ENHANCEMENTS TO SPIHT-BASED AND CSPIHT-BASED CODERS

SEE TOH CHEE WAH
(B. Eng. (Hons), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2003

ABSTRACT

SPIHT and CSPIHT are two robust and efficient compression algorithms for encoding monochrome and color images, respectively. This thesis introduces two enhancements, *Virtual Level* and *Flexible Spatial Orientation Tree*, which can be used to improve the compression efficiencies of both SPIHT and CSPIHT. These enhancements are designed, implemented and tested by the author. Evaluation of compression results shows that both enhancements improve the compression efficiencies of SPIHT and CSPIHT, but only under the special circumstance that less DWT decomposition levels are performed due to image dimensions and limitation of the original Spatial Orientation Tree structure. Among the two enhancements, *Flexible Spatial Orientation Tree* allows the compression of arbitrary sized images, yields better compression and is clearly superior over *Virtual Level*.

ACKNOWLEDGEMENTS

The author would like to express his heart-felt gratitude to his supervisor, Prof. Ashraf A. Kassim for all his valuable ideas, guidances and suggestions throughout the duration of his master degree. Secondly, the author would like to thank Mr. Lee Wei Siong for sharing his experience in video coding that had helped the author in one way or another. Last but not least, the author would also like to thank Mr. Francis Hoon for taking care of his logistics needs.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

BMA	Block Matching Algorithm
CODEC	COmpression/DECompression
CIF	Common Intermediate Format
CSPIHT	Color Set Partitioning In Hierarchical Tree
CSPIHT-VL	CSPIHT using VL
CSPIHT-FSOT	CSPIHT using FSOT
CVCS	CSPIHT-based Video Coding System
DCT	Discrete Cosine Transform
DWT	Discrete Wavelet Transform
EZW	Embedded Zertree Wavelet
FSOT	Flexible Spatial Orientation Tree
H.261	ITU-T Recommendations
H.263	ITU-T Recommendations
IC	Integrated Circuit
JPEG	Joint Photographic Expert Group
JPEG2000	Joint Photographic Expert Group 2000 image format
KL	Karhunen-Loève
LIP	List of Insignificant Pixel
LIS	List of Insignificant Set
LSP	List of Significant Pixel
MPEG	Moving Picture Experts Group
MSE	Mean Square Error
PCM	Pulse Code Modulation
PRA	Pel Recursive Algorithm
PSNR	Peak Signal to Noise Ratio
QCIF	Quarter Common Intermediate Format
RGB	Red-Green-Blue
SAD	Sum of Absolute Difference
SPIHT	Set Partitioning In Hierarchical Tree
SPIHT-VL	SPIHT using VL
SPIHT-FSOT	SPIHT using FSOT
SAD	Sum of Absolute Difference
SOT	Spatial Orientation Tree
VL	Virtual Level
YUV	Luminance-Chrominance color space

1 INTRODUCTION

Since the introduction of television, there has been an interest in reducing the transmission bandwidth and yet preserving a smooth and acceptable image and video quality. Television broadcasting systems were first designed to be analog, meaning that signals are transmitted as a continuous-valued representation of the picture. In recent years, through the rapid development of the computer industry, digital video systems is available where the picture is represented by a stream of bi-level values of one and zero.

Digital video systems, which are expected to replace analog video systems, are considered superior due to several reasons. Firstly, digital video systems are able to guarantee signal quality and completely determine the signal performance at the point of creation, which implies a perfect reproduction of received video signals. This is a huge advantage over analog systems that suffers from signal degradation that arises from cascaded processing, transmission or storage. Secondly, advances in integrated circuit (IC) technology have made digital hardware smaller, cheaper, more reliable and easier to design as compared to analog hardware. Thirdly, some processes like signal delay and dynamic video special effects are easier accomplish in the digital domain.

Figure 1-1 shows the elements of creation, storage, transmission and reproduction for analog television, digital television and computer.

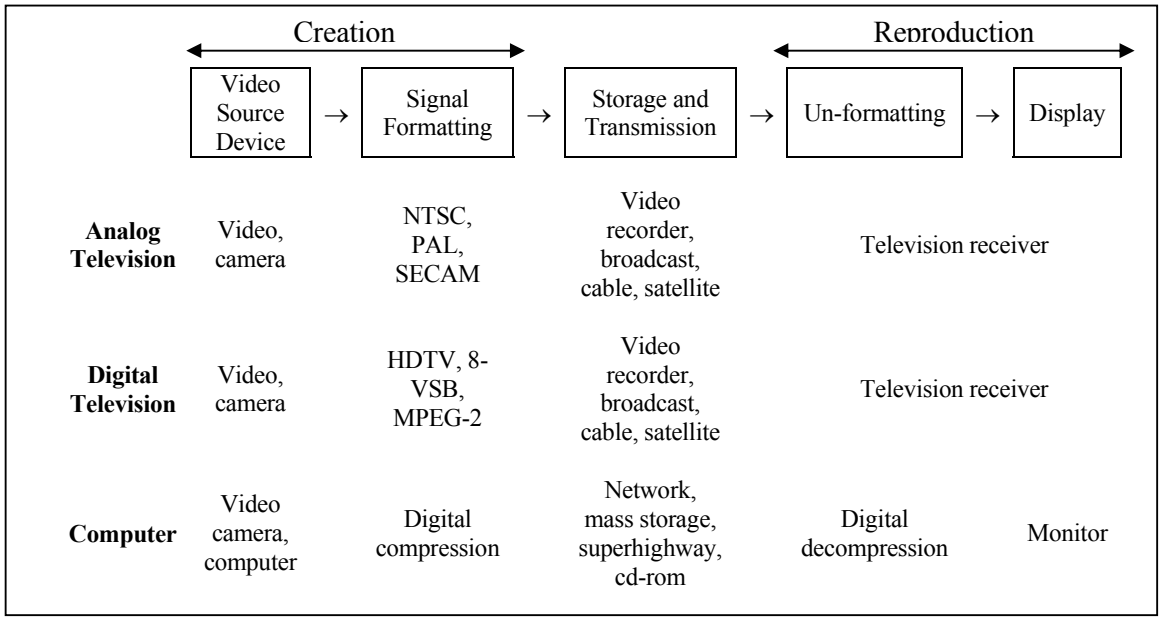


Figure 1-1 Elements of video systems

Storage and transmission are two important parts of a video system. In a digital video system, the requirements for both storage and transmission are extremely high. Consider a full motion video which is 640×480 with 24 bits/pixel, 30 frames/second and lasts for 10 seconds. The storage capacity and transmission bandwidth needed will be 2.21 gigabits and 0.221 gigabits/sec respectively, which are too high for real world application. This has called for image and video compression.

Compression refers to the science of reducing the amount of data that is used in conveying information. Compression relies on the fact that information is not random and exhibits certain degree of order and patterns, hence by extracting these order and patterns, the 'redundancy' within the information can be

eliminated or reduced so that less amount of data is needed to convey the same information.

Lossless compression allows perfect recovery of data, meaning that if some data is compressed, we can reverse the process and get back the same data that we started with. Some examples of lossless compression technique include run length coding and entropy coding. Although perfect recovery of compressed data is usually considered as the ideal of compression, there are two serious disadvantages that limit the usefulness of lossless compression. Firstly, lossless compression only offers small compression ratio; secondly, it is data dependent and cannot guarantee a constant output data rate, which may be needed for a transmission channel.

Lossy compression achieves better compression ratio by removing irrelevant information, which the end recipient cannot perceive that it is missing. In most cases information that is close to irrelevant is also removed if the loss in quality is small as compared to the savings in data. The objective of lossy compression is hence to maximize compression ratio or minimize bit rate and minimize the loss in quality.

In most image and video compression scheme, lossy compression is usually first applied to generate a compressed bit stream, which will then be further compressed by using lossless compression. JPEG [1] and JPEG2000 [2] are two existing international standards for compressing still images while H.261, H.263, MPEG1 and MPEG2 [1] are used for video compression.

The emergence of international standards for encoding image and video data had enabled a wide range of applications that make use of digital image and video storage and transmission. A wide range of digital image and video applications are expected to continue to grow and includes (i) video conferencing and videotelephony; (ii) home entertainment digital video; (iii) broadcast digital television; (iv) video database and video-on-demand; (v) medical imaging; (vi) image/video transmission system on wireless devices. There are growing needs for better compression techniques to enable such applications.

1.1 OVERVIEW OF THESIS

Set Partitioning In Hierarchical Tree (SPIHT) and Color Set Partitioning In Hierarchical Tree (CSPIHT) are two robust and efficient compression algorithms for compression of monochrome and color images respectively. In this report, two enhancements known as *Virtual Level* (VL) and *Flexible Spatial Orientation Tree* (FSOT), which can be applied to SPIHT and CSPIHT, will be introduced. The performance of these enhancements will be analyzed and evaluated extensively in this report.

Traditional image compression analysis are performed on large and standard sized images like lena and barbara that are 512×512. There are many situations where such analyses are inadequate in evaluating the efficiencies of an image compression algorithm. A design consideration of JPEG2000, for example, is that it must be able to encode images with arbitrary dimensions. Such an requirement is necessary because in the real world, images come in all sort of

dimensions. A good compression algorithm must hence be robust enough to work well in compressing these images with arbitrary dimensions.

As SPIHT and CSPIHT are both unable to compress images with arbitrary dimensions, their applications in the real world will be very much limited. One of the enhancements, FSOT, is able to overcome this problem completely. With this added capability, it is then possible to compare SPIHT-FSOT with JPEG2000 using images with dimensions other than those of the standard test images.

In recent years, we have witnessed a wide spread adoption of handheld devices demanding image and video compression technologies. Such devices are limited in its display area, which give rise to the importance of compressing small images. It has already being shown that the upcoming JPEG2000 image compression algorithm is superior to SPIHT when compressing standard images like lena and barbara [3][4]. There is, however, not a single study on the compression efficiency of JPEG2000 in compressing small images. In this report, analysis of image compression is hence primarily focused on images that are smaller than standard test images. Our results show that SPIHT using FSOT is superior as compared to JPEG2000 in compression of smaller images

Video compression is often performed on sequences with QCIF and CIF dimensions, which are much smaller than normal test images. As SPIHT using FSOT is superior in compressing smaller images, it also improves the performance of the video compression system that uses it as the core quantization block.

The rest of the thesis is as follows:

Chapter 2 first introduces the background of image coding, focusing on the theory and application of DWT. This is followed by the introduction of general principles behind video coding, in particular, BMA will be explained in greater detail. Several DWT-based compression techniques, including the SPIHT and CSPIHT compression algorithm, will be explained in Chapter 3.

In Chapter 4 and Chapter 5, the enhancements VL and FSOT are introduced and their algorithms are explained. The image compression improvements are included in Chapter 6 while the video compression improvements are included in Chapter 7. Chapter 8 is a conclusion to this thesis. The paper of SPIHT [5] is added to the end of this report as Appendix I.

2 IMAGE AND VIDEO CODING

This chapter aims introducing readers to the basic concepts of image and video compression.

2.1 IMAGE CODING

This section provides a short overview on image coding, focusing on the basic modules that make up an image encoder, and introduces some of the common techniques employed in implementing the individual modules.

Image coding exploits the visual redundancy within an image so as to achieve compression. This is generally implemented by three separate components that are shown in Figure 2-1.

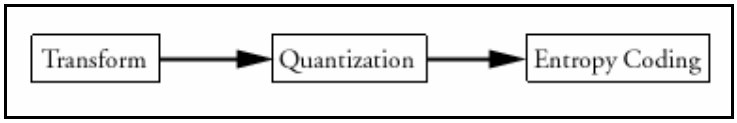


Figure 2-1 Image coding

Although it is possible to perform source coding on the original signal (the image), it is often more useful to first transform the signal into another domain where processing can be more effectively applied. The main advantage of transform coding is that the coefficients in the transform domain are decorrelated, meaning that the autocovariance matrix of the signal is diagonalized, which allows the coefficients to be quantized independently. The Karhunen-Loève (KL) [6] transform can achieve exact diagonalization, but is impractical as it is signal

dependant. This led to the use of other transforms that can approximate this diagonalization property.

