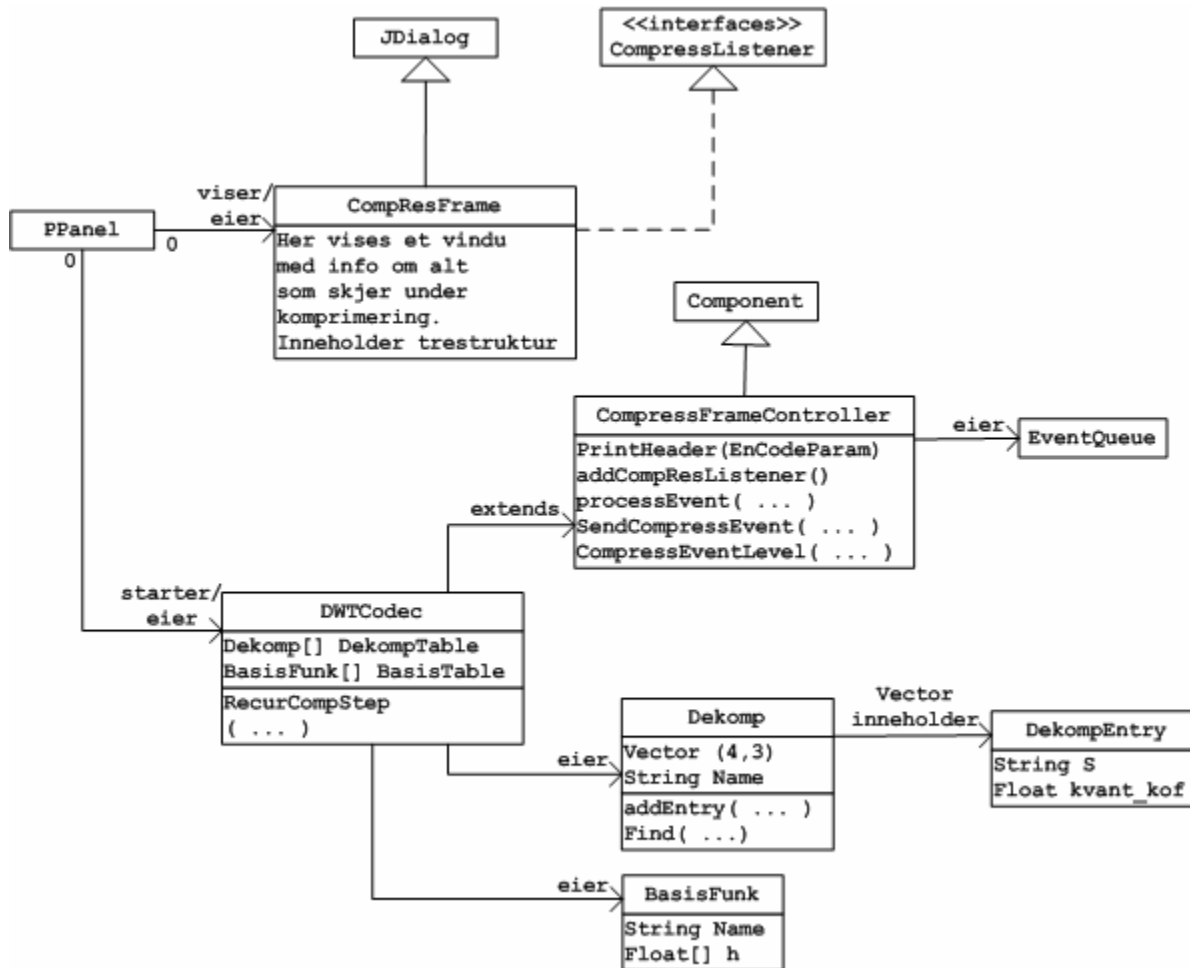


Figure 9.3 Conceptual Classdiagram of the still-image application

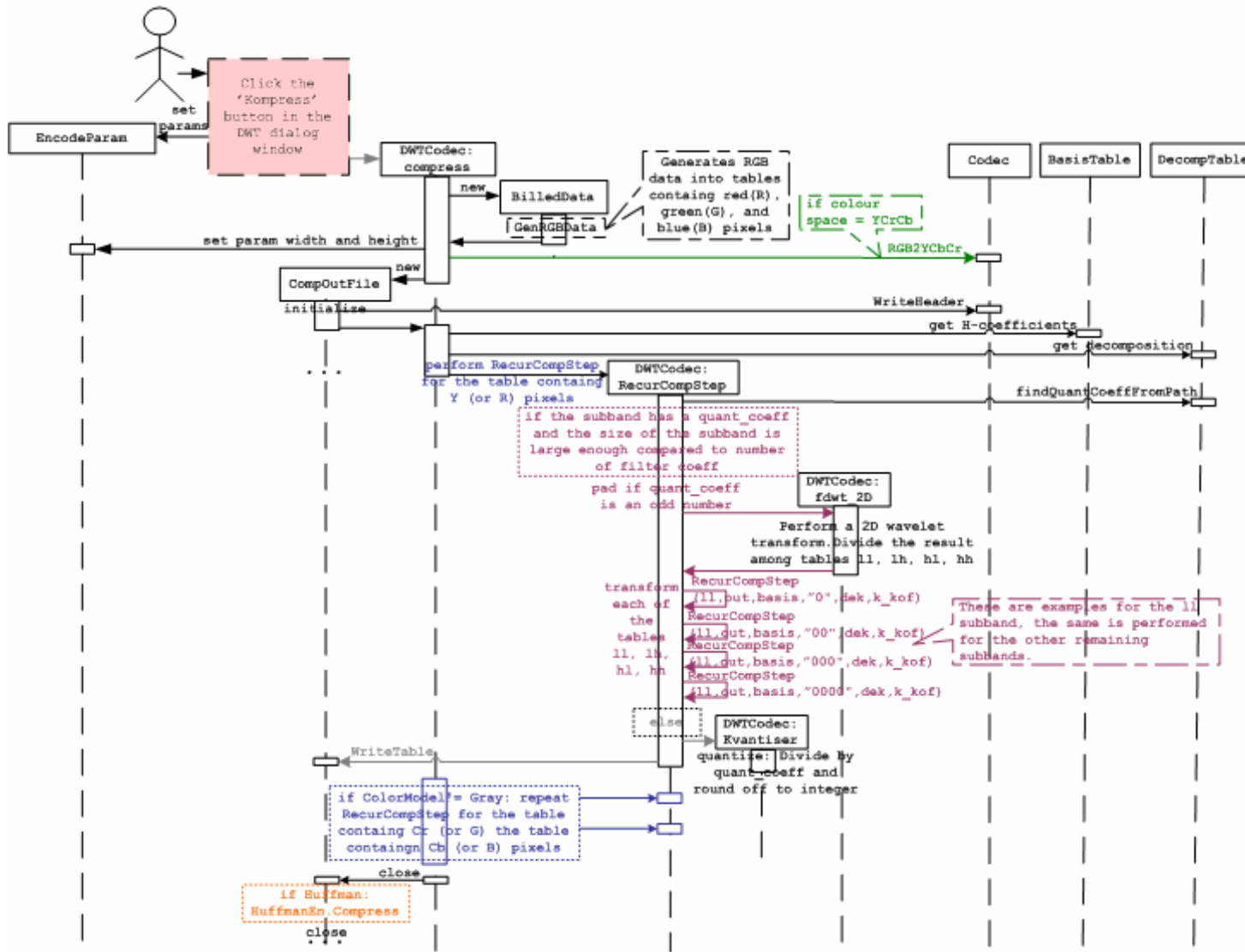


Figure 9.4 Sequence diagram: Compressing an image and writing to file

9.3.2 Transforming and quantizing the image

The transformation is separable and uses a 1-D DWT, first one step for all the rows and then one step for all the columns. The function performing the actual DWT, `DWTCodec.fdw_2d(...)`, is based on the equations <xx> and <yy>, and uses periodic convolution, and padding with the second last value when transforming an odd number of values. The result is divided into four tables, containing the frequency bands LL, HL, LH and HH. The method, `idwt_2d`, which is performing the inverse transform joins these tables and removes any padding.

<sett inn formlene på side 83>

As seen in Figure 9.4 the call to `DWTCodec.fdw_2d(...)` is executed from the `DWTCodec.RecurCompStep(...)` method. The main purpose of `RecurCompStep` is to make sure that the transformation is performed using the selected decomposition. The only decomposition implemented in this application, is 'mallat', seen in Figure 9.5. The decomposition is represented by a class called 'Dekomp', which contains a vector, with a 'DekompEntry' for each subband. The 'DekompEntry' class has two attributes, one containing the subband which is represented, and one relative quantization coefficient. It is the 'Dekomp' class which is used to control the `RecurCompStep`. In addition, this class has a method for controlling how many levels to transform the subbands, and how the subbands should be quantified.

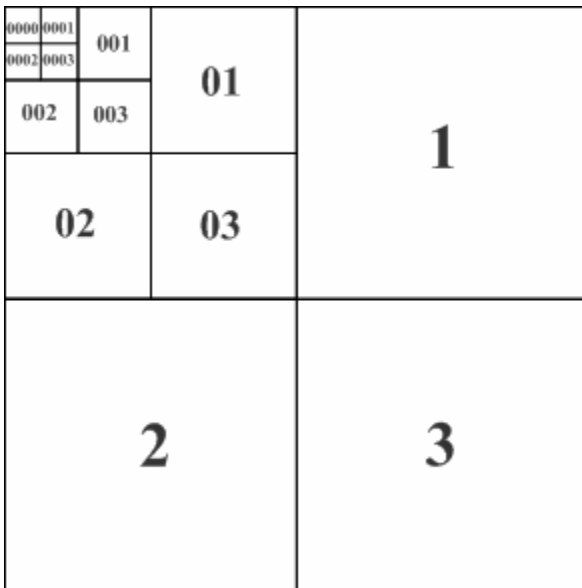


Figure 9.5 Mallat decomposition – from JPEG2000

After the image has been transformed, all the images coefficients are quantified by multiplying each coefficient by a quantizing factor, and then rounded to an integer. The quantizing factors can vary for each colour components, and can be set by the user. When decoding the image, all the coefficients are scaled with the same factors. The quantization and scaling are performed by the `DWTCodec.Kvantiser(...)` and `DWTCodec.Scale(...)`

9.3.3 Entropy encoding

Three types of lossless coding are implemented in this application; Traverse coding using sign bit, RLE 8/16 and Huffman.

After transformation the values are represented by integers (4 bytes), even though most values are small enough to be represented by one byte the largest values are not. In later processes, it is easier to deal with single bytes, so tables with integers are therefore converted to four times as large tables containing bytes. The usual way to do this, is to read one and one integer into four and four bytes, but since most values only uses one byte, a lot of them will have three consecutive zeroes followed by a non-zero value. 3 bytes is not enough to get significant profit from RLE. But in the traverse coding, which was developed by the programmers [11] by studying the data structure during the compression process, structures the values by first reading the least significant byte, then the second least significant, etc. Then at the end of the byte table, we get long sequences with consecutive zeros, only interrupted by values so large that they have to use the most significant bytes. For negative values a sign bit is used, because negative values are coded with twos complement (-1 is coded as FF FF FF FF). Not by using the most significant bit, as most usually, but by using the least significant bit to indicate negativity when sat to 1.

The traverse coding is followed by RLE 8/16 (if it is selected by the user). This implementation uses 8 or 8/16 bit to store the sequences, and it is adjusted to this application, only compressing sequences of consecutive zeroes, because that is almost the only consecutive sequences that exists in our data. Zero is therefore used as escape character, so after a zero, a value telling how many subsequent zeroes there are.

The last step in the encoding process is the Huffman coding. This implementation is quite similar to the one used in JPEG, and we will only briefly explain it. The Huffman coding is generated from the length of the codes. First we generate a frequency table for the data to be coded. Then the code lengths are calculated from the frequency table. The huffman codes are generated from these code lengths, and finally the data is coded with the generated huffman codes.

The traverse and RLE coding are executed in the class 'CompOutFile', if selected by the user. The RLE is implemented by two wrappers, RLEOutputStream and RLEInputStream. The RLEOutputStream are wrapped around a Java OutputStream during the initialize of the CompOutFile, seen in Figure 9.4. The traverse coding are executed in `CompOutFile.writeTable(...)`. The Huffman coding is executed in `CompOutFile.close(...)`, and all the methods regarding this can be found in the class 'HuffmanEn'.

10 Prototype Design

We have chosen a programming language, found a framework, and some quite important procedures for our CODEC, and hopefully this will give us the head start we need. But where shall we go from here? How do we handle audio? How do we structure our file format? How do we estimate motion? Before implementing our prototype we sat down and tried to structure questions, open issues, thoughts and ideas. In this chapter we present some of the plans, design models, and diagrams developed during this stage.

Part 3 Java Prototype – MediaCODEC

We start by looking at the goal of this prototype; what are we supposed to accomplish? In our thesis definition we state that if time permits, we should develop a software video CODEC prototype based on wavelet, and that this prototype should consider advantages and disadvantages of our literature study. The definition gives room for different approaches, so we decide to divide the programming task into several stages, where each stage is a partial goal and joint together they make an adequate video CODEC application. After sketching, modelling, remodelling, and re-remodelling, our design phase results in The Programming Progression Plan. Even though it is our goal, we do not think it is realistic to believe that we can implement this entire plan; we probably should be satisfied if we are able to develop the intra-frame CODEC in step a) to f). The purpose of this list is to make a structured, extendable application, where the different algorithms are easy to improve or exchange. In addition, after each step the functionality of the prototype is enhanced, we can stop programming, having a half-finished application, but we will still have something that works. This is a quite important quality for us since the <priority> of the prototype is "if time permits".

To handle digital video, our prototype has to know how to handle the following:

- Format — how the contents are arranged in a file or network stream.
- Multiplexing — how multiple media streams are put mixed together into a single byte-stream. While it might be convenient to write all the video data to a file and then all the audio, the resulting file could be difficult or impossible to play at a consistent speed, so you "multiplex," or interleave, the streams together, putting the pieces of each stream that represent the same time close to one another.
- Encoding — how the media is encoded and compressed.
- Rendering — how to present the decoded/decompressed data to the screen, speakers, etc.

We use the RGB space as output to the renderer, so JMF can handle the rendering, but we have to specify the MDWT format and implement a multiplexer, demultiplexer (parser), encoder, and decoder that can handle this format.

10.1 A coarse CODEC sketch

We name the CODECs:

- Intra-frame CODEC: MDWT – Motion Discrete Wavelet Transform
- Inter-frame CODEC: MEWT – Motion Estimation Wavelet Transform

Both CODECs will use the same file format, but we give them different file extensions, MDWT will have *.mdw, and MEWT have *.mew. <Er det nødvendig?>

We start by making a course sketch, as seen in

Figure 10.1, to figure out how to handle different type of input and output. If the input is a file with extension *.mdw or *.mew the prototype will decode it with the appropriate decoder and render it to the screen. If the input is of another file type or captured from e.g. a web camera, the user have to decide what compression scheme to use and if it shall be stored to file or rendered to screen. We also want to support rendering files and capture direct to screen using JMF player built-in functionality. This feature will enable the user to play video<media hvis audio kan fungere>files of the file formats supported by JMF, like an ordinary media player, or to show capture from a web camera. We will refine the sketch in the following sections; MDWT (Motion Discrete Wavelet Transform) CODEC and MEWT (Motion Estimation Wavelet Transform) CODEC.

10 Prototype Design

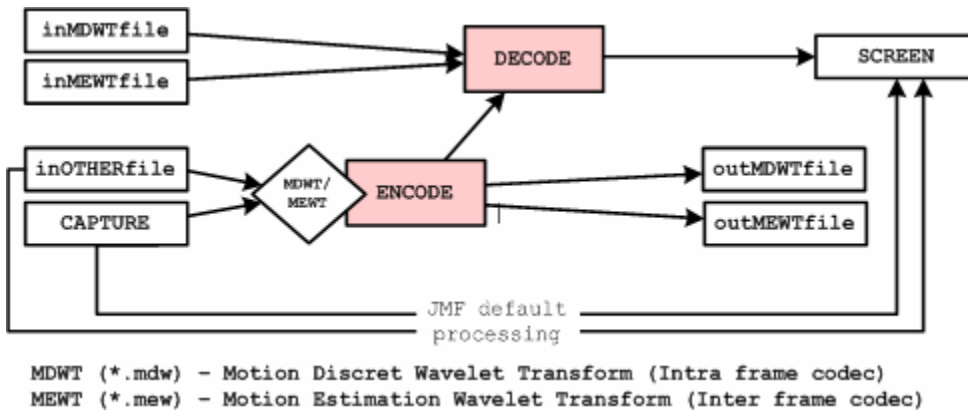


Figure 10.1 Coarse sketch of how to handle different type of input and output

10.2 Layout and User Interaction

After designing some basic functionality for the prototype, we sketch the user interaction for compressing, displaying, or storing digital media. A UML Use Case inspired diagram is seen in Figure 10.2.

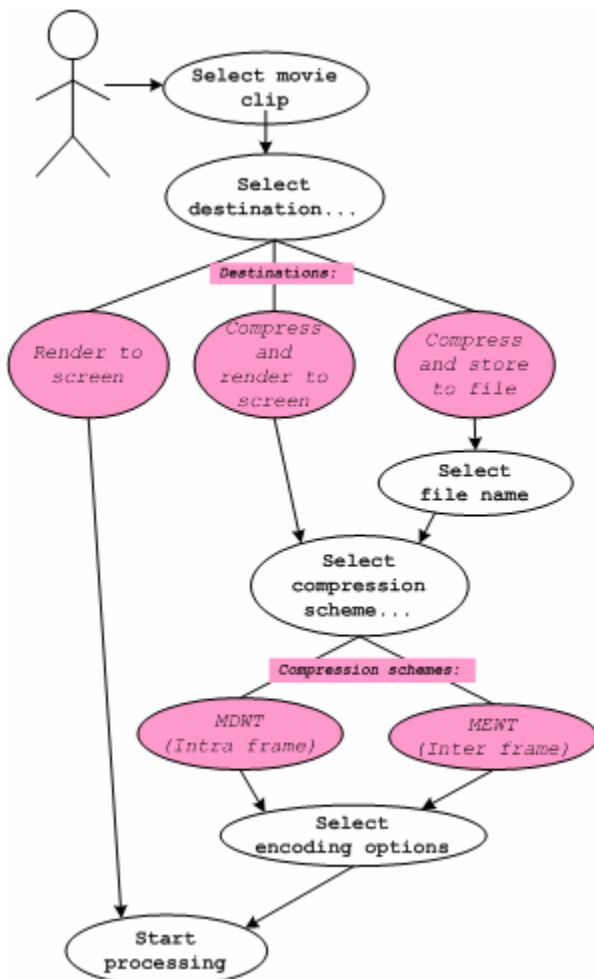


Figure 10.2 User interaction for compressing, displaying, and storing a movie clip

Part 3 Java Prototype – MediaCODEC

To solve the encoding options necessary, we make several layout sketches by using the graphic interface in VB (Visual Basic) editor; the last version is seen in Figure 10.3. We choose to put all options in one modal dialog window, and enable/disable them based on earlier selections.

Table 10.1 gives an overview over what is enabled/disabled and when. For more information about the different options, see section 10.4, 10.5, and the User Guide. All encoding options for the DWT and the entropy encoding are derived from the still image compression application.