Quantization exploits the characteristics of energy concentration in the transform domain so that compression can be achieved. If a transformation T on signal x results in a vector v (see equation 2.1) with a few large coefficients, these coefficients are considered as high energy and can be coded with higher precision than other coefficients.

$$v = Tx \tag{2.1}$$

After quantization, signal x' can be reconstructed by performing inverse transform on v', which is the quantized version of v:

$$x' = T^{-1}v' \tag{2.2}$$

JPEG [1] and JPEG2000 [2] are two well-known image compression schemes that use Discrete Cosine Transform (DCT) [7] and Discrete Wavelet Transform (DWT) [7] , respectively. It has been shown that JPEG2000 has much better compression performance. DWT, which is normally used for lossy compression, breaks up the input signal into constituent parts by applying sequential bandpass filters to an entire frame, and divides the image into a collection of sub-images separated by their frequency composition. Natural images comprised of low frequency have gradual brightness changes and smooth background areas with less contrast. These images can be represented with fewer terms and still be reconstructed with little distortion.

There are numerous quantization techniques available. In this thesis, the Set Partitioning in Hierarchical Tree (SPIHT) [5] and Color Set Partitioning in Hierarchical Tree (CSPIHT) [12] algorithms are used for quantization of the DWT coefficient map, and will be discussed extensively from Chapter 3 onwards.

2.1.1 WAVELET AND DWT

A wave is generally defined as an oscillating function both in time and space, and a wavelet is a "small wave", which has its energy concentrated in time so that it can be used as a tool to analyze transient, non-stationary, or time-varying phenomena. Wavelet [11] retains the wavelike characteristics but at the same time allows simultaneous frequency and time analysis.

The fundamental idea behind wavelets is to analyze according to scale. Before delving into the DWT it is important to first understand how wavelet analyzes by scale. In the following sub-section, multi-resolution analysis will be introduced to illustrate the concept of analysis according to scale.

2.1.1.1 Multi-resolution Analysis

Wavelet transform is based on the concept of multi-resolution analysis [8,11], which consists of breaking up $L^2(\mathbf{R})$ into a nesting of spanned space

$$\ldots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \ldots \tag{2.3}$$

where $V_m \to L^2(\mathbf{R})$ as $m \to \infty$ and $V_m \to \{0\}$ as $m \to -\infty$. Because of the definition of V_j , the spaces have to satisfy a natural scaling condition

$$f(t) \in V_j \Leftrightarrow f(2t) \in V_{j+1} \tag{2.4}$$

By defining a set of scaling functions in terms of integer translate of the basic scaling function,

$$\varphi_k(t) = \varphi_k(t-k),\ k \in \mathbf{Z},\ \varphi \in L^2 \tag{2.5}$$

the subspace of $L^2(\mathbf{R})$ spanned by the scaling functions can be defined as

$$V_0 = \overline{Span_k\{\varphi_k(f)\}} \tag{2.6}$$

for all integer k from $-\infty$ to $+\infty$. From the basic scaling function, a family of functions can be generated by scaling and translation

$$\varphi_{j,k}(t) = 2^{j/2}\varphi(2^j t - k) \tag{2.7}$$

Where the span the basic scaling function over $k \in \mathbf{Z}$ is

$$V_j = \overline{Span_k\{\varphi_k(2^j t)\}} = \overline{Span_k\{\varphi_{j,k}(t)\}} \tag{2.8}$$

From (2.4), $f(t)$ can be written as

$$f(t) = \sum_k a_k \varphi(2^j t + k) \tag{2.10}$$

The nesting of subspaces in (2.3) combined with (2.10) imply that if $\varphi \in V_0$ and $V_0 \subset V_1$, we have $\varphi \in V_1$ the space spanned by $\varphi(2t)$. Thus allowing $\varphi(t)$ to be expressed as:

$$\varphi(t) = \sum_n h(n)\sqrt{2}\varphi(2t - n) \tag{2.10}$$

where $h(n)$ are called the filter coefficients of φ.

Wavelets can now be defined by a set of functions $\psi_{j,k}(t)$ that span the difference between spaces spanned by the various scales of the scaling function. By defining the orthogonal complement of V_j in V_{j+1} as W_j, it is thus required that

$$<\varphi_{j,k}(t), \psi_{j,l}(t)> = 0, j,k,l \in \mathbf{Z} \tag{2.11}$$

From (2.3), the wavelet spanned subspace W_0 can be defined such that

$$V_1 = V_0 \oplus W_0 \tag{2.12}$$

which extends to

$$V_2 = V_0 \oplus W_0 \oplus W_1 \tag{2.13}$$

and in general,

$$L^2 = V_0 \oplus W_0 \oplus W_1 \oplus \ldots \tag{2.14}$$

The scale of the initial space V_0 is arbitrary. We can further extending V_0 to $-\infty$ such tha

$$L^2 = \ldots W_{-2} \oplus W_{-1} \oplus W_0 \oplus W_1 \oplus \ldots \tag{2.15}$$

Note how the wavelet spaces are related to V_0. Since the wavelets reside in the space spanned by the next finer scaling function, $W_0 \subset V_1$, we can write

$$\psi(t) = \sum_n g(n)\sqrt{2}\varphi(2t-n) \tag{2.16}$$

for some filter coefficient $g(n)$. By orthogonality implied by definition of (2.16), it is required

$$g(n) = (-1)^n h(1-n) \tag{2.17}$$

2.1.1.2 Discrete Wavelet transform

Having constructed the set of functions φ and ψ that can span the entire $L^2(\mathbf{R})$, according to (2.14) any function $f(t) \in L^2(\mathbf{R})$ can be written as

$$f(t) = \sum_k c_j(k)\varphi_k(t) + \sum_k \sum_{j=0}^{\infty} d_j(k)\psi_{j,k}(t) \tag{2.18}$$

The coefficients in equation (2.18) are called the *discrete wavelet transform* (DWT) of the signal $f(t)$. The first summation gives a coarse approximation of $f(t)$ and the second summation gives the details. If φ and ψ are each orthogonal and given (2.11), the coefficients can be written as

$$c_j(k) = \langle f(t), \varphi_{j,k}(t) \rangle = \int f(t) 2^{j/2} \varphi(2^j - k)dt \tag{2.19}$$

and

$$d_j(k) = \langle f(t), \psi_{j,k}(t) \rangle = \int f(t) 2^{j/2} \psi(2^j - k)dt \tag{2.20}$$

Recall (2.10), scaling and time translating the equation yields

$$\varphi(2^j - k) = \sum_n h(n)\sqrt{2}\varphi\left[2(2^j - k) - n\right] = \sum_n h(n)\sqrt{2}\varphi(2^{j+1}t - 2k - n) \tag{2.21}$$

Substituting (2.21) into (2.19),

$$c_j(k) = \sum_n h(n) \int f(t) 2^{j+1/2} \varphi(2^{j+1}t - 2k - n)dt \tag{2.22}$$

By substituting $m = 2k + n$, (2.22) becomes

$$c_j(k) = \sum_m h(m - 2k) \int f(t) 2^{j+1/2} \varphi(2^{j+1}t - m)dt \tag{2.23}$$

The scaling function inside the integral is at scale $j + 1$, thus (2.23) can be further simplified to

$$c_j(k) = \sum_m h(m-2k)c_{j+1}(m) \tag{2.24}$$

Similarly,

$$d_j(k) = \sum_m g(m-2k)c_{j+1}(m) \tag{2.25}$$

(2.24) and (2.25) are actually a convolution operation followed by a down sampling of factor 2. These equations demonstrates that scaling and wavelet coefficients at scale j –1 can be obtained by convolving the coefficients at scale j by the time-reversed filter coefficients h and g followed by decimating. These two equations form the algorithm needed to perform forward DWT which is used in encoding.

For reconstruction of the original signal from the scaling function and wavelet coefficients, first consider a signal $f(t) \in V_{j+1}$,

$$f(t) = \sum_k c_{j+1}(k)2^{(j+1)/2}\varphi(2^{j+1}t-k) \tag{2.26}$$

Rewriting in terms in the next lower scale

$$f(t) = \sum_k c_j(k)2^{j/2}\varphi(2^j t-k) + \sum_k d_j(k)2^{j/2}\psi(2^j t-k) \tag{2.27}$$

Substituting (2.10) and (2.16) into (2.27),

$$f(t) = \sum_k c_j(k)\sum_n h(n)2^{(j+1)/2}\varphi(2^{j+1}t-2k-n) + \sum_k d_j(k)\sum_n g(n)2^{(j+1)/2}\psi(2 \tag{2.28}$$

Taking inner product on both sides of (2.28) with $\varphi(2^{j+1}t - k')$,

$$c_{j+1}(k) = \sum_m h(k-2m)c_j(m) + \sum_m g(k-2m)d_j(m) \tag{2.29}$$

(2.29) is the reconstruction equation that constitutes a up-sampling the scaling function and wavelet coefficients by factor 2 followed by a convolution with their respective filter coefficients. The set of decomposition and reconstruction operations (2.24), (2.25) and (2.29) is known as Mallat's algorithm [8] for fast orthogonal DWT.

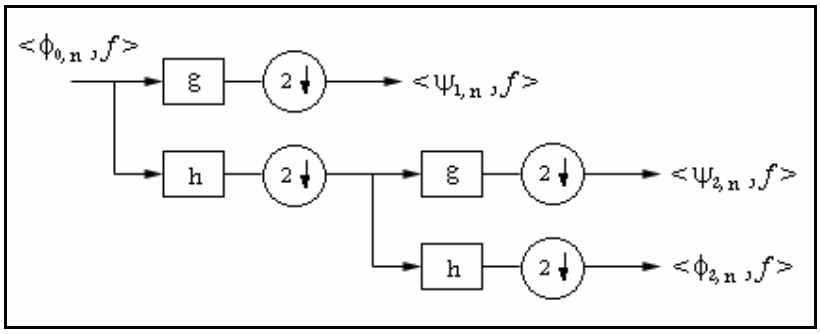


Figure 2-2 Signal decomposition via wavelet filter bank

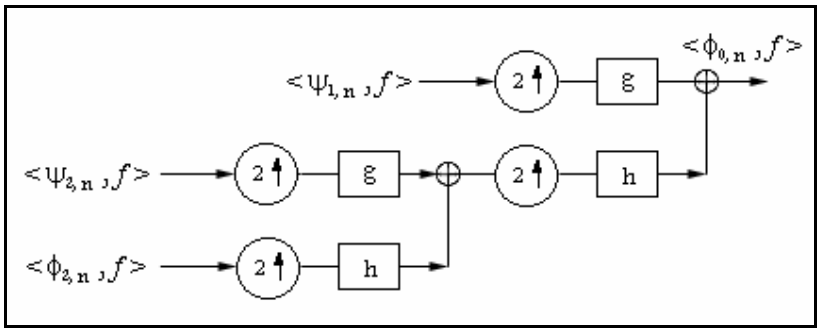


Figure 2-3 Signal reconstruction via wavelet filter bank

The algorithm structure illustrated in Figure 2-2 and 2-3 will be the basis for implementing DWT of images in this project.

2.1.2 QUANTIZATION

Generally, it is desired to have an encoding scheme that allows for incremental decoding on the receiver side so that a coarse image can be progressively refined as more bits are received. One example of such an image format is the *Progressive JPEG* [1]. An *embedded bit stream* is one that all encoded bits at a lower bit rate will be embedded in the beginning of the bit stream of higher bit rate. This characteristic allows truncated bit stream to be decoded up to the point where it is truncated, hence allowing progressive decoding.

There are several techniques hat can be used to quantize DWT coefficient map as an *embedded bit stream*, but all of them have one similarity, they group and send the bits of the decomposition coefficients in order of significance. These techniques will be further discussed in Chapter 3.

2.2 VIDEO CODING

Video coding is made up of two main types: intra-frame coding and inter-frame coding. Although it is possible to encode the whole video purely on a frame-by-frame basis, it is often more appropriate and less costly to apply *inter-frame coding*, which encodes the *displaced frame difference* image and exploits inter-frame redundancies. By only sending the information that changes from a reference frame, temporal redundancy can be effectively eliminated if there is little motion between consecutive frames.

Most research interests are focused on hybrid schemes that combine both intra-frame coding and inter-frame coding. H.261, H.263, MPEG1 and MPEG2 [1] are compression standards that use both inter and intra-frame compression.

Intra-frame coding exploits redundancies within a single frame and is the same as image coding. This section will focus on inter-frame coding. An overview of inter-frame coding is first provided and some of the methods available will be discussed, followed by a detailed discussion on block based motion estimation and compensation for inter-frame coding. Lastly, the details on residual signal processing will be documented.

In inter-frame coding, temporal redundancy is exploited to achieve compression. Predictive methods are used to predict the desired frame based on a reference frame. Figure 2-4 shows the general inter-frame coding model.

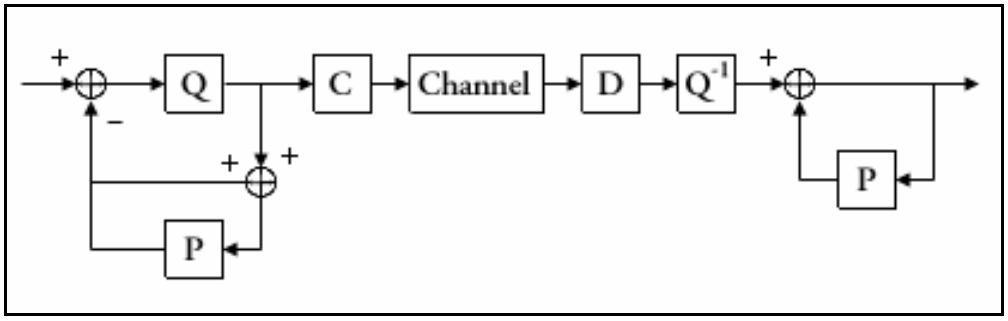


Figure 2-4 Predictive encoder and decoder

The prediction block P can be any predictive coding method. The simplest form of predictive coding will be one that simply encodes the difference in pixel value of consecutive frames. However, such scheme will only be useful in video

sequence where there is minimal motion. Generally, inter-frame coding with motion estimation and compensation will be more effective.

Motion estimation identifies object movement and represents these movements as vectors. Motion examines the movement vectors to achieve data compression. Motion estimation and compensation are used in conjunction to effectively predict a frame based on a reference frame. Compared with the simple frame difference scheme, motion estimation and compensation can reduce the predictive errors of the predicted frame. Thus, the high temporal redundancy, which exists due to high correlation between consecutive frames, is eliminated more effectively.

Motion may exist in a complex form involving translations, rotations and scaling. It will be extremely tedious and difficult to estimate all these combinations. Most current estimation models consider only translation, and had been proven to be successful [9]. As a result, most motion estimation and compensation techniques make the following assumptions: (1) objects translate in a plane parallel to the camera, (2) illumination is spatially and temporally uniform and (3) occlusion of one object by another, and uncovered background are neglected.

There are 2 main types of motion estimation algorithm: i) pel-based estimation in *Pel Recursive Algorithms* (PRA), and ii) block-based estimation in *Block Matching Algorithms* (BMA). PRA is complex and has a slow rate of convergence, this leads to the adaptation of BMA methods in this project, which will be discussed in detail in the next sub-section.

2.2.1 BLOCK MATCHING ALGORITHM

BMA[9] are the most commonly used motion estimation and compensation method in the industry, as BMA has a good balance between computational complexity and coding performance. Popular video format like H.261, H.263 and MPEG uses BMA. In BMA, the image frame is broken down into blocks, and it is assumed that the translations of these blocks are equal. Figure 2-5 and Figure 2-6 show the process of motion estimation and compensation using BMA.

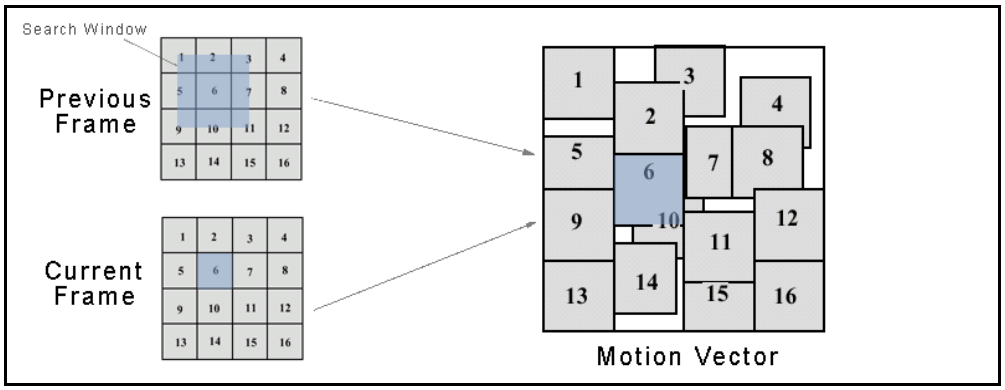


Figure 2-5 Motion estimation using BMA

In Figure 2-5, both previous frame and current frame are partitioned into 16 blocks as shown. Motion vectors are then estimated by searching the best match within the search window (of the corresponding block) in the previous frame. There are several matching criteria that are used to identify the best match, the most frequently used is the *Sum of Absolute Difference* (SAD) [9]:

$$SAD = \sum_{x=1}^{M} \sum_{y=1}^{N} \left| B_{i,j}(x,y) - B_{i-u,j-v}(x,y) \right| \tag{2.30}$$

From equation 2.30, it is clear that motion estimation is extremely computational expensive as SAD must be re-calculated whenever a new location is to be tested within the search window. To find the best match, an exhaustive search must be performed and this can lead to delay. Other search techniques that are less computational intensive can be applied with a trade off in reconstructed image quality.

Motion compensation is a process whereby the motion vectors are used to reconstruct the original (current) frame. Figure 2-6 illustrates how this is done. By using the motion vector derived in motion estimation, the best match block is extracted from the previous frame and pasted into the reconstructed frame. There are 16 blocks that are identified by the 16 motion vectors, hence forming the complete reconstructed frame.

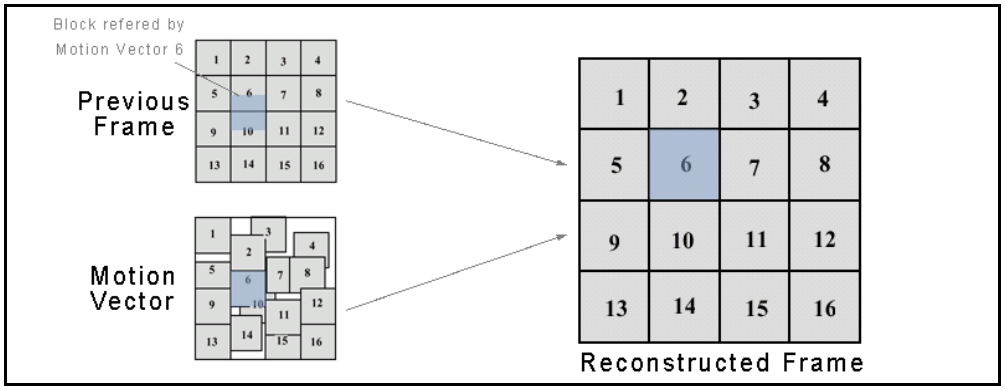


Figure 2-6 Motion compensation using BMA

2.2.2 RESIDUAL ERROR PROCESSING

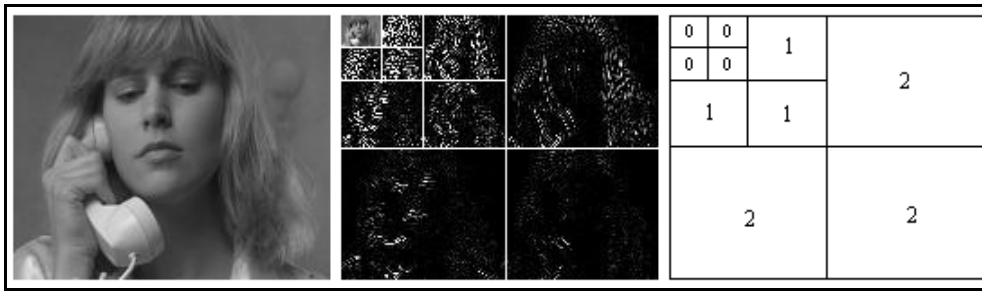
Reconstructed image using BMA consists of blocks from the reference frame that are predicted. To improve reconstructed image quality, extra encoding

is performed on the predictive error, which also known as the residual error, so that the reconstructed image is closer to the original image.

Residual error signal is usually processed using intra-frame coding techniques, so that a residual error bit-stream can be generated to supplement the motion vectors. The combined motion vectors and residual error bit streams will then be extracted and decoded for reconstruction of signal.

3 WAVELET-BASED COMPRESSION ALGORITHMS

In this chapter, several algorithms that are used for quantization in the wavelet domain will be introduced.



(a) (b) (c)

Figure 3-1 Two level DWT of suzie
(a)Original Image. (b) DWT Coefficient Map (c) Sub-band Decomposition

Figure 3-1 (b) shows how the coefficient map looks like after a two level DWT on image (a) is performed. Different levels of decomposition in (c) represent the different sub-bands that are present in (b). Sub-bands labeled '2' contain high frequency components, and sub-bands label '0' are areas where low frequency components are located. As a result of the smoothness of natural image, energy is distributed in DWT coefficient map such that lower sub-bands have higher energy, in particular, the highest amount of energy is concentrated on the top-left portion of the level 0 sub-band. There are also observed self-similarities across sub-bands that exist as a form of decreasing energy, meaning that coefficients corresponding to the same spatial position is smaller in higher sub-band. These characteristics are exploited by various quantization techniques to achieve compression.

3.1 EZW

One of these techniques is the Embedded ZeroTree Wavelet (EZW) algorithm, which is developed by Shapiro [10] based on 3 concepts: (i) hierarchical sub-band decomposition, (ii) exploitation of self-similarity across scales of an image wavelet transform to predict absence of significant information, and (iii) transmission of refinement bits by ordered bit plane. EZW is a simple and effective DWT coefficients quantization algorithm that generates a fully embedded bit stream in order of importance.

Trees can be effectively used to represent the DWT coefficient map because of the sub-sampling performed during DWT. Nodes at the higher sub-bands can be considered as descendant to nodes from the lower sub-band that correspond to the same spatial location on the image. A coefficient in the lower sub-band can be thought of having 4 descendants in the next higher sub-band, of which each of theses descendants will have another 4 descendants each. A quad tree, which all its descending nodes are equal or smaller than its root, can hence be formed. Figure 3-2 shows the scanning path used by EZW.

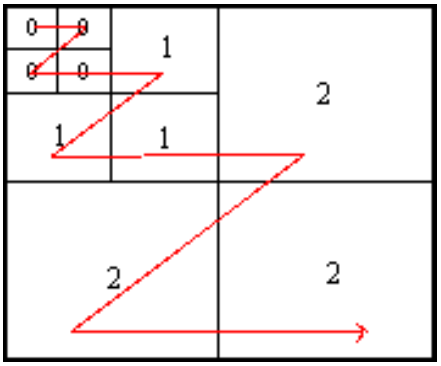


Figure 3-2 EZW scanning order

EZW scans the DWT coefficient map in a zigzag manner, which effectively reduces the number of symbols needed to encode the tree as values of coefficients decrease with increasing sub-band level. Interestingly, if encoding and decoding is terminated in the middle of a pass of EZW coding, there will be no visible mosaic effect as opposed to progressive JPEG. Figure 3-3 shows a visual comparison of between EZT and progressive JPEG with the same rate of encoding and decoding.

(a) (b)

Figure 3-3 Lena reconstructed based on different schemes with the same rate
(a)DWT + EZW. (b) Progressive JPEG

3.2 SPIHT AND CSPIHT

In *EZW*, a high bit budget is allocated to code the significance map and the predefined zig-zag scanning order requires the entire lower sub-band to be scanned before the coefficients in the higher sub-band. Such extra overheads can be eliminated by more effectively exploiting the characteristics of DWT map, which is offered by the SPIHT algorithm developed by Said and Pearlman [5].

Although both SPIHT and EZW are fast, simple and capable of producing an embedded bit stream, SPIHT is superior to EZW in one very important aspect: SPIHT's compression performance is much better than EZW [5]. Our research efforts hence focus on SPIHT for its superiority.