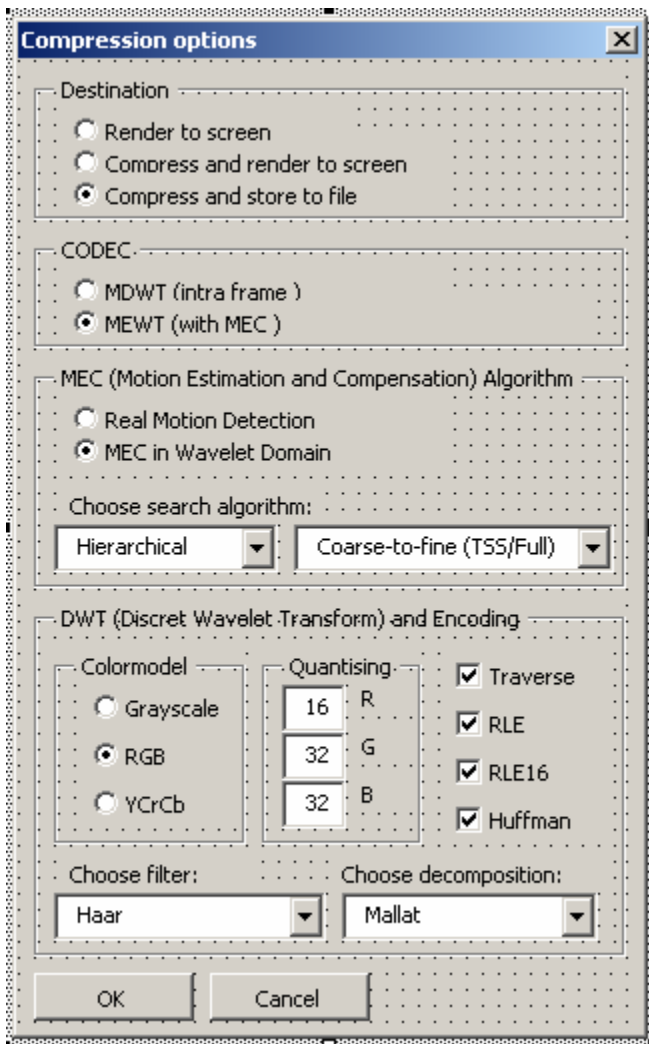


Figure 10.3 Layout sketch for compression options

Table 10.1 Compression options

Category	Action command	Enabled	Enables
	OK/Cancel	Always	
Destination, default	Render to screen	Always	All other options are disabled
Destination	Compress and render to screen	Always	CODEC and DWT options are always enabled
Destination	Compress and store to file	Always	CODEC and DWT options are always enabled
CODEC	MDWT (intra-frame)	Depending on destination	MEC options are

10 Prototype Design

			disabled
CODEC, default	MEWT (with MEC)	Depending on destination	MEC options are enabled
MEC algorithm	Real Motion Detection	Depending on CODEC	
MEC algorithm, default	MEC in Wavelet Domain	Depending on CODEC	
MEC algorithm, default search algorithm depends on chosen MEC algorithm	Choose search algorithm	Depending on CODEC. Alternatives in the first combo box depend on the selected MEC algorithm.	Alternatives in second combo box depends on selection in the first combo box
DWT and Encoding, Colormodel, default RGB	Grayscale, RGB, YCrCb	Depending on destination	Quantising textfields and labels
DWT and Encoding, Quantising, Default: 16R, 32G, 32B		Depending on destination and colormodel	
DWT and Encoding, Encoding options, all are default checked	Traverse, RLE/RLE16 and Huffman	Depending on destination	We do not support optional choices for Huffman. Huffman is always checked and disabled.

The compression option dialog window is the only layout design we do for our prototype. To display and play the video, we decide to use standard JMF control panel. For the superior GUI-framework we want to use a MDI <(multiple...)> frame, which can contain several child frames and a standard menu (we develop the MDIframe derived from [11]), and we choose to use the systems look and feel. Look and feel is how Java defines the appearance of an application, like if you run a Windows OS (operating system) your application will look like a standard Windows applications, and if you run on Sun Solaris you application will look like a Sun Solaris application.

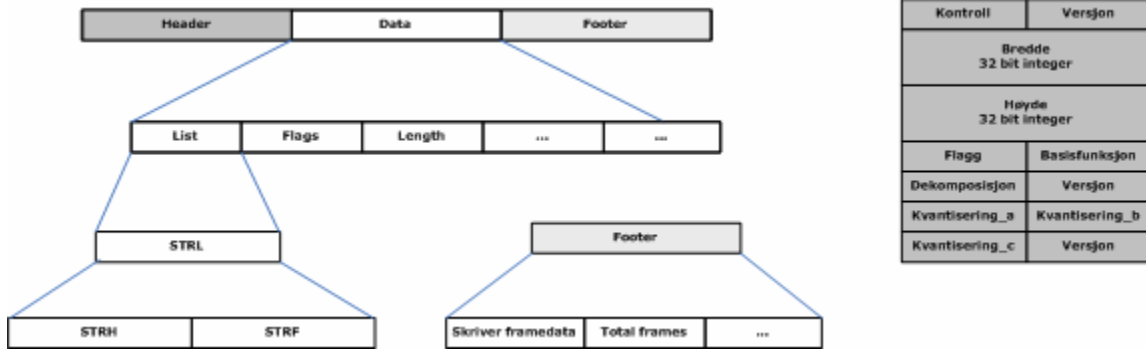
Table 10.2 contains the menu items:

Table 10.2 Menu Items

File Menu		View Menu		Window Menu	
Open Image	<i>Derived from [11]</i>	Image Options	<i>Derived from [11]</i>	Arrange All	<i>Derived from [11]</i>
Open Video	<i>Select a video file. The compression option dialog is displayed.</i>	Video Options	<i>Select some options regarding the video compression, e.g to show a debug/message window</i>	Split	<i>Derived from [11]</i>
Capture	<i>Capture video from a web camera (and audio from microphone). The compression option dialog is displayed</i>				
Exit	<i>Derived from [11]</i>				

10.3 File Format

We wanted to make the file format as easy as possible. We studied the QUICKTIME [13] and AVI [14] to find out how it can be done.



10.4 MDWT (Motion Discrete Wavelet Transform) CODEC

The MDWT CODEC codes a video sequence as a series of DWT⁵ images, each corresponding to one frame of video (i.e. a series of intra-coded frames). No attempt is made to exploit the inherent temporal redundancy in the moving video sequence and so the compression performance will be poor compared with inter-frame CODECs. The reason why we design and implement this CODEC is to get the framework up and running. All the parts (file opening, multiplexer, encoder, decoder, file writer, etc.) implemented in this stages, are with some adjustments reusable in an inter-frame CODEC. In Figure 10.4 we give a coarse overview over the processes in the MDWT CODEC.

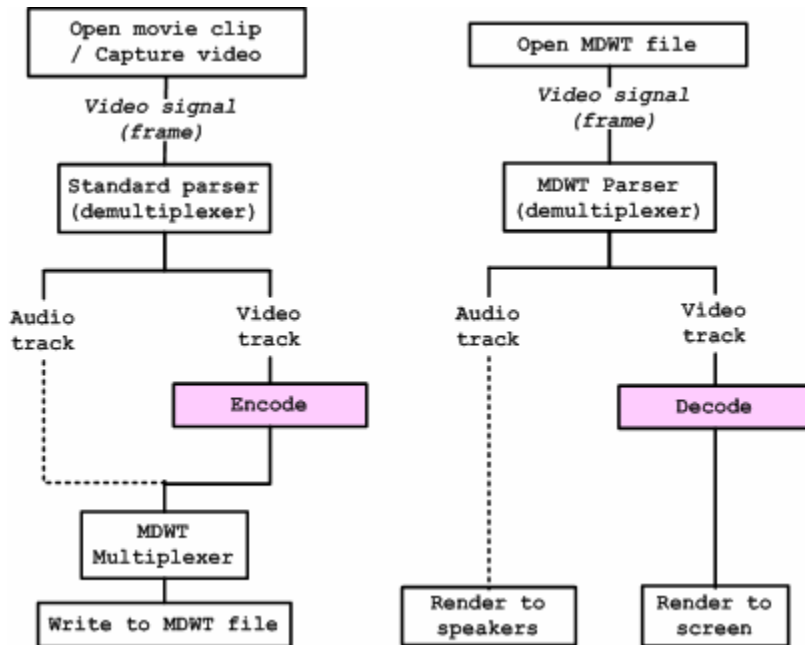


Figure 10.4 Coarse overview of the processes in the MDWT CODEC

Figure 10.5 gives a bit more details of what happens in the encoder/decoder. All the compression algorithms used in this figure are the same as developed in [11]. We plan to separate the “pieces” of the still image application, and set them together in our architecture, using the JMF 2.1.1.