3.2.1 SET PARTITIONING IN HIERARCHICAL TREE

SPIHT is defined by a tree structure called *spatial orientation tree* (SOT). SOT is a tree spanned to link up coefficients of that belongs to the same spatial location in the image domain, and performs magnitude test on the *branches* of the tree. Every *branch* can be considered as a partitioned subset of the pixels that correspond to the same region in an image. If found to be significant, the *branch* will be partitioned into new and smaller *branches* and the significant test is further applied to the new *branches*. This process repeats until all individual significant coefficients are identified in the branches. Hence, SPIHT actually propagates into the next sub-band if any partitioned coefficients sets in the higher sub-band of the same spatial orientation are found to be significant. In this manner, SPIHT reduces the number of coefficients to be process in dominant/sorting pass.

Figure 3-4 demonstrates how the SOT is defined in a 2-level sub-band decomposition map. This spatial relationship on the hierarchical pyramid exploits spatial self-similarity across the sub-bands.

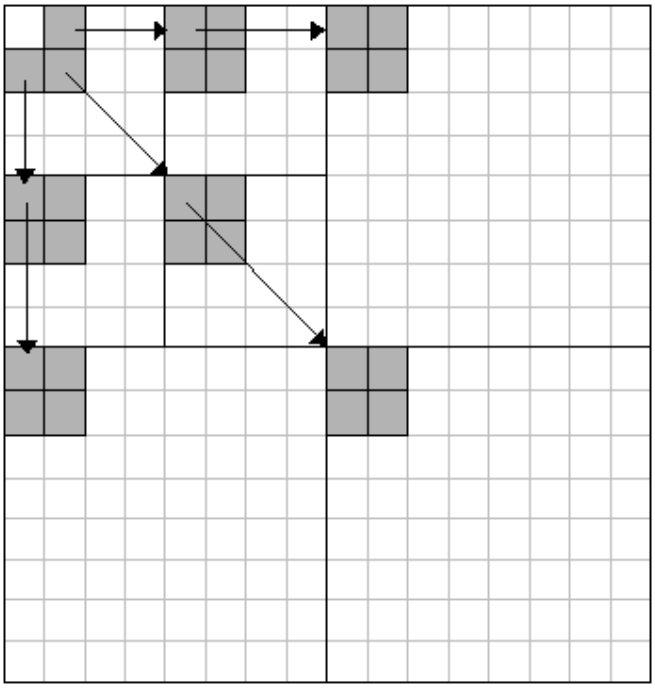


Figure 3-4 Spanning relationship used in SOT of SPIHT

The SPIHT algorithm maintains three lists, namely, *List of Insignificant Pixel* (LIP), *List of Insignificant Set* (LIS) *and List of Significant Pixel* (LSP). A sorting pass is used to identify the significance of sets and pixels with respect to the current threshold, and a refinement pass is used to increase the precision of the identified significant coefficients in the previous sorting pass. The algorithm of SPIHT is shown in pseudo-code form in Figure 3-5.

```
1) Initialization:
1.1) Output n= ⌊log2 (max (i, j) {| ci, j |}⌋;
1.2) Set the LSP, LIP and LIS as empty lists;
1.3) Add the coordinates (i,j) ∈ H to LIP and LIS as Type A;
2) Sorting Pass:
2.1) For each entry (i,j) in LIP do:
      2.1.1) Output S(i,j);
      2.1.2) If S(i,j)=1 then
      -Move (i,j) to LSP;
      -Output sign of ci,j;
2.2) For each entry (i,j) in LIS do:
      2.2.1) If entry is TYPE A then
```

```
      -Output S(Descendant of (i,j));
      +If S(Descendant of (i,j))=1 then
            For each offspring (k,l) of (i,j) do:
            -Output S(k,l);
            -if S(k,l)==1 then add (k,l) to LSP and output sign of
c_{k,l};
            -if S(k,l)==0 then add (k,l) to end of LIP;
      +If L(i,j)!=0 move (i,j) to end of LIS as TYPE B and go
2.2.2;
      +otherwise remove (i,j) from LIS;
      2.2.2) If the entry is TYPE B then
            -Output S(L(i,j));
            +If S(L(i,j))=1 then
                  -Add each element in L(i,j) to end of LIS as
TYPE A ;
                  -Remove (i,j) from LIS;
3) Refinement Pass:
For each entry (i,j) in LSP, except those from the last sorting
pass:
      Output the nth most significant bit of c_{i,j};
4) Quantization-Step Update:
Decrement n by 1 and go to step 2.

Conventions used:
O (i,j): set of coordinates of all offsprings of node (i,j).
D (i,j): set of coordinates of all descendant of node (i,j).
H: Set of coordinates of all SOT roots (nodes in lowest sub-band).
L (I,j): D(i,j) – O(i,j).
```

Figure 3-5 Pseudo code of SPIHT

3.2.2 COLOR SET PARTITIONING IN HIERARCHICAL TREE

CSPIHT, which is used for compression of color images, is based on SPIHT with a slight twist to include two additional color spaces. Figure 3-6 shows how CSPIHT forms its SOT.

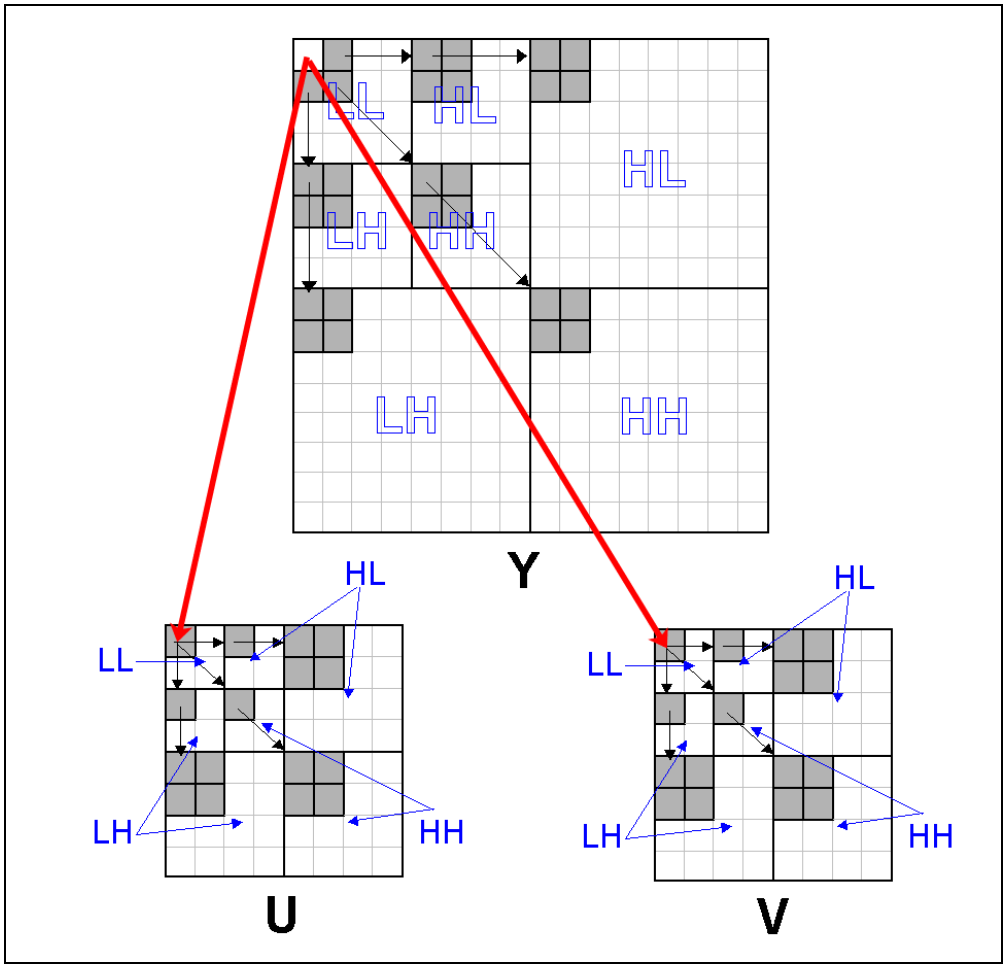


Figure 3-6 Spanning of SOT in CSPIHT

The color spaces used is in YUV format instead of RGB format, which is commonly used for displaying of images. This is due to the high correlations that exist in RGB color space that can be effectively minimized in the YUV color space. Moreover, luminance values and chrominance values of the same spatial location are related in such a way that: if the luminance values exhibit high/low transitions, so do corresponding chrominance values [11]. This implies that if a coefficient in the luminance DWT map is high in values, the corresponding coefficient values in the chrominance DWT map will also be high.

The CSPIHT algorithm [12],[13] exploits this relationship to code color images by creating a comprehensive spatial orientation tree (SOT) that links the each of the SOTs of the YUV (one luminance and two chrominance) planes, as illustrated in Figure 3-6. The linking of Y plane is identical to that used in SPIHT. It is observed that in SPIHT, for each set of 2x2 tree roots in the luminance SOT, there is one without descendants. In the CSPIHT scheme, these nodes without descendants are linked to chrominance tree roots, hence exploiting the interdependency that exists between the transformed luminance and chrominance coefficients. Therefore, by linking the two chrominance tree roots as the offspring of "childless" luminance tree root, a single symbol is able to code the parallel multiple zerotree roots. In this manner, the significant maps of the YUV plane can be coded efficiently together. Such a scheme has an additional advantage of producing embedded streams of luminance and chrominance bits, which allows a color reconstruction of the image to be performed at any termination point in the bit streams.

As shown in Figure 3-6, SOT of CSPIHT links the Y plane to U and V planes such that 1 out of every 4 coefficients on the LL band of Y plane is linked to 1 coefficient on LL bands of U and V planes. Clearly, such a structure requires the LL band of U and V planes to have width and height half of the LL band of Y plane. Every coefficient on LL bands of U and V are then linked to 1 coefficient on the adjacent HL, LH and HH bands. In this case, it is possible for LL band and its adjacent HL, LH and HH bands of U and V planes to have the same odd width and/or height. Such a structure is usually used in YUV 4:2:0 format and when the same number of DWT decomposition level is applied to the 3 planes.

However, if the LL bands of all 3 planes have the same dimensions, then 1 out of every 4 coefficients on LL band of Y plane is linked to 4 coefficients on the LL bands of U and V planes. Such a structure is usually used with YUV 4:4:4 format and when the same number of DWT decomposition level is applied to all 3 planes. It is clear that in this case, all subbands of Y, U and V planes have even width and even height. Alternatively, it is also possible to perform an additional DWT decomposition level on both U and V planes and use the first structure instead.

The algorithm of CSPIHT is exactly the same as SPIHT and the pseudo code of the CSPIHT algorithm is shown in Figure 3-5.

3.2.3 LIMITATION OF SPATIAL ORIENTATION TREE STRUCTURE

Since coefficients in the transform domain are decorrelated, better compression result is expected when more DWT decomposition levels are performed. In general, a minimum of 5 DWT decomposition levels should be used in transform coding.

SPIHT uses SOT structures that require all subbands of the DWT map(s) to have even width and even height. SOT of CSPIHT also requires all subbands of luminance plane to have even width and even height, allowing only the LL bands and the adjacent HL, LH, HH bands of U and V planes to have the same odd width and/or height. Depending on image dimensions, these requirements may impose a limitation on the maximum level of DWT decomposition that can be

performed. For example, to encode a QCIF size (176x144) image or video, only 3 DWT decomposition levels can be performed, while the structure is totally unusable on image or video with an odd width or height.

Due to restriction of SOT structure, both SPIHT and CSPIHT are impractical to be used in real world applications, where images and video come in various dimensions. Two new enhancements, which use *Virtual Level* and *Flexible Spatial Oriental Trees* structure respectively, are proposed to solve this problem, and will be discussed extensively in the subsequent chapters.

4 ENHANCEMENT USING VIRTUAL LEVELS

In this chapter, we propose the use of Virtual Levels as an enhancement to SPIHT and CSPIHT based coding schemes.

4.1 SPIHT USING VIRTUAL LEVELS

Figure 4-1 shows the SOT structure of a DWT map with a Virtual Level (VL) stacked above the LL band. This VL has half the width and half the height of LL, and every ”virtual pixel” in it has four offsprings in LL. Another VL, which has half the width and height of the first VL, can be stacked above the first VL. This can be repeated as long as the last VL has even width and even height. SPIHT-VL is basically a form of the original SPIHT algorithm with the following modifications:

1. During initialization, instead of adding all pixels in LL to LIP and LIS as type A, all pixels in highest VL are added only to the LIS as type A.

2. When a type A set in LIS is found to be significant and its offsprings reside on a VL, then:

    - its offsprings will not be added to LIP;

- it will not be moved to end of LIS as type B, instead, it will be removed from LIS and its four sub-sets will be added to end of LIS as type A.

The above modifications enable the deployment of one or more Virtual Levels (VLs), which can yield additional compression efficiency. Clearly, if only one VL is deployed, the second modification is not necessary as there will be no virtual offspring.

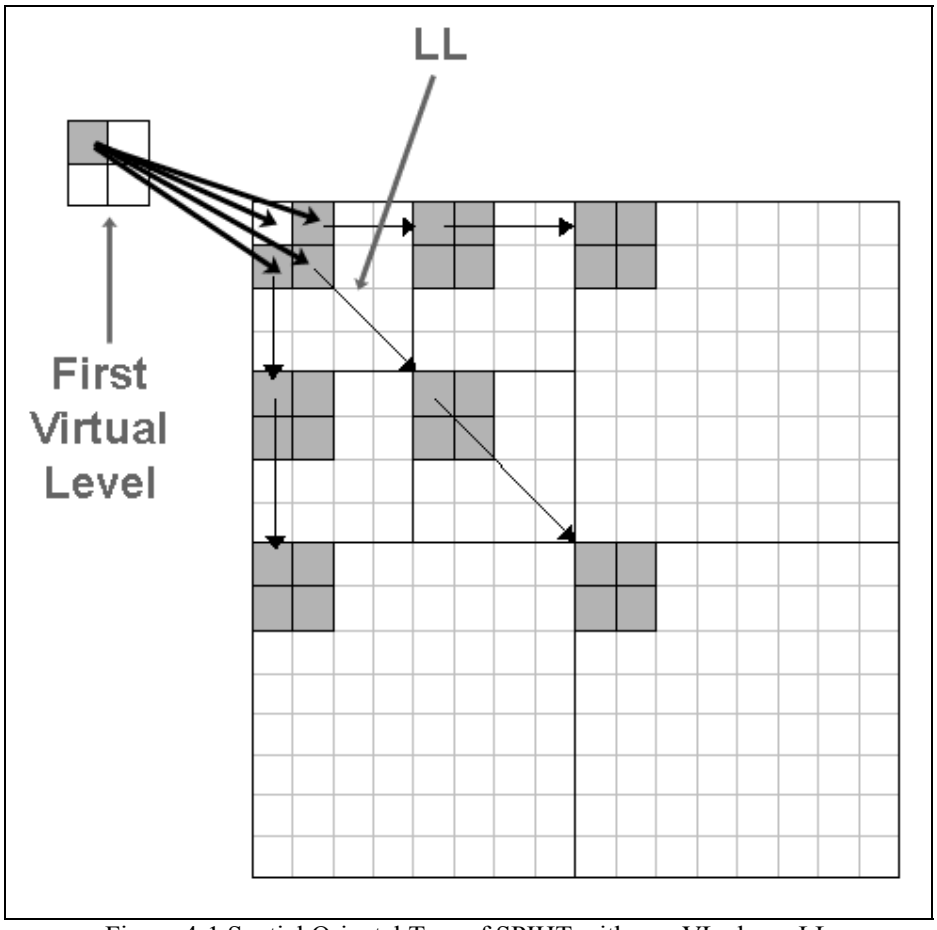


Figure 4-1 Spatial Oriental Tree of SPIHT with one VL above LL

Every virtual pixel on the first Virtual Level has four offsprings in the LL band, as shown in Figure 4-1. Since these four offsprings are arranged in 2x2 blocks, it effectively requires the LL band to have even width and even height. However, such a requirement is acceptable because the original SOT structure already guarantees all subbands to have even dimensions.

There are several advantages of using a Virtual Level. Firstly, SPIHT-VL yields better performance over SPIHT by creating longer branches with VL(s), which reduces the bits required to sort the initial LIP and LIS. Secondly, with longer branches, each initial set in LIS is also able to encapsulate more sets which leads to better compression efficiency. Thirdly, it is worth noting that the initial LIP and LIS sets are disjoint in the original SPIHT algorithm as the pixels that are added to LIP are not contained in the sets added to LIS. Hence the correlation between large coefficients in LL band and the corresponding spatial locations in other bands is not exploited. With the use of Virtual Levels, coefficients on LL band and the corresponding coefficients in higher bands will be grouped under the same set, hence improving compression efficiency.

4.2 CSPIHT USING VIRTUAL LEVELS

CSPIHT Using Virtual Levels or CSPIHT-VL is similar to SPIHT-VL, except that the LL band of luminance band is linked to the LL bands of chrominance bands. Figure 4-2 shows the structure of CSPIHT-VL with one VL. Like SPIHT-VL, this VL has half the width and half the height of LL (of the luminance plane), and every ”virtual pixel” in it has four offsprings in LL, as

shown in Figure 4-2. Another VL, which has half the width and height of the first VL, can be stacked above the first VL. This can be repeated as long as the last VL has even width and even height. CSPIHT-VL is basically a form of the original CSPIHT algorithm with the following modifications:

1. During initialization, instead of adding all pixels in LL to LIP and LIS as type A, all pixels in highest VL are added only to the LIS as type A.

2. When a type A set in LIS is found to be significant and its offsprings reside on a VL, then:

    - its offsprings will not be added to LIP;

    - it will not be moved to end of LIS as type B, instead, it will be removed from LIS and its four sub-sets will be added to end of LIS as type A.

The above modifications enable the deployment of one or more Virtual Levels (VLs), which can yield additional compression efficiency. Clearly, if only one VL is deployed, the second modification is not necessary as there will be no virtual offspring.

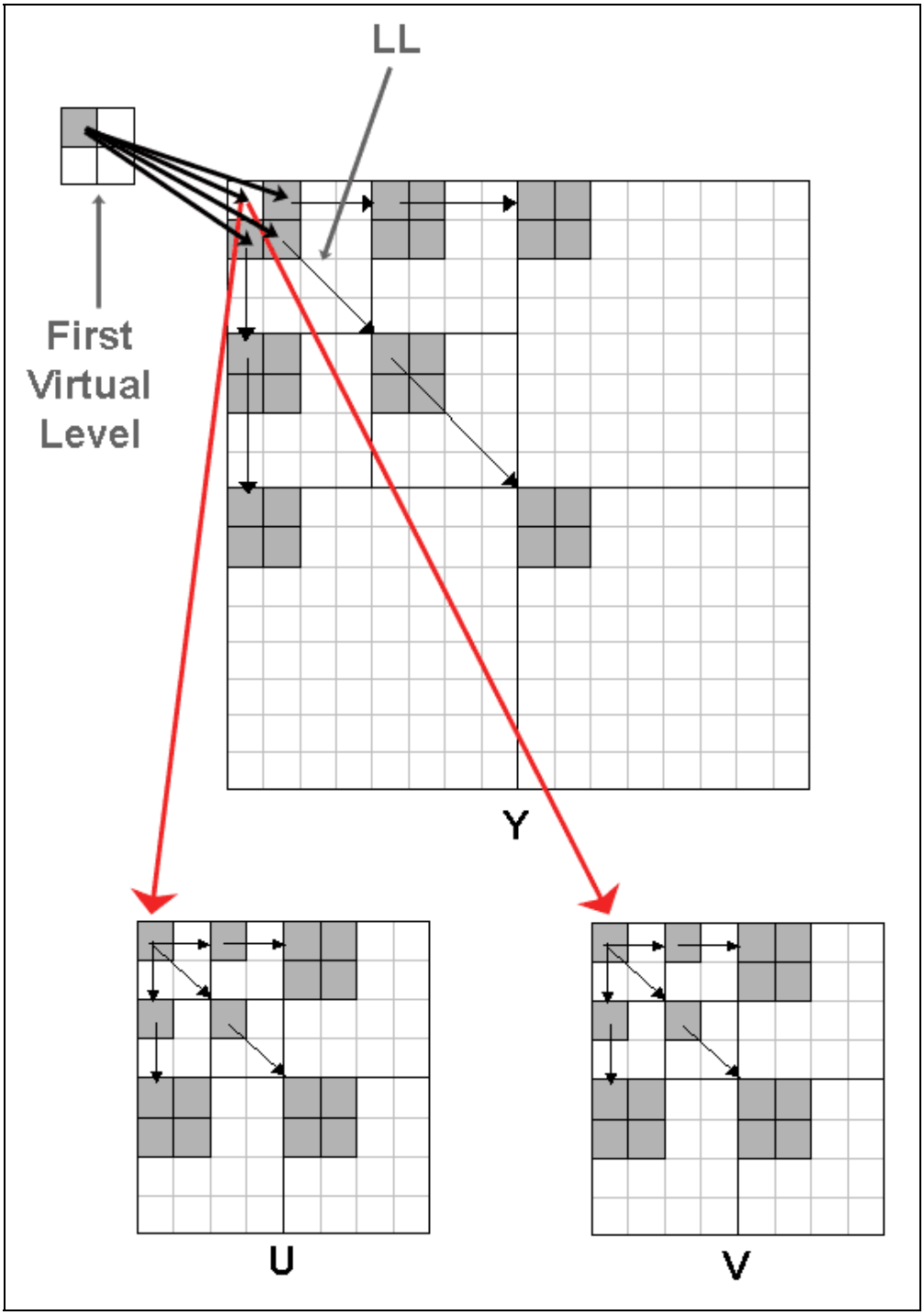


Figure 4-2 Spatial Oriental Tree of CSPIHT with one VL above LL of luminance plane

As the original CSPIHT algorithm requires the LL band of the luminance plane to have even width and even height, at least one VL can be deployed. There are several advantages of using a Virtual Level. Firstly, CSPIHT-VL yields better performance over CSPIHT by creating longer branches with VL(s), which reduces the bits required to sort the initial LIP and LIS. Secondly, with longer branches, each initial set in LIS is also able to encapsulate more sets which leads to better compression efficiency. Thirdly, it is worth noting that the initial LIP and LIS sets are disjoint in the original SPIHT algorithm, hence the correlation between large coefficients in LL band and the corresponding spatial locations in other bands is not exploited. With the use of Virtual Levels, coefficients on LL band and the corresponding coefficients in higher bands will be grouped under the same set, hence improving compression efficiency. In addition, using a VL exploits the correlation between luminance and chrominance bands, which is not exploited in the original SOT for CSPIHT since the initial sets that contain luminance coefficients and chrominance coefficients are disjoint.

5 ENHANCEMENT USING FLEXIBLE SPATIAL ORIENTATION TREE

A main limitation of SOT is its requirement that all subbands must have even width and even height. The use of VL is able to boost compression performance, however, it is unable to overcome this generic limitation of the SOT. A Flexible Spatial Oriental Tree (FSOT) structure is proposed to solve this problem.

Clearly, any 1-D signal that has even length can undergo DWT and the resultant subbands will both have equal length. Similarly, any 2-D signal that has even width and even height can undergo DWT and the resultant 4 subbands will all have equal dimensions. However, complexities arise when we have to perform DWT on an odd length signal. To reduce the complexity, the additional coefficient is always assigned to the scaling coefficients.

5.1 DWT ON 1-D SIGNAL OF ARBITRARY LENGTH

Figure 5-1 shows two possible outcomes of DWT on even length signals, each yielding two even-length subbands and two odd-length subbands respectively. Figure 5-2 shows the two possible outcomes of DWT on odd length signals. These 4 cases form the basis for our analysis in the 2-D case.