⁵ DWT is the fileformat used in [11]

10 Prototype Design

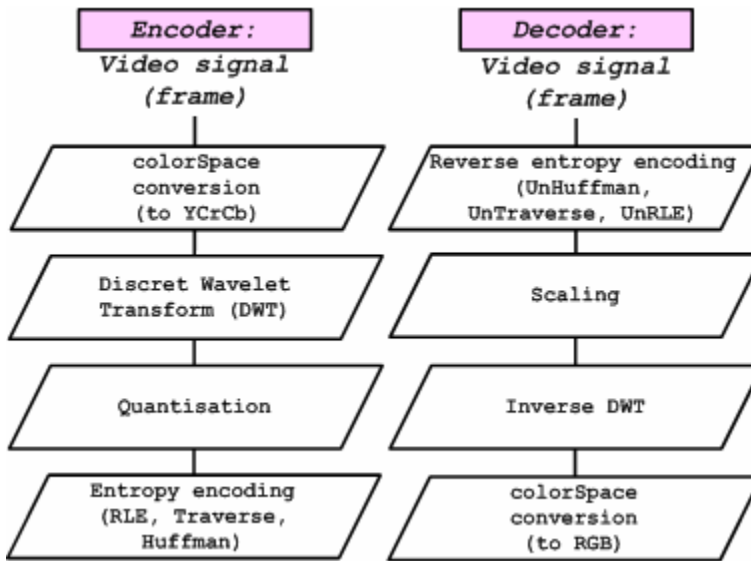


Figure 10.5 Overview of the intra-frame encoder and decoder

Part 3 Java Prototype – MediaCODEC

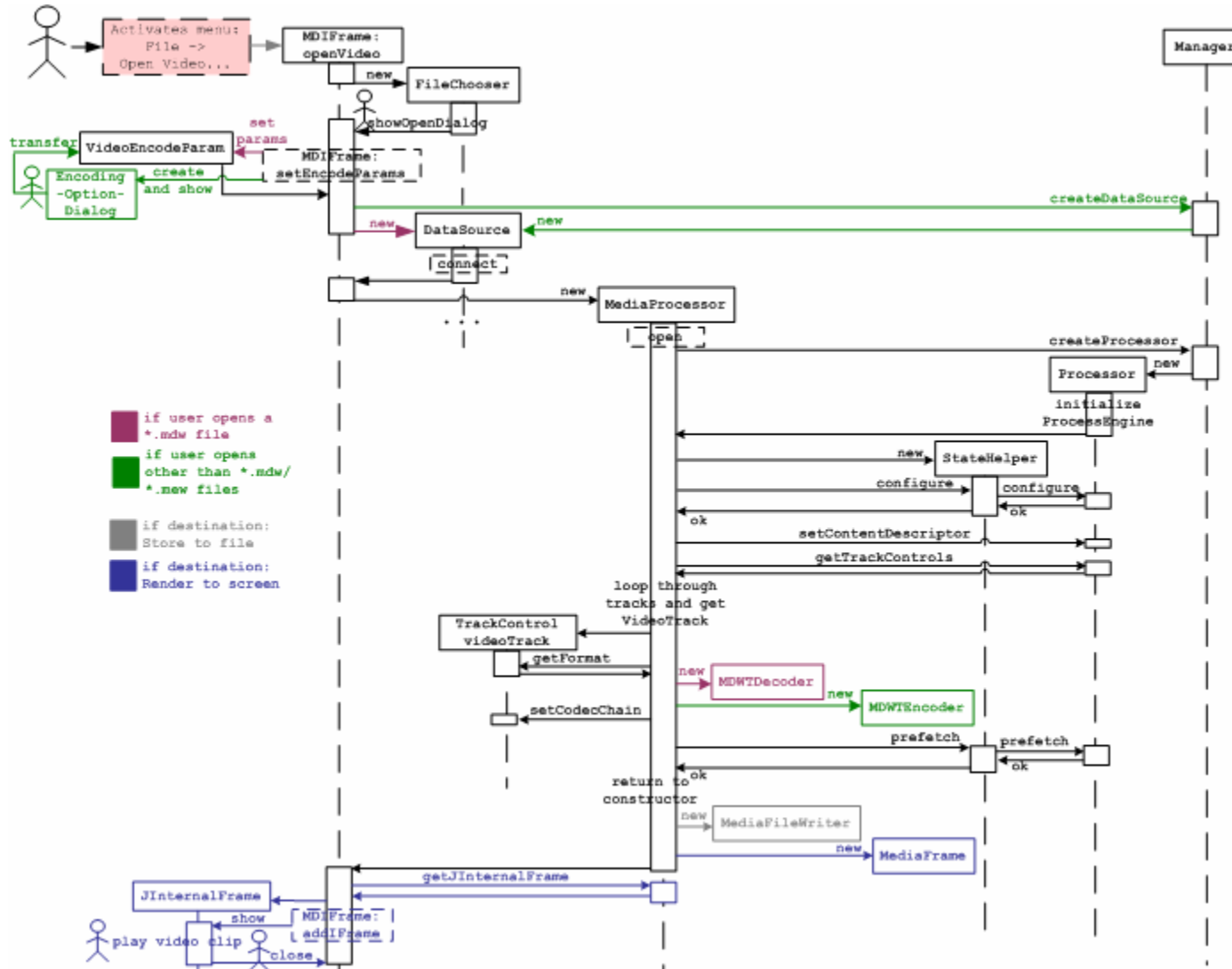


Figure 10.6 Sequence diagram: Preparing the Processor for playback

10.5 MEWT (Motion Estimation Wavelet Transform) CODEC

The MEWT CODEC codes a video sequence as a series of DWT images, each corresponding to one frame of video (i.e. a series of intra-coded frames). No attempt is made to exploit the inherent temporal redundancy in the moving video sequence and so the compression performance will be poor compared with inter-frame CODECs. The reason why we design and implement this CODEC is to get the framework up and running. All the parts (file opening, multiplexer, encoder, decoder, file writer, etc.) implemented in this stages, are with some adjustments reusable in an inter-frame CODEC.

10.6 The Programming Progression Plan

We want to develop our prototype step-by-step. Each stage adds some functionality to the prototype and joint together all steps make an adequate video CODEC. The purpose of this plan is to make a structured and extendable prototype.

- a) Implement a UI (User Interface) framework based on JMF, including methods for opening and playing movie clips
- b) Extend this framework and get access to each frame in the playing movie clip
- c) Implement the intra-frame encoder and decoder, based on the compression algorithms from [11]. The result should be rendered directly to screen, so we do not have to handle demultiplexing, multiplexing, file writing, and file format
- d) Implement a file format, and handle storing to a file
- e) Implement handling for demultiplexing and multiplexing
- f) Implement UI for selecting compression options. Clean up the code and get rid of all hard coded variables/constants

By this stage we hope to have a working intra-frame CODEC. In the next steps we want to improve the prototype by implementing an inter-frame CODEC. The main difference between an intra- and inter-frame CODEC is that in an inter-frame CODEC we do temporal redundant removal. In the first attempt to remove temporal redundant data we implement the simplest form of motion estimation.

- g) Implement the reusable parts of the intra-frame CODEC for the inter-frame CODEC (we want to keep the intra-frame CODEC).
- h) Extend the encoder; Implement an algorithm to find a residual frame by detecting real motion, for now we use the compression algorithms from [11] to encode the residual frame
- i) Extend the decoder; Implement an algorithm to calculate the current frame from a reference frame and a residual frame

We should have a working inter-frame CODEC, but the MEC algorithm is in its simplest form. Now is the time to start working on a real MEC algorithm, and take into consideration advantages and disadvantages of our literature study.

- j) Improve the motion detection algorithm by <using the original frame as the reference frame every x^{th} frame to reduce problem with drifting>
- k) Implement a loop to choose whether to use intra- or inter-coded frame from a calculated SNR (Signal-to-Noise-ratio) value

Part 3 Java Prototype – MediaCODEC

- l) Implement a real Motion Estimation and Compensation (MEC) algorithm. This MEC algorithm should take place after the DWT has been performed (so-called in-band prediction or MEC in the wavelet domain)

When this work is done, we can try to improve the transform and encoding parts of the CODEC according to the literature study.

- m) Implement biorthogonal wavelet filters to use for DWT
- n) Implement static Huffman tables, instead of generating them during encoding
- o) Implement an improved DWT based on SPITH or SPECK
- p) Improve the entropy encoding; traverse, RLE and Huffman. <Motion vectors, residual frame and reference (intra) frame> should be encoded differently.

11 Implementation and Development

The current implementation contains a framework with support for opening and playing video sequences of those content types supported by the JMF API, including quicktime (*.mov) and msvideo (*.avi). Capturing video clips from a web camera is also supported, the captured sequence can either be rendered direct to screen, or stored to a 'quicktime' or 'avi' file. Our implementation does not currently support encoding the captured video directly with our encoder; it has to be intermediately stored to a file. We have implemented a so-called intra-frame CODEC (no attempt is done to remove temporal redundancy), the MDWT (Motion DWT) CODEC. It includes a MDWT encoder and decoder; this is where the DWT compression algorithms have been exploited. In addition, we have implemented a MDWT file format and a multiplexer/demultiplexer to handle writing and reading this format. Initially we implemented support for sound tracks in this multiplexer/demultiplexer, but this was removed in order to more correctly estimate the real compression performance of our CODEC.

Experimental testing has showed that our MDWT CODEC can yield a compression rate of 13:1 in a small resolution video clip, with good visual quality on single frames. The major problem of our CODEC is as initially concerned, the computationally complexity. Small-resolution video becomes very jerky, while larger-resolution video clips are considerably worse, maybe able to show one frames pr second. This is due to the combination of the computationally complexity of the implemented DWT and the CODEC, the 'slow nature' of the Java programming language, and most-likely and inefficient method used to draw and represent the frames.

Figure 10.6 is a sequence diagram, showing the most important classes, methods and messages when a user activates the menu item: File -> Open Video... The Menu item File -> Capture... is almost the same, the main difference is that instead of creating a FileChooser and selecting a file, the user must select capture device and some capture options. The sequence diagram mainly shows how the processor is being prepared for playback. (The processor has to reach a state (be prefetched and realized) before it can be played). Sequences during playback, file writing and building the GUI-frame is not included in this diagram.

So what happens after the processor is ready and the video clip can be played? After playback of the media is started (e.g. by a user clicking the play button), the following takes place (conceptual description, se også Figure 10.4 and <i JMF section>):

- The DataSource reads a chunk of data (a frame) from a media source
- The MediaProcessor separates the tracks and get information from the data by using an appropriate Multiplexer plug-in (the Multiplexer has been selected by a handler (TrackControl), using a ContentDescriptor). The Multiplexer receives a frame-worth of data and separates the track, frame by frame.
- The frame (of the videoTrack) is decoded/encoded using the appropriate CODEC (the CODEC-plug-in was set in setCodecChain)

The videoTrack is rendered using a RENDER plug-in, selected by a handler, using the output format of the CODEC, or if the user has chosen to store to a file, the data is demultiplexed and written to file using a DataSink (chosen by the Manager using the DataSource (a Processor can be a DataSource...)).