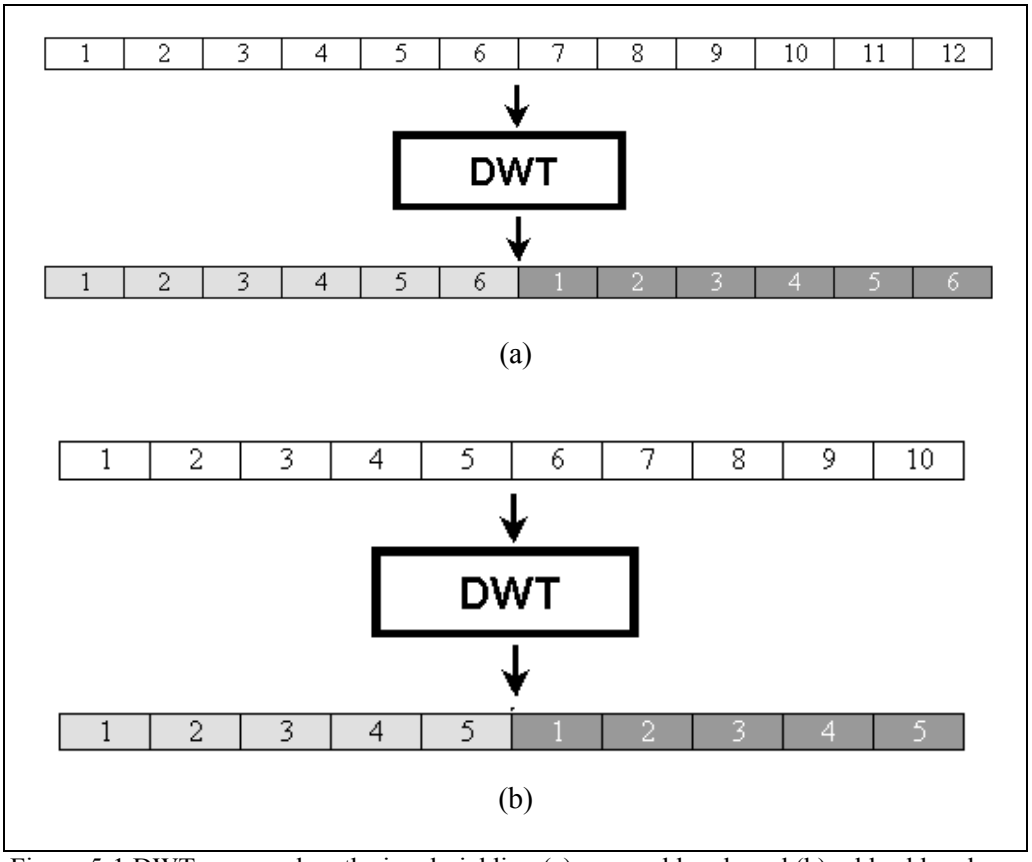


Figure 5-1 DWT on even length signal yielding (a) even subbands and (b) odd subbands

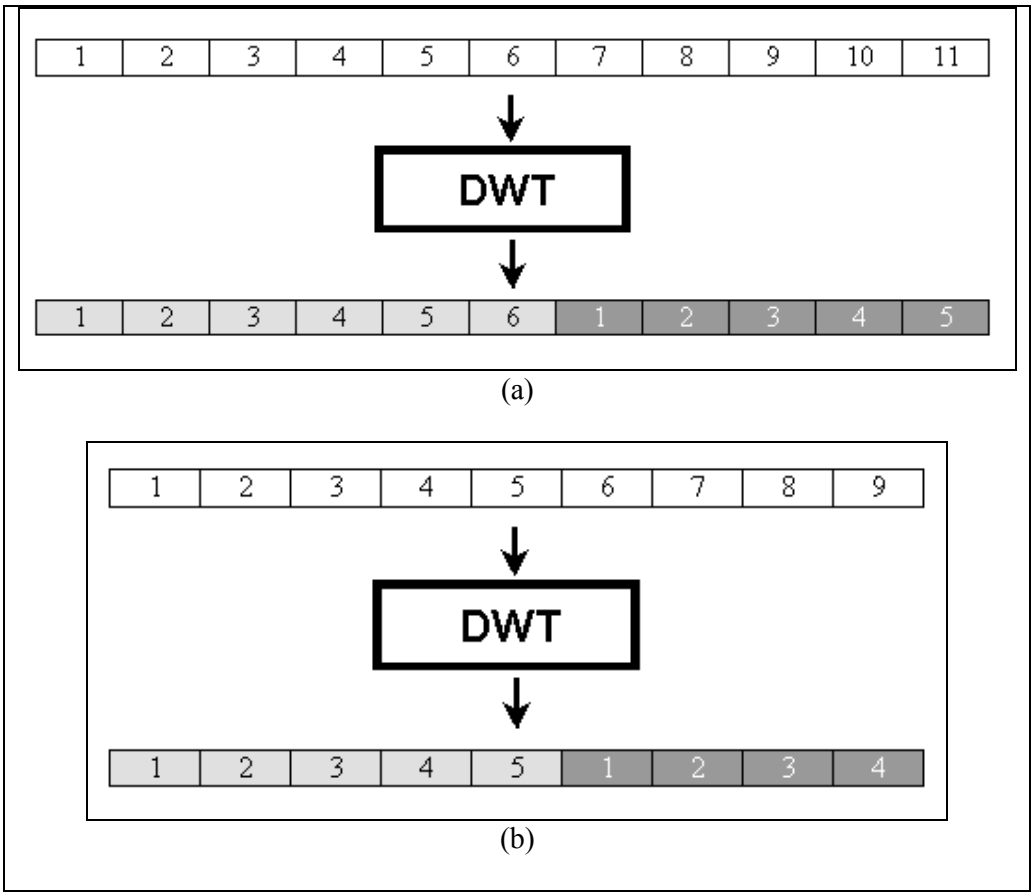


Figure 5-2 DWT on odd length signal yielding (a) even scaling coefficients and odd wavelet coefficients and (b) odd length signal yielding odd scaling coefficients and even wavelet coefficients

In general, it is possible to consider a 1-D signal that undergoes 2 DWT decomposition levels and observe the relationship between the lengths of the subbands. Figure 5-3 shows a 1-D signal of length k decomposed via 2 DWT levels into 3 subbands of length a, b and c respectively.

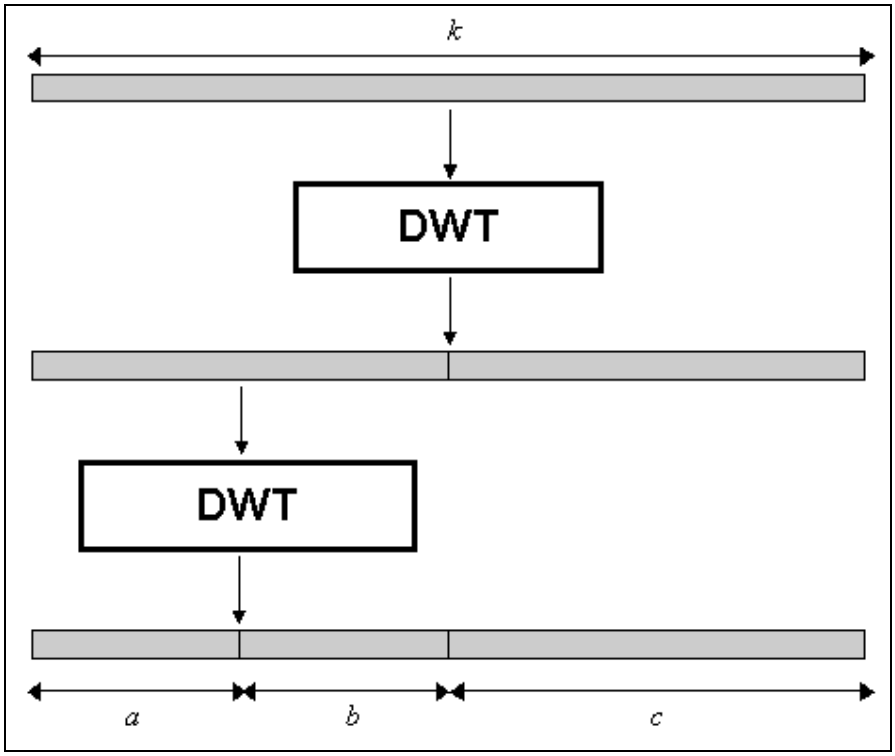


Figure 5-3 1-D signal with 2 levels DWT decomposition

It is important to note that the relationship between *a* and *b* is completely described by the 4 combinations illustrated in Figure 5-1 and Figure 5-2. These 4 combinations also fully describe the relationship between (*a*+*b*) and *c*. Using these two relationships, it is straight forward to evaluate the relationship between *b* and *c* with the help of Table 5-1.

Table 5-1 Corresponding combinations of a, b and c based on value of k

Length				Even/Odd		
k	**a**	**b**	**c**	**a**	**b**	**c**
24	6	6	12	Even	Even	Even
25	7	6	12	Odd	Even	Even
26	7	6	13	Odd	Even	Odd
27	7	7	13	Odd	Odd	Odd
28	7	7	14	Odd	Odd	Even
29	8	7	14	Even	Odd	Even
30	8	7	15	Even	Odd	Odd
31	8	8	15	Even	Even	Odd

Table 5-1 shows a full cycle of combinations of *a*, *b* and *c* using 8 consecutive values for *k*. Clearly, *b* and *c* are related such that:

$$a = b \quad \text{or} \quad a = b + 1 \tag{5.1}$$

$$c = 2b \quad \text{if c is even} \tag{5.2}$$

$$c = 2b \pm 1 \quad \text{if c is odd} \tag{5.3}$$

5.2 DWT ON 2-D SIGNAL WITH ARBITRARY DIMENSION

Using the four cases in 1-D signal DWT as a basis, it is now possible to examine and categorize different subband compositions of 2-D signals. Figure 5-4 shows the dimensions of a 2-D DWT map with 1 level decomposition and Table 5-3 shows the relationships between X1, X2, Y1 and Y2.

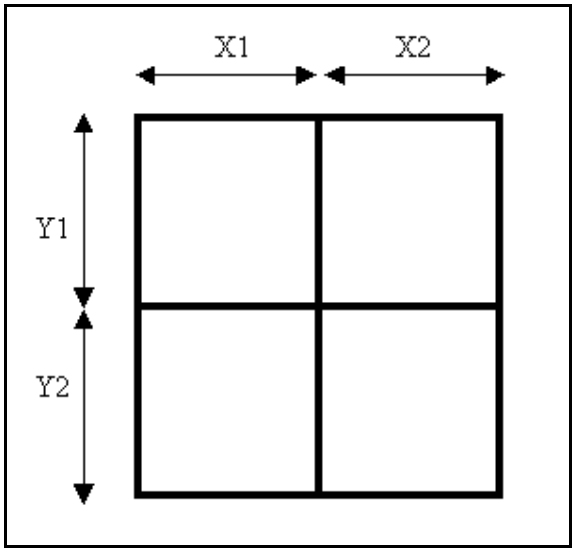


Figure 5-4 DWT map showing 1 level of DWT decompostiton

Table 5-2 Relationship between dimensions

X1	X2	Y1	Y2	Relation between X1 and X2	Relation between Y1 and Y2
Even	Even	Even	Even	X1=X2	Y1=Y2
Odd	Odd	Even	Even	X1=X2	Y1=Y2
Even	Odd	Even	Even	X1=X2+1	Y1=Y2
Odd	Even	Even	Even	X1=X2+1	Y1=Y2
Odd	Odd	Odd	Odd	X1=X2	Y1=Y2
Even	Odd	Odd	Odd	X1=X2+1	Y1=Y2
Odd	Even	Odd	Odd	X1=X2+1	Y1=Y2
Even	Odd	Even	Odd	X1=X2+1	Y1=Y2+1
Odd	Even	Even	Odd	X1=X2+1	Y1=Y2+1
Odd	Even	Odd	Even	X1=X2+1	Y1=Y2+1

5.3 SPIHT USING FSOT

Figure 5-5 shows the SOT structure of SPIHT. The structure starts from the LL band, where every 2x2 block will have 1 coefficient without any offspring, 1 coefficient with 2x2 offsprings in the LH^1 band, 1 coefficient with 2x2 offsprings in the HL^1 band and 1 coefficient with 2x2 offsprings in the HH^1 band. Each coefficient in LH^1, HL^1 and HH^1 also has 2x2 offsprings, which correspond to its spatial location and reside in LH^2, HL^2 and HH^2 respectively. This relationship will be repeated up to the highest subbands, where all coefficients have no offspring. Since all subbands are partitioned into 2x2 blocks, the structure can only be used if all subbands have even width and even height.

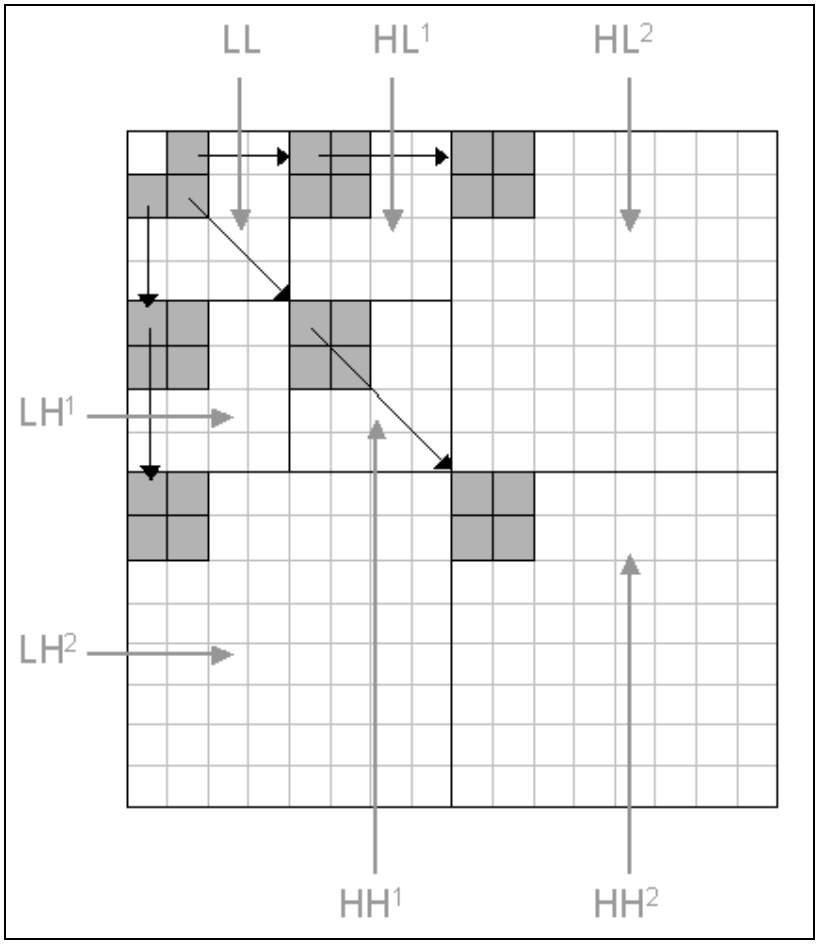


Figure 5-5 SOT of SPIHT

Depending on the image dimensions, such a requirement limits the maximum DWT decomposition levels that can be performed and hence degrades the performance of SPIHT. To overcome this constraint, a Flexible Spatial Orientation Tree (FSOT) structure can be used.

The FSOT structure is based on SPIHT's original SOT structure and makes use of the following additional rule sets.

1. **Linking from LL to HL^1, LH^1, HH^1:**
    a. For a LL band that has odd width/height, all pixels on the last column/row will have no offspring.
    b. If (width of LL > width of HL^1), then all pixels on the last column of LL will have no offspring.
    c. If (height of LL > height of LH^1), then all pixels on the last row of LL will have no offspring.
    d. For each LH^1, HL^1 and HH^1 band that has odd width, a pixel on last column will have the same ascendant as the pixel to its left.
    e. For each LH^1, HL^1 and HH^1 band that has odd height, a pixel on last row will have the same ascendant as the pixel above.
2. **Linking from parent-subband HL^x, LH^x, HH^x to child-subband HL^{x+1}, LH^{x+1}, HH^{x+1}:**
    a. If (parent-subband-width*2 > child-subband-width) then last column of parent-subband will have no offspring.
    b. If (parent-subband-height*2 > child-subband-height) then last row of parent-subband will have no offspring.
    c. For a child-subband that has odd width, a pixel on last column will have the same ascendant as the pixel to its left.
    d. For a child-subband that has odd height, a pixel on last row will have the same ascendant as the pixel above.

Figure 5-6 Rule sets for FSOT

Although relationships 5.1, 5.2 and 5.3 are derived for the 1-D case, it can be easily extended to the 2-D case. Rule set 1 is obtained by observing relationship 5.1, Figure 5-4 and Table 5-2. Rule set 2 is obtained by analyzing Table 5-1 and relationships 5.2 and 5.3.

As an illustration, consider the case of encoding a QCIF size (176x144) image. Only 3 DWT decomposition levels can be performed if SPIHT's original SOT is used, and the resultant LL band dimension is 22x18. FSOT is able to overcome this limitation and allows more DWT decomposition levels to be performed. Figure 5-7 shows a FSOT structure of the top-left 22x18 coefficients if 5 DWT decomposition levels are performed.

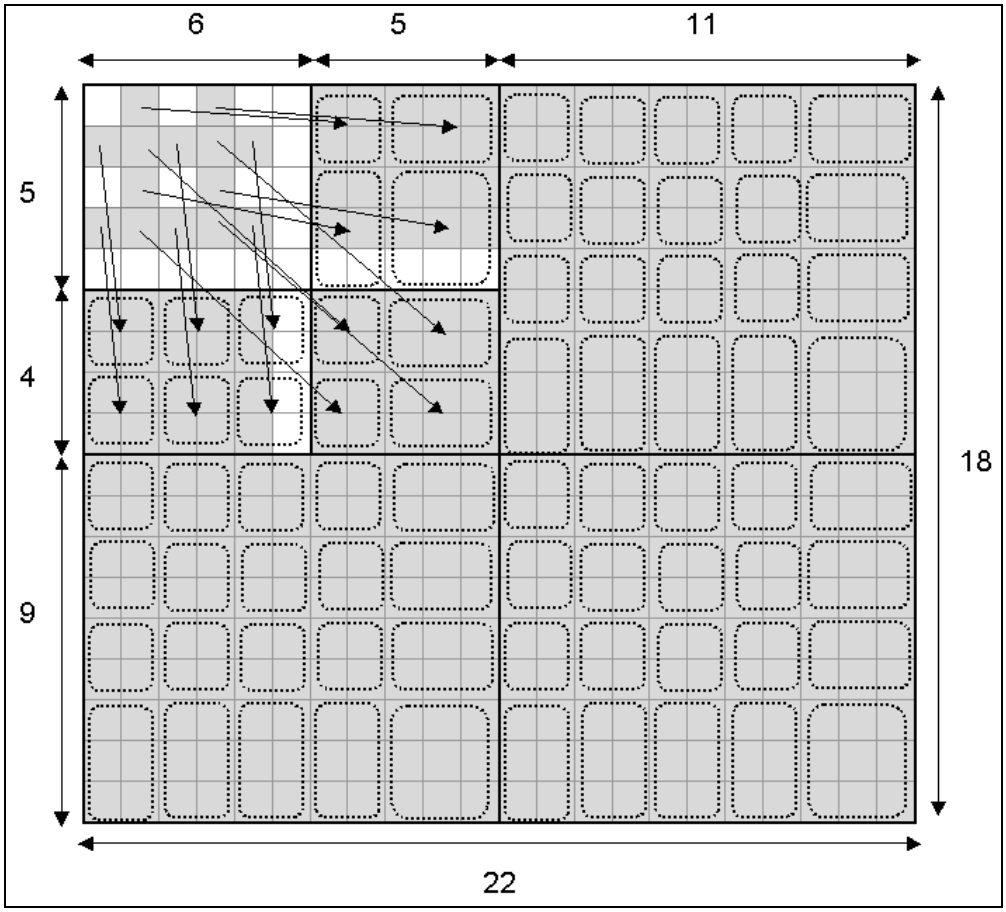


Figure 5-7 FSOT of QCIF size image showing the top-left 22x18 coefficients. White coefficients denote coefficients without offspring and dotted rectangle denotes group of coefficients with the same parent.

5.4 CSPIHT USING FSOT

CSPIHT using FSOT, or CSPIHT-FSOT, can be used to compress color images, which are usually compressed either in YUV 4:2:0 format or YUV 4:4:4 format, both comprising of 3 color planes of Y, U and V. These color planes are first separately transformed up to some desired levels of DWT decomposition, and then a FSOT is used to link up each DWT map individually. The 3 DWT maps now need to be linked up, like in the case of CSPIHT. Depending on image format

and levels of DWT decomposition performed on each color planes, LL subbands of the color planes may have different or same dimensions, giving rise to different scenarios.

For a YUV 4:2:0 image with 5 levels of DWT decomposition on each color planes, the LL subbands of U and V planes will have width and height half (rounded up) of LL subband of Y plane (scenario 1). For a YUV 4:4:4 image with 5 levels of DWT decomposition on each color planes, the LL subbands of all color planes will all have the same dimensions (scenario 2), however, if one additional level of DWT decomposition is performed on both U and V planes, then the LL subbands of U and V planes will have width and height half (rounded up) of LL subband of Y plane (scenario 1).

As shown in Figure 3-6, the original SOT of CSPIHT is only designed to handle scenario 1, which is acceptable because we can always perform one more DWT decomposition level on U and V plane to change from scenario 2 to scenario 1. Hence by using FSOT on each color plane, a new rule set is also designed to link up LL subband of Y plane to LL subbands of U and V planes for scenario 1 only. This new rule set is shown in Figure 5-8.

3. Linking from LL of Y plane to LL of U and V planes :

a. If (width-of-LL-of-Y < width-of-LL-of-U-and-V * 2), then a coefficient on the last column of LL of U and V will have same parent as coefficient on the left.
b. If (height-of-LL-of-Y < height-of-LL-of-U-and-V * 2), then a coefficient on the last row of LL of U and V will have same parent as coefficient on top.

Figure 5-8 Rule set to link up FSOT of different color planes

As an illustration, consider performing 5 DWT decomposition levels on a QCIF size (176x144) color image with YUV 4:2:0 color format. Figure 5-11 shows the FSOT structure that links up the 3 color planes utilizing all 3 rule sets. To better magnify the DWT maps, only the top-left 22x18 coefficients of each color plane are displayed.

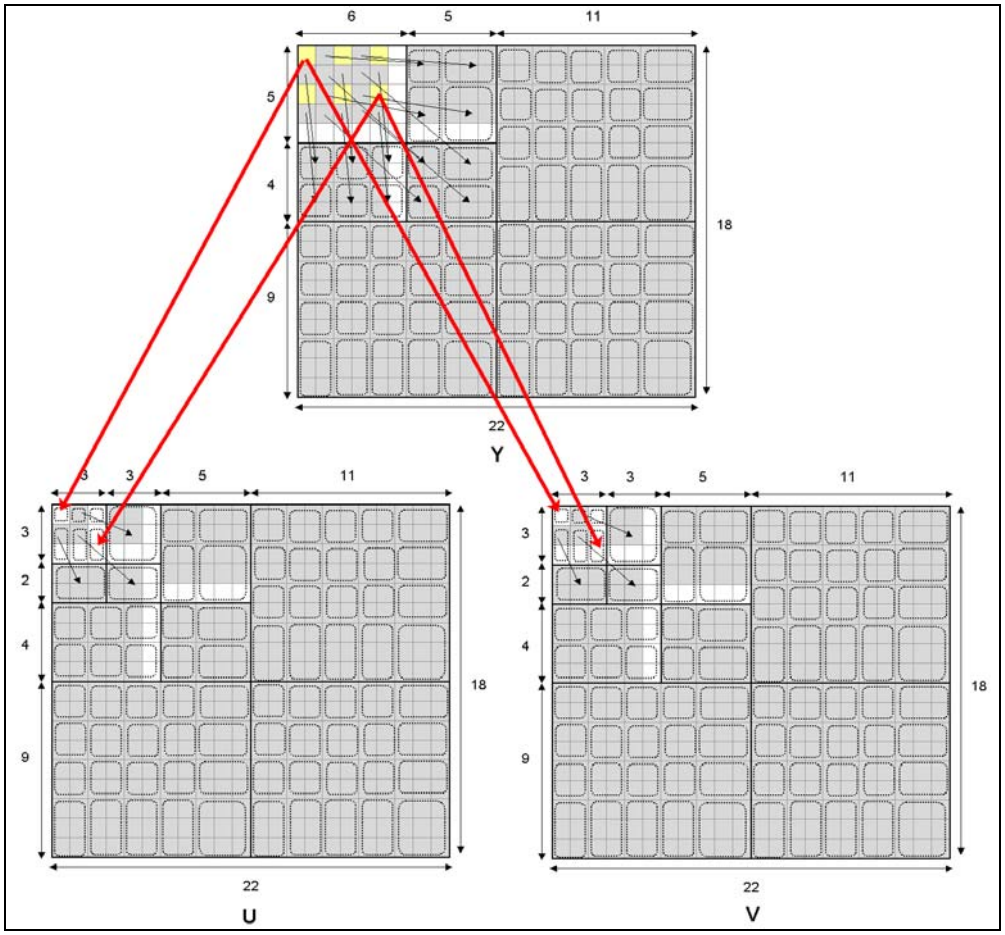


Figure 5-9 FSOT of QCIF size color image (YUV 4:2:0 format) showing the top-left 22x18 coefficients of each color plane. White coefficients denote coefficients without offspring and dotted rectangle denotes group of coefficients with the same parent. Yellow coefficients on LL band of Y plane denote coefficients that have offspring in U and V planes.