12 User Guide

In this part we will give a brief introduction on how to use the MediaCODEC application.

When you start the program, you should see an empty MDI () window with a menu containing File, View, and Window, as in

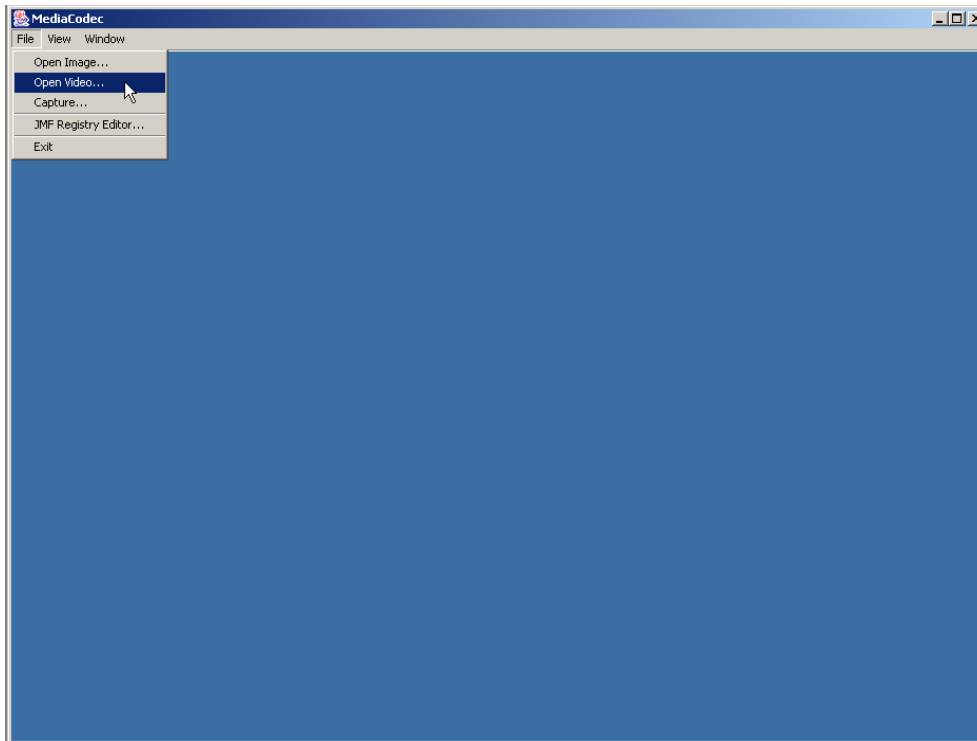


Figure 12.1 MDI window with menu

12.1 The File Menu

Open Image...

Displays a selected image and allow you to compress it with the algorithms developed in [11]. This is the original image compression application presented in section 9.3, we have not done any changes that affect this compression process (and that also explain why most of the prompts for this

Part 3 Java Prototype – MediaCODEC

menu item are in Norwegian). If you like information beyond what we have provided, take a look at [11].

When you choose “Open Image” a standard “open file” dialog box is displayed.

- Select the image you want to compress and click Open. Now you should see the selected image.
- Click the “Compress” button in the toolbar below the image, marked with red in Figure 12.2. A dialog window is displayed.
- Choose your compression scheme, and mark your options
- If you like to compress the image right away click the “Komprimer” button, marked with green in Figure 12.2. Otherwise you can click Ok, and your options will be saved for later.
- When the “Komprimer” button in the dialog window is clicked the compression process is carried out, and finally the compressed image and an information window is displayed.

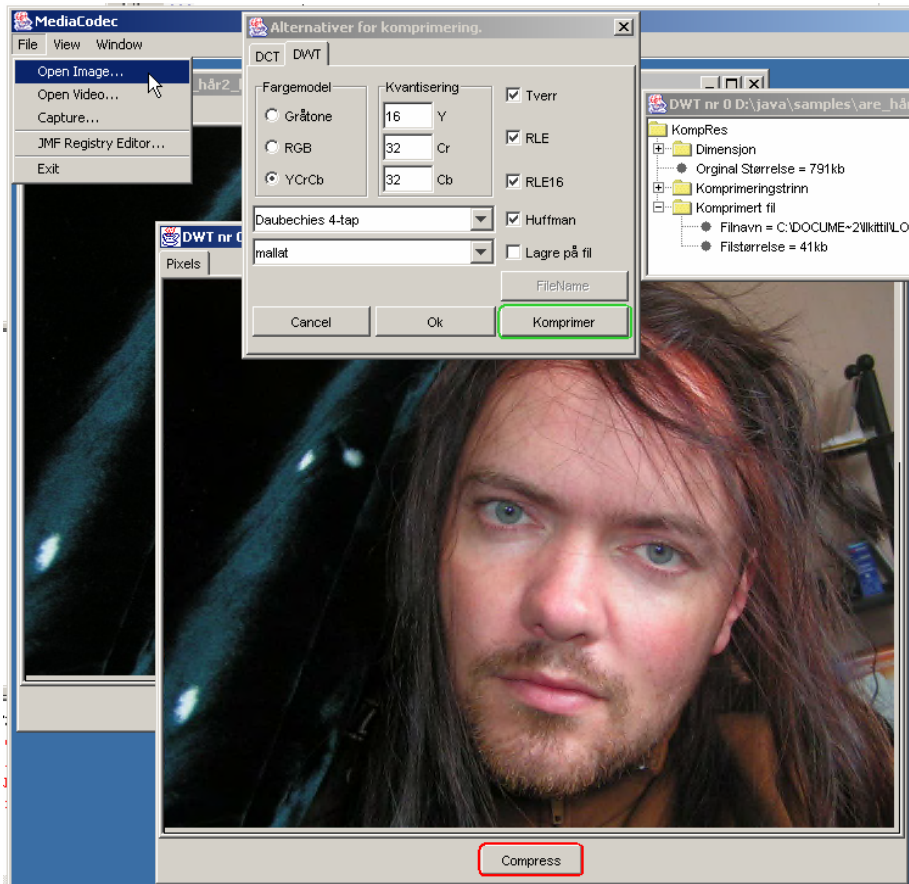


Figure 12.2 Menu item: File->Open Image...

Open Video...

Displays or stores a selected video clip. See section 12.2 The Open Video Menu Item for further information.

Capture...

Capturing video clips from a web camera is also supported, the captured sequence can either be rendered direct to screen, or stored to a 'quicktime' or 'avi' file (note that the MDWT choice in the dialog box can not be used). Our implementation does not currently support encoding the captured video directly with our encoder; it has to be intermediately stored to a file. Detecting capture device can be done as shown in Figure 12.3.

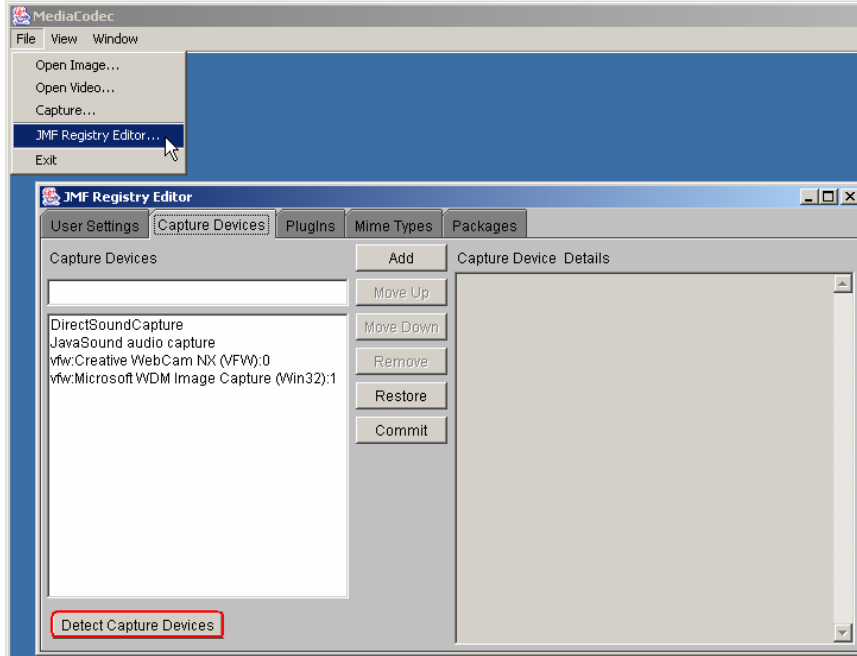


Figure 12.3 JMF Registry Editor: Detect Capture Devices

JMF Registry Editor...

Opens the JMF Registry Editor, which is an application developed by Sun⁶ which can be used to configure JMF. We included the JMF Registry Editor in our application for convenience during implementation and testing. If you want a user guide on JMF Registry Editor or download the source code, see [12].

Exit

Closes the MDI window and all its containing frames

12.2 The Open Video Menu Item

When you choose "Open Video" a standard "open file" dialog box is displayed, as seen in Figure 12.4. In the "Files of type:" drop down list you can see the file types supported by our prototype⁷. The file type MotionDWT, with the extension ".mdw", is the format which uses the compression scheme developed in this prototype.

⁶ This menu item request that jmaps library are in the classpath, it was included in the JMF version we installed, jmf-2_1_1e-windows-i586.exe. (It was a part of jmf.jar)

⁷ The file types supported are the same as supported by JMF, and all restrictions regarding file type support in JMF also applies to our prototype.

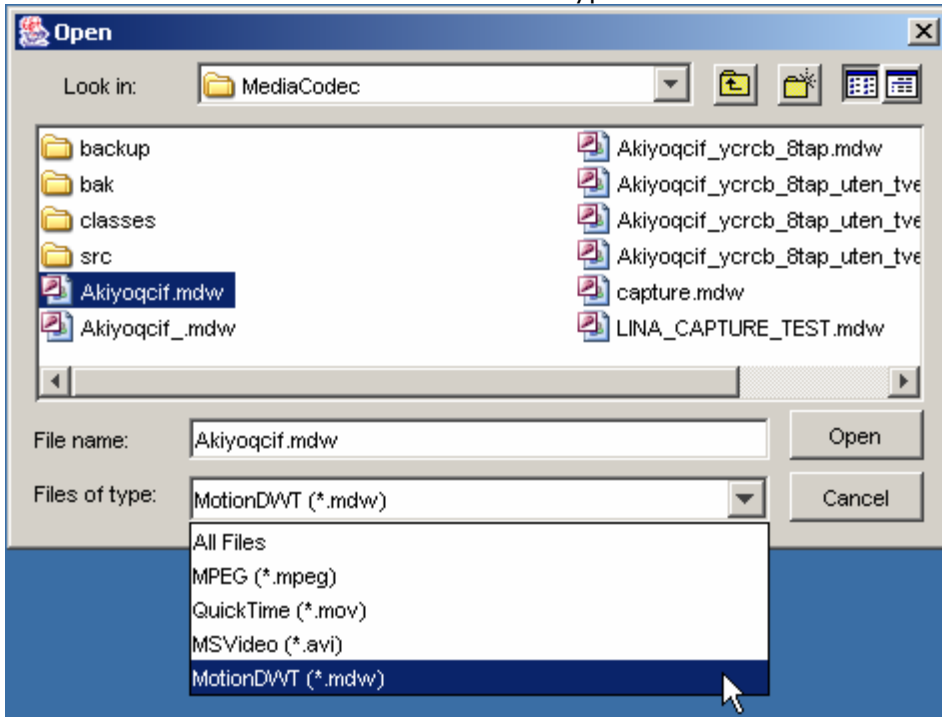


Figure 12.4 Open file dialog


- Select the video clip you want to process and click Open.
- If you selected a file with the extension “*.mdw”, you should see a window containing the video clip, as in Figure 12.5. Click the  button to start the video clip.



Figure 12.5 Video panel with presentation controls

- If you selected another video file type, a dialog window, like Figure 12.6, is displayed.

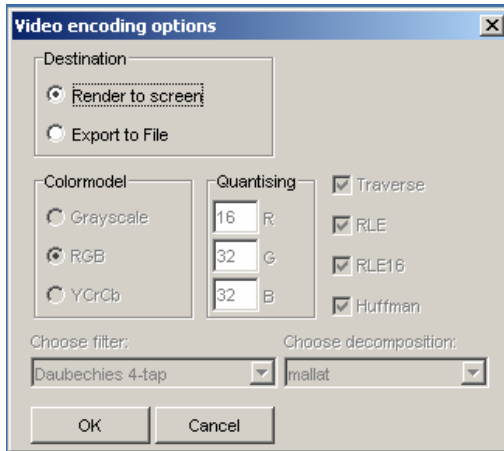


Figure 12.6 Video encoding options dialog

- If you select "Render to screen" as the destination, the file is played using the compression scheme specified in the format. <If the file is raw>
- If you select "Export to File" as the destination, you are now able to set options which will be used during compression.
 - Colour model
 - Quantizing
 - Filter
 - Decomposition
 - Traverse
 - RLE/RLE16
- When you have marked all your options, click OK to start the process
- Be prepared, this can take a while, a message box appears when the process is finished.

12.3 The View Menu

Image Options...

Display options for the compressed still image, belongs to the Image Compression program. This menu item has no effect on video clips.

12.4 The Window Menu

Arrange All

Arranges all child-windows, which are not minimized, from top to bottom inside the MDIFrame

Split

Splits the space inside the MDIFrame evenly among all open (not minimized) child-windows

Part 4 Conclusion

13 Experimental Testing and Comparisons

13.1 Testing of Image- and Video CODEC algorithms in VcDemo

Here we will show a visual (subjective) and numerical (objective) comparison of SPIHT, EZW, JPEG2000 and Baseline JPEG, all with a resolution level of 0.5 bits per pixel (bpp), or a compression ratio of 48:1. SPIHT, EZW and JPEG2000 are all compressed with 6 decomposition levels.



Figure 13.1 Original (256x256)



Figure 13.3 EZW



Figure 13.2 SPIHT



Figure 13.4 JPEG2000



Figure 13.5 JPEG

We used a software image compression learning tool called VcDemo, developed by the Information and Communication Theory Group at Delft University of Technology, and the software including test images can be downloaded from [1]. This program is not been optimized for ultimately comparing of PSNR, visual, or encoding/decoding time performance, but it gives a good indication of how different compression aspects affect an image. This experimentation is therefore intended to illustrate the visual and numerical similarity between the image coders mentioned. We will present the values we got to show how little the PSNR values differ between each image coder, and given that this is not an optimized comparison test, we have only compressed the Lena image, and the results we got are shown in Figure 13.2.

Table 13.1 PSNR Performance at 0.5 bpp and 48:1 compression ratio

Algorithm	PSNR (dB) Lena (256x256)
SPIHT	32.7
JPEG2000	32.3
EZW	30.5
JPEG	29.4

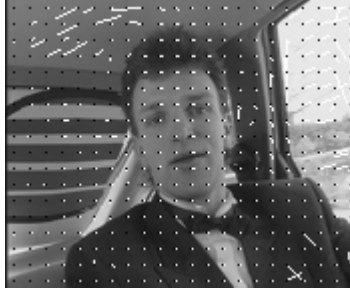
This shows that JPEG2000 and SPIHT are very similar in that of visual quality and PSNR values, and as expected, the visual quality of EZW is hardly noticeable, but slightly inferior compared to SPIHT at this bit rate. JPEG performs the worst, but even at 0.5 (bpp) it still shows quite good visual quality. Usually JPEG2000 performs slightly better than SPIHT, and the reason for why it does not this time is probably due to the software implementation of the algorithm, and we believe that not all of the quality enhancing mechanisms in the JPEG2000 standard is implemented here.

We have also used VcDemo to provide an illustration of a motion estimation of a video clip. Here we show one frame of the "Carphone" video clip.

a) Current frame



b) Motion Compensated Prediction



c) Frame Difference



d) Motion Compensated Frame Difference

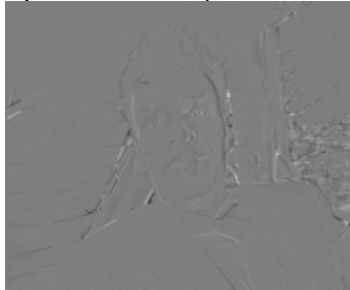


Figure a) is the current frame.

Figure b) is a motion compensated prediction frame, which is how the encoder thinks the next frame will look like.

Figure c) is the difference between the current and the previous frame, and we clearly see that this frame has more energy or image data in it than figure d)

Figure d) is the difference between a motion compensated frame and the current frame, and the better the encoder estimates the next frame, the less energy will the motion compensated frame difference consist of. This frame can be transferred instead of the original frame, or the frame difference to save bandwidth and transmission time.

13.2 Comparison of Phase-Correlated and Block-based Motion Estimation

Phase-correlation ME is very computationally efficient and it produces much smoother motion field with low entropy than the BM method does. Phase-correlation works better than the block-based method in the case of large scale translational motion, while BM is more suitable for predicting regular and small scale motion and multiple-object movement.

It is also found that the block size greatly affects the ME performance. The blocks should be large enough to group pixels with similar motion, but should be small enough to separate pixels with different motion and multiple-object movement. [42,71]

14 Conclusion

Even though wavelet-based video compression has been an area of research for more than a decade, it has been considered complex and inefficient until recently.

During our thorough investigation of existing research work we have found that by using a lifting-based three-dimensional wavelet transform with motion compensated temporal filtering; effective and highly scalable coding can be achieved, while additionally minimizing the well-known problem with visually disturbing ghosting artifacts in the low-pass band. Further, when combining this technique with a complex motion model, like for example a deformable triangle mesh, improved PSNR performance and visual quality can be gained.

If performing motion compensated temporal filtering in the wavelet domain, by using a shift-invariant wavelet transform, it additionally permits the independent temporal filtering of each resolution of the input signal. This enables many potential developments for multi-resolution decoding and can be a viable approach for fully scalable video compression.

It has also been found, that coding efficiency of both spatial-domain and wavelet-band motion compensated temporal filtering can be improved by using an optimized multihypothesis motion estimation algorithm.

Further combination with other type of optimised algorithms, like improved error resilience, can be done to improve an even better system, but it is important to consider the trade-off between computational complexity and compression performance.

Experimental testing in several independent research work has shown that systems employing shift invariant MCTF for many sequences is comparable or superior in terms of mean PSNR to the highly-optimized DCT-based MPEG-4 AVC, over a large range of bit-rates. While at the same time, offer the advantage of fully-embedded coding.

It is difficult to give a single conclusion whether wavelet-transforms are ready to replace the DCT in video CODECs, but what we can say is that from our point of view, it seems that the fundamental problem with wavelet-based temporal removal while obtaining the properties of multiresolution structure, has been solved, and that wavelet-based video CODECs now can compete with DCT-based video CODERS.

Since scalable representations are important for efficient utilization of limited channel capacity, wavelet-based video CODERS has applications in many areas including simulcast, videoconferencing and remote video browsing.

14.1 Further Development of the Prototype

The CODEC is implemented in its most primitive form, and has a lot of potential for further development.

Foremost we would strongly emphasize to improve the efficiency of the video CODEC, there is several ways to attack this problem; the most interesting seen from a wavelet-perspective is to improve the implementation of the DWT transform using a lifting based scheme, but a faster and even more effective computational saving

14 Conclusion

could be gained by changing the Huffman coding algorithm to uses pre-generated statistic tables. We would also strongly suggest implementing the actual CODEC classes in C++ and using native interfaces, like the SUN's JMF API usually does for its incorporated VIDEO CODECs. When gaining an acceptable computationally efficiency, we would suggest to continue to develop the proposed programming plan in section 10.6 in order to improve the compression efficiency.

References

- [1] B. Bhatt, D. Birks and D. Hermreck, "About digital video compression," IEEE Spectrum Magazine, vol. 34, no. 10, pp. 22-23, October 1997.
- [2] <http://www.qualcomm.com>
- [3] <http://www.nttdocomo.com/>
- [4] M. Budagavi, W. R. Heinzelman, J. Webb and R. Talluri, "WirelessMPEG-4 Video Communication on DSP Chips," IEEE Signal Processing Magazine, vol.17, no. 1, pp. 36-53, January 2000.
- [5] Compression for Great Digital Video – Ben Waggoner, August 2002
- [6] Video Codec Design, Developing Image and Video Compression Systems – Ian E. G. Richardson, April 2002
- [7] Video compression demystified – Peter Symes, December 2000
- [8] Elements of wavelets for engineers and scientists, Dwight F. Mix, 2003
- [9] Z. Xiong, K. Ramchandran, M. T. Orchard and Y.-Q. Zhang, "A Comparative Study of DCT- and Wavelet-Based Image Coding," IEEE Trans. Circuits Syst. Video Technol., vol. 9, pp. 692-695, Aug. 1999
- [12] D. A. Huffman, "A Method for the Construction of Minimum Redundancy Codes," Proceedings of the IRE, Vol. 40, pp. 1098-1101, 1952
- [13] I. H. Witten, R. M. Neal & J. G. Cleary, "Arithmetic Coding for Data Compression," Comm. ACM 30 (June 1987), 520-540.
- [14] Wavelet Image and Video Compression, Pankaj N. Topiwala, Kluwer Academic Publishers
- [15] <http://fag.grm.hia.no/fagstoff/perhh/htm/fag/matem/datwww/wavelet.htm>
- [16] Howard L. Resnikoff, Raymond O. Wells, Jr., "Wavelet Analysis, The Scalable Structure of Information", Springer-Verlag New York, Inc., 1998.
- [17] Stéphane Jaffard, Yves Meyer, Robert D. Ryan, "Wavelets, Tools for Science & Technology", Society for Industrial and Applied Mathematics (SIAM), 2001.
- [18] <http://www.beyonddiscovery.org/content/view.article.asp?a=1952>
- [19] <http://perso.wanadoo.fr/polyvalens/clemens/wavelets/wavelets.html>
- [20] <http://www-sigproc.eng.cam.ac.uk/~ngk> (Complex Wavelet Design Package: <http://www-sigproc.eng.cam.ac.uk/publications/ngk/qshiftgen.zip>)
- [21] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," IEEE Transactions on Image Processing, vol. 1, no. 2, pp. 205-220, April 1992.
- [22] N G Kingsbury:
"Image Processing with Complex Wavelets",
Phil. Trans. Royal Society London A, September 1999, on a Discussion Meeting on
"Wavelets: the key to intermittent information?", London, February 24-25, 1999.
- [23] M. D. Adams, "The JPEG-2000 still image compression standard—
Tech. Rep. distributed with the Jasper JPEG-2000 software," [Online]. Available:
<http://www.ece.ubc.ca/~mdadams/jasper>, Tech.Rep. N2412, ISO/IEC
JTC1/SC29/WG1, Sept. 2001.
- [24] D. S. Taubman and M. W. Marcellin, JPEG2000: Image Compression
Fundamentals, Standards and Practice. Boston: Kluwer Academic Publishers, 2002.
- [25] Y. Andreopoulos, A. Munteanu, G. Van der Auwera, J. Cornelis and P.
Schelkens, "Complete-to-overcomplete discrete wavelet transforms: theory and
applications," IEEE Trans. on Signal Processing, to appear. 2004
- [26] Y. Andreopoulos, A. Munteanu, J. Barbarien, M. Van der Schaar, J. Cornelis and
P. Schelkens, "In-band motion compensated temporal filtering," Signal Processing:
Image Communication (special issue on "Interframe Wavelet Video Coding"), to
appear

- [27] L. Luo, F. Wu, S. Li, Z. Zhuang, "Layer-correlated Motion Estimation and Motion Vector Coding for the 3D-Wavelet Video Coding", IEEE, pp. 791-794, 2003
- [28] Shapiro, J. M. Embedded Image Coding Using Zerotrees of Wavelet Coefficients, IEEE Trans. SP, vol. 41, no. 12, Dec. 1993, pp. 3445-3462.
- [29] A. Islam and W. Pearlman. "An embedded and efficient low-complexity hierarchical image coder," in Visual Communications and Image Processing, K. Aizawa, R. Stevenson, and Y. Zhang, Eds., San Jose, CA, Jan. 1999, Proc. SPIE 3653, pp. 294-305.
- [30] W. Pearlman, A. Islam, N. Nagaraj, and A. Said. "Efficient, low-complexity image coding with a set-partitioning embedded block coder," IEEE Trans. on Circuits and Systems for Video Technology, 2003.
- [31] J. E. Fowler, Embedded Wavelet-Based Image Compression: State of the Art, Information Technology 45, Oldenburg Verlag, 2003
- [32] Cohen, A., Daubechies, I., Feauveau, J., "Bi-orthogonal bases of compactly supported wavelets"; Comm. Pure Appl. Math., 45 (1992), 485-560.
- [33] W. Sweldens, "The lifting scheme: a custom-design construction of biorthogonal wavelets," Applied and Computational Harmonic Analysis, vol. 3, no. 2, pp. 186-200, April 1996.
- [34] Daubechies, I. and W. Sweldens. FACTORING WAVELET TRANSFORMS INTO LIFTING STEPS. J. Fourier Anal. Appl., Vol. 4, Nr. 3, 1998, preprint.
- [35] B.-J. Kim and W. A. Pearlman. "An embedded wavelet video coder using three-dimensional set partitioning in hierarchical trees (SPIHT)," in Proc. IEEE DCC'97, 1997, pp. 251-260.
- [36] R. Kutil and Andreas Uhl. "Hardware and Software Aspects for 3-D Wavelet Decomposition on Shared Memory MIMD Computers," in Proceedings of ACPC'99, volume 1557 of Lecture Notes on Computer Science, pp. 347-356, Springer-Verlag, 1999.
- [37] C.I. Podilchuk, N.S. Jayant, and N. Farvardin, "Three-Dimensional Subband Coding of Video," IEEE Transactions on Image Processing, vol. 4, no.2, February 1995.
- [38] A. Secker and D. Taubman, "Lifting-based invertible motion adaptive transform (LIMAT) framework for highly scalable video compression," submitted to IEEE Trans. Image Proc., 2002.
- [39] D. S. Taubman and A. Zakhor, "Multi-rate 3-D subband coding of video," IEEE Trans. Image Processing, vol. 3, pp. 572-588, Sept. 1994.
- [40] [J. Ohm, "Three dimensional subband coding with motion compensation," IEEE Trans. Image Proc., vol. 3, pp. 559-571, Sep 1994.
- [41] J.F.A. Magarey and N.G. Kingsbury, "Motion estimation using complex wavelets," Tech. Rep. CUED/F-INFENG/TR.226, Cambridge University Engineering Department, Aug. 1995.
- [42] J.F.A. Magarey, Motion estimation using complex wavelets, Ph.D. thesis, Cambridge University Department of Engineering, 1997.
- [43] J.F.A Magarey and N.G. Kingsbury, "Motion estimation using a complex-valued wavelet transform," IEEE Trans. on Signal Processing, special issue on wavelets and filter banks, vol. 46, no. 4, pp. 1069-84, April 1998.
- [44] D.J. Fleet and A.D. Jepson, "Computation of component image velocity from local phase information," Intern. J. Comput. Vis., vol. 5, pp.77-104, 1990.
- [45] J.G. Daugman, "Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters," J. Opt. Soc. Am. A, vol. 2, no. 7, pp. 1160-1169, July 1985.
- [46] N G Kingsbury, "The dual-tree complex wavelet transform: a new technique for shift invariance and directional filters", Proc. 8th IEEE DSP Workshop, Bryce Canyon, Aug 1998.

- [47] N G Kingsbury, "The dual-tree complex wavelet transform: a new efficient tool for image restoration and enhancement", Proc. EUSIPCO 98, Sept 1998.
- [48] N G Kingsbury, "Shift invariant properties of the Dual-Tree Complex Wavelet Transform", Proc. ICASSP 99, Phoenix, AZ, paper SPTM 3.6, March 16-19, 1999.
- [49] N G Kingsbury, "A Dual-Tree Complex Wavelet Transform with improved orthogonality and symmetry properties", Proc. IEEE Conf. on Image Processing, Vancouver, September 11-13, 2000, paper 1429.
- [50] N G Kingsbury, "Complex wavelets for shift invariant analysis and filtering of signals", Journal of Applied and Computational Harmonic Analysis, vol 10, no 3, May 2001, pp. 234-253.
- [51] N G Kingsbury and J F A Magarey, "Wavelet Transforms in Image Processing", Proc. First European Conference on Signal Analysis and Prediction, Prague, June 24-27, 1997, pp 23-34. (Invited paper.)
- [52] C W Shaffrey, N G Kingsbury and I H Jermyn, "Unsupervised Image Segmentation via Markov Trees and Complex Wavelets", Proc. IEEE Conf. on Image Processing, Rochester NY, Sept 23-25, 2002, paper 2324.
- [53] E P Simoncelli, W T Freeman, E H Adelson and D J Heeger, "Shiftable multiscale transforms", IEEE Trans. on Information Theory, 38(2), pp 587-607, March 1992.
- [54] A. Jalobeanu, N. Kingsbury, J. Zerubia, "Image deconvolution using Hidden Markov Tree modeling of complex wavelet packets", Proc. IEEE Conf. on Image Processing, Greece, Oct 8-10, 2001.
- [55] P F C de Rivaz and N G Kingsbury, "Bayesian Image Deconvolution and Denoising using Complex Wavelets", Proc. IEEE Conf. on Image Processing, Greece, Oct 8-10, 2001, paper 2639.
- [56] A Jalobeanu, L Blanc-Feraud and J Zerubia; "Satellite image deconvolution using complex wavelet packets", Proc. ICIP 2000, Vancouver, Sept 2000.
- [57] S Hatipoglu, S K Mitra and N G Kingsbury, "Texture Classification using Dual-Tree Complex Wavelet Transform", Proc. 7th International IEEE Conference on Image Processing and Its Applications, Manchester, England, July 12-15, 1999, pp 344-347.
- [58] P F C de Rivaz and N G Kingsbury, "Complex wavelet features for fast texture image retrieval", Proc. IEEE Conf. on Image Proc., Kobe, Japan, October 25-28, 1999.
- [59] P Hill and D Bull, "Rotationally Invariant Texture Features using the Dual-Tree Complex Wavelet Transform", Proc. ICIP 2000, Vancouver, Sept 2000.
- [60] A H Kam, T T Ng, N G Kingsbury and W J Fitzgerald, "Content based image retrieval through object extraction and querying", Proc. IEEE Workshop on Content-based Access of Image and Video Libraries (CBAIVL-2000), Hilton Head, South Carolina, June 12, 2000.
- [61] J Romberg, H Choi, R Baraniuk and N G Kingsbury, "Multiscale classification using complex wavelets", Proc, ICIP 2000, Vancouver, Sept 2000.
- [62] P F C de Rivaz and N G Kingsbury, "Fast segmentation using level set curves of complex wavelet surfaces", Proc. ICIP 2000, Vancouver, Sept 2000.
- [63] P Loo and N G Kingsbury, "Digital watermarking using complex wavelets", Proc. ICIP 2000, Vancouver, Sept 2000.
- [63] P Loo and N G Kingsbury, "Motion estimation based registration of geometrically distorted images for watermark recovery", Proc SPIE Conference on Security and Watermarking of Multimedia Contents III, San Diego, January 2001, paper 4314-68
- [65] S G Mallat, "A Wavelet Tour of Signal Processing, Academic Press, 1998.
- [---23---] H. Sari-Sarraf and D. Brzakovic, "A shift-invariant discrete wavelet transform," IEEE Trans. Signal Processing, vol. 45, no. 10, pp. 2621-2626, Oct. 1997.
- [66] P. Dutilleul, "An implementation of the *algorithme à trous* to compute the wavelet transform," in Wavelets: Time-Frequency Methods and Phase Space, J.-M.

- Combes, A. Grossman, and P. Tchamichian, Eds., pp. 298–304. Springer-Verlag, Berlin, Germany, 1989, Proceedings of the International Conference, Marseille, France, December 14–18, 1987.
- [67] Andreopoulos et al_IMBCTF_www.pdf
- [68] H. W. Park and H. S. Kim, "Motion estimation using low-band-shift method for wavelet-based moving picture coding", IEEE Trans. on Image Processing, Vol.9, No.4, pp.577-587, April 2000.
- [69] H. S. Kim and H. W. Park, "Wavelet-based moving-picture coding using shift-invariant motion estimation in wavelet domain," Signal Processing: Image Communication, vol. 16, no.7, pp 669-679, April 2001.
- [70] same as [24]
- [71] D. Taubman, "High performance scalable image compression with EBCOT," IEEE Trans. Image Processing, vol. 9, pp. 1158–1170, 2000
- [72] X. Li, "High Performance scalable coding based on motion compensated prediction of wavelet coefficients", submitted to VCIP'2001, San Jose, Jan. 2001
- [73] F. Dufaux, F. Moscheni, and M. Shutz, "Motion compensated wavelet transform coding," in Proceedings of the International Picture Coding Symposium, Sacramento, CA 1994
- [74] Van der Auwera, A. Munteanu, G. Lafruit, and J. Cornelis, "Video coding based on motion estimation in the wavelet detail images," in Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, Seattle, WA, May 1998, vol. 5, pp. 2801-2804
- [75] R. Calderbank, I. Daubechies, W. Sweldens, and B. Yeo, "Wavelet transforms that map integers to integers," Appl. Comput. Harmon. Anal., vol. 5, pp. 332–369, July 1998.
- [76] M. Adams and F. Kossentini, "Reversible integer-to-integer wavelet transforms for image compression: Performance evaluation and analysis," IEEE Trans. Image Processing, vol. 9, pp. 1010–1024, June 2000.
- [77] Saha, S. and Vemuri, R. Adaptive Wavelet Coding of Multimedia Images, Proc. ACM Multimedia Conference, Nov. 1999
- [78] M. Ohta and S. Nogaki, "Hybrid Picture Coding with Wavelet Transform and Overlapped Motion-Compensated Interframe Prediction Coding," IEEE Trans. Signal Proc., vol. 41, pp 3416-3424, Dec.1993. 58
- [79] K. Shen and E. J. Delp, "Wavelet Based Rate Scalable Video Compression," IEEE Trans. Circuits Syst. Video Technol., vol. 9, pp. 109-122, Feb. 1999.
- [80] D. Marpe and H. L. Cycon, "Very Low Bit-Rate Video Coding Using Wavelet-Based Techniques," IEEE Trans. Circuits Syst. Video Technol., vol. 9, pp. 85-94, Feb. 1999
- [81] S. A. Martucci, I. Sodagar, T. Chiang, and Y.-Q. Zhang, "A zerotree wavelet video coder," IEEE Transactions on Circuits and Systems for Video Technology, vol. 7, no. 1, pp. 109–118, February 1997.
- [Cal96] Calderbank, A. R. and I. Daubechies, W. Sweldens, B.-L. Yeo WAVELET TRANSFORMS THAT MAP INTEGERS TO INTEGERS. Proceedings of the IEEE Conference on Image Processing. Preprint, 1996. IEEE Press, 1997. To appear.
- [Che95] Chen, W.-K., editor. THE CIRCUITS AND FILTERS HANDBOOK. Boca Raton, FL (USA): CRC Press, 1995. The Electrical Engineering Handbook Series.
- [Cla97] Claypoole, R. and G. Davis, W. Sweldens, R. Baraniuk. NONLINEAR WAVELET TRANSFORMS FOR IMAGE CODING. Asilomar Conference on Signals, Systems, and Computers. Preprint, 1997. To appear.
- [Dau97] Daubechies, I. and W. Sweldens. FACTORING WAVELET TRANSFORMS INTO LIFTING STEPS. J. Fourier Anal. Appl., Vol. 4, Nr. 3, 1998, preprint.

Part 4 Conclusion

- [Kov97] Kovacevic, J. and W. Sweldens WAVELET FAMILIES OF INCREASING ORDER IN ARBITRARY DIMENSIONS. To appear in IEEE Transactions on Image Processing. Preprint 1997.
- [Mor82] Morlet, J. and G. Arens, I. Fourgeau, D. Giard. WAVE PROPAGATION AND SAMPLING THEORY. Geophysics, Vol. 47 (1982), p. 203-236.
- [Sto98] Stoffel, A. REMARKS ON THE UNSUBSAMPLED WAVELET TRANSFORM AND THE LIFTING SCHEME. Elsevier Science. Preprint, 1998.
- [Swe96a] Sweldens, W. THE LIFTING SCHEME: A CONSTRUCTION OF SECOND GENERATION WAVELETS. Siam J. Math. Anal, Vol. 29, No. 2 (1997). Preprint, 1996.
- [Swe96b] Sweldens, W. BUILDING YOUR OWN WAVELETS AT HOME. In: Wavelets in Computer Graphics. ACM SIGGRAPH Course Notes, 1996.
- [Uyt97a] Uytterhoeven, G. and A. Bultheel. THE RED-BLACK WAVELET TRANSFORM. Technical report TW271, Department of Computer Science. Leuven: Katholieke Universiteit Leuven, 1997.
- [Uyt97b] Uytterhoeven G. and F. Van Wulpen, M. Jansen, D. Roose, A. Bultheel. WAILI: WAVELETS WITH INTEGER LIFTING. Technical report TW262, Department of Computer Science. Leuven: Katholieke Universiteit Leuven, 1997.
- [Uyt97c] Uytterhoeven G. and D. Roose, A. Bultheel. WAVELET TRANSFORMS USING THE LIFTING SCHEME. Report ITA-Wavelets-WP1.1, Department of Computer Science. Leuven: Katholieke Universiteit Leuven, 1997.
- [Wei94] Weiss, L. G. WAVELETS AND WIDEBAND CORRELATION PROCESSING. IEEE Signal Processing Magazine, January (1994), p. 13-32.

References to Part PROTOTYPE:

- [1] <http://java.sun.com> - Java Sun web sider
- [6] <http://java.sun.com/jmf> - Java Sun's JMF web sider
- [11] Kjetil Haslum, Terje Gjøsæter, Rolf Bjerke, Bjørn-Atle Wiik - Bildekomprimeringsmetoder, Hovedprosjekt for ingeniørutdanningen, Avdeling for teknologi, Grimstad, 25.05.2001
- [8] <http://gcc.gnu.org/java> - GNU - GCJ
- [13] Quicktime File Format
- [14] AVI File Format
- [12] JMFRegistry User's Guide - Java Sun's web sites