6 IMAGE COMPRESSION PERFORMANCE

In this chapter, we evaluate the image compression performance of the SPIHT and CSPIHT enhancements proposed in Chapter 4 and Chapter 5. Section 6.1 shows the comparative improvements of the two enhancements, while section 6.2 presents the comparisons between SPIHT-FSOT and JPEG2000.

6.1 NUMERICAL RESULTS

In this section we evaluate the performance improvements that are obtained by using VL and FSOT. To provide a comprehensive comparison of performance, 3 sets of images with different dimensions will be used. The first set comprises of standard 512×512 test images like Lena and Barbara, the second set comprises of CIF sized (352x288) images while the last set comprises of QCIF sized (176x144) images. In our work, the Daubechies 9/7-tap biorthogonal filter bank [14] and whole-sample symmetric extension [15] are used to perform DWT in obtaining the results, which are binary coded without using arithmetic coding.

6.1.1 COMPRESSION RESULTS OF STANDARD 512X512 TEST IMAGES

Table 6-1 shows the result of compressing monochrome 512×512 Lena and Barbara with 8 bits/pixel using 5 DWT decomposition levels. SPIHT and SPIHT-FSOT both produce the same compression result, this is because all

subbands even width and even height, hence the structures of SOT and FSOT in this case are exactly the same. Compression efficiency of SPIHT-VL is only marginally better than SPIHT and the improvement, approximately 0.1 dB at low bit rate and 0.01 dB at high bit rate, is rather insignificant. Comparing SPIHT-VL using 1 VL, 2 VL and 3 VL, it is also clear that very little improvement can be gained by deploying more VL, and in some cases the result may worsen. Figure 6-1 and Figure 6-2 show the compressed images.

Table 6-1 Results of compressing monochrome 512x512 images with 8 bits/pixel.

	PSNR (dB)				
		SPIHT-VL			SPIHT-FSOT
bpp	SPIHT	1 VL	2 VL	3 VL	
Lena					
0.0625	27.752	27.858	27.858	27.856	27.752
0.125	30.591	30.639	30.639	30.639	30.591
0.250	33.629	33.657	33.656	33.656	33.629
0.5	36.778	36.794	36.794	36.794	36.778
1.0	39.918	39.925	39.925	39.925	39.918
Barbara					
0.0625	23.067	23.099	23.099	23.099	23.067
0.125	24.400	24.460	24.459	24.459	24.400
0.250	27.062	27.082	27.081	27.081	27.062
0.5	30.829	30.847	30.847	30.847	30.829
1.0	35.791	35.801	35.801	35.801	35.791

Figure 6-1 Lena compressed by SPIHT and SPIHT-FSOT at 0.5 bpp

Figure 6-2 Lena compressed by SPIHT-VL with 1 VL at 0.5 bpp

Table 6-2 shows the result of compressing color Lena and Barbara with 24 bits/pixel (4:4:4 format) using 5 DWT decomposition levels for luminance plane and 6 DWT decomposition levels for chrominance color planes. We use one extra

DWT decomposition level for chrominance planes to link up the LL bands as explained in Section 5.4. CSPIHT and CSPIHT-FSOT yield different results due to differences in the structures deployed, which are shown in Figure 3-6 and Figure 5-9 respectively. The differences, however, are extremely marginal. CSPIHT-VL also shows very slight improvement over CSPIHT, which is consistent with our observation between SPIHT-VL and SPIHT. Similarly, it is shown that using more VL does not yield any significant improvement in compression efficiency. Figure 6-3, Figure 6-4 and Figure 6-5 show the compressed images.

Table 6-2 Results of compressing color 512x512 images with 24 bits/pixel.

bpp	Color plane	PSNR (dB)				
		CSPIHT	CSPIHT-VL			CSPIHT-FSOT
			1 VL	2 VL	3 VL	
Lena						
0.0625	Y	28.258	28.361	28.366	28.366	28.282
	U	34.291	34.366	34.366	34.366	34.253
	V	34.392	34.472	34.472	34.472	34.529
0.125	Y	30.889	30.937	30.937	30.937	30.906
	U	36.086	36.105	36.105	36.105	36.060
	V	36.058	36.121	36.127	36.127	36.079
0.25	Y	33.666	33.705	33.706	33.706	33.672
	U	37.750	37.761	37.761	37.761	37.759
	V	37.613	37.623	37.623	37.623	37.624
0.5	Y	36.432	36.450	36.451	36.451	36.437
	U	39.269	39.275	39.275	39.275	39.271
	V	39.234	39.247	39.246	39.247	39.238
1.0	Y	38.858	38.858	38.858	38.858	38.858
	U	40.654	40.661	40.662	40.662	40.654
	V	40.834	40.855	40.855	40.855	40.836
Barbara						
0.0625	Y	24.500	24.541	24.542	24.542	24.516
	U	35.688	35.746	35.746	35.746	35.792
	V	35.078	35.125	35.125	35.125	35.072
0.125	Y	25.833	25.905	25.907	25.908	25.847
	U	36.531	36.531	36.531	36.531	36.537
	V	36.289	36.289	36.289	36.289	36.289

0.25	Y	28.211	28.301	28.303	28.303	28.212
	U	37.452	37.452	37.452	37.452	37.461
	V	37.566	37.566	37.566	37.566	37.581
0.5	Y	32.068	32.088	32.088	32.088	32.071
	U	39.098	39.104	39.104	39.104	39.100
	V	39.324	39.225	39.225	39.345	39.337
1.0	Y	36.572	36.591	36.592	36.592	36.575
	U	40.808	40.810	40.810	40.810	40.808
	V	41.075	41.348	41.348	41.348	41.347

Figure 6-3 Lena compressed by CSPIHT at 0.5 bpp

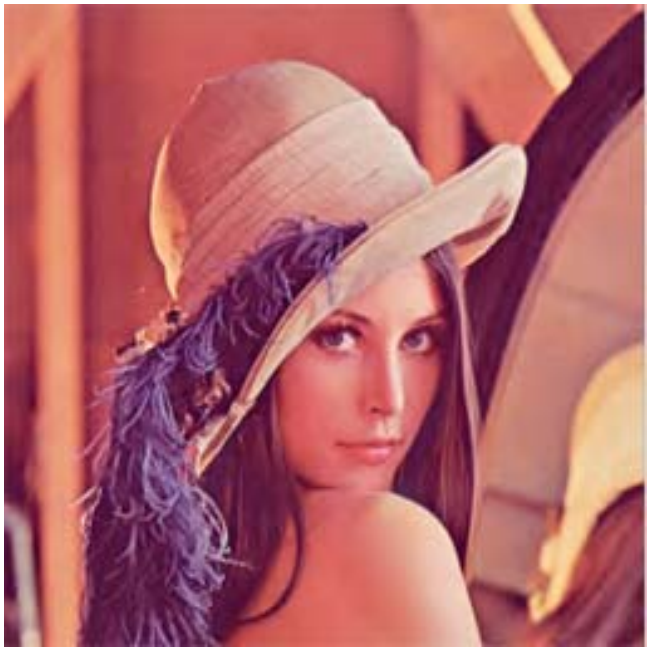


Figure 6-4 Lena compressed by CSPIHT-VL with 1 VL at 0.5 bpp

Figure 6-5 Lena compressed by CSPIHT-FSOT at 0.5 bpp

6.1.2 COMPRESSION RESULTS OF CIF SIZED (352X288) IMAGES

Table 6-3 shows the result of compressing luminance data of the first frame of paris.cif and tempete.cif, using 4 DWT decomposition levels for SPIHT and SPIHT-VL, and 5 DWT decomposition levels for SPIHT-FSOT. Less DWT decomposition levels are used for SPIHT and SPIHT-VL due to the constraint of image dimension. Only one VL is used for SPIHT-VL because the first VL has dimension 11x9, which prevents more VL from being deployed.

It is observed that SPIHT-VL outperforms SPIHT very marginally at low bit rates with approximately 0.15 dB improvement, which diminishes to 0.07 dB at high rates. The performance of SPIHT-FSOT is always higher than that of SPIHT-VL, but the magnitude of improvement is so small that they are almost identical. Figure 6-6, Figure 6-7 and Figure 6-8 show the compressed images.

Table 6-3 Results of compressing luminance data of CIF images (352x288).

	PSNR (dB)		
	SPIHT	SPIHT-VL (1 VL)	SPIHT-FSOT
bpp	4 DWT decomposition levels	4 DWT decomposition levels	5 DWT decomposition levels
First frame of paris.cif			
0.0625	19.491	19.644	19.669
0.125	21.185	21.302	21.327
0.250	23.305	23.397	23.408
0.5	27.036	27.113	27.108
1.0	32.459	32.520	32.528
First frame of tempete.cif			
0.0625	21.303	21.486	21.532
0.125	22.874	22.968	22.978
0.250	24.861	24.978	24.994
0.5	27.561	27.637	27.642

1.0	31.919	31.983	31.999

Figure 6-6 First frame of paris.cif compressed by SPIHT at 0.5 bpp

Figure 6-7 First frame of paris.cif compressed by SPIHT-VL with 1 VL at 0.5 bpp

Figure 6-8 First frame of paris.cif compressed by SPIHT-FSOT at 0.5 bpp

Table 6-4 shows the result of compressing first frame of paris.cif and tempete.cif, whose format are 4:2:0. 4 DWT decomposition levels are used in transforming the 3 color planes for CSPIHT and CSPIHT-VL, while 5 DWT decomposition levels are used in the case of CSPIHT-FSOT. Less DWT decomposition levels are used for CSPIHT and CSPIHT-VL due to the constraint of image dimension. Only one VL is used for CSPIHT-VL because the first VL has dimension 11x9, which prevents more VL from being deployed.

In general, CSPIHT-VL is marginally better than CSPIHT and the improvement diminishes as bit rate increases. CSPIHT-FSOT outperforms CSPIHT-VL but the differences are so minute that the compression results are almost identical. These observations are consistent with observations on compression of luminance data only. Figure 6-9, Figure 6-10 and Figure 6-11 show the compressed images.

Table 6-4 Results of compressing CIF images (352x288) with 4:2:0 format.

bpp	Color plane	PSNR (dB)		
		CSPIHT	CSPIHT-VL (1 VL)	CSPIHT-FSOT
		4 DWT decomposition levels	4 DWT decomposition levels	5 DWT decomposition levels
First frame of paris.cif				
0.0625	Y	19.232	19.457	19.515
	U	26.617	25.743	26.399
	V	27.340	27.259	27.797
0.125	Y	20.823	21.073	21.121
	U	27.447	27.716	27.920
	V	28.021	28.019	28.481
0.25	Y	22.931	23.093	23.110
	U	29.002	29.056	29.524
	V	29.626	29.626	29.944
0.5	Y	26.516	26.612	26.630
	U	31.321	31.439	31.546
	V	31.867	31.867	32.047

1.0	Y	31.558	31.633	31.640
	U	34.812	34.842	34.919
	V	35.220	35.241	35.406
First frame of tempete.cif				
0.0625	Y	20.946	21.235	21.359
	U	26.382	26.838	27.684
	V	29.785	29.846	30.514
0.125	Y	22.592	22.739	22.780
	U	28.708	28.624	29.186
	V	31.752	31.718	32.178
0.25	Y	24.433	24.605	24.682
	U	29.899	30.138	30.287
	V	32.997	33.357	33.760
0.5	Y	27.119	27.208	27.242
	U	32.003	32.110	32.268
	V	35.160	35.189	35.145
1.0	Y	31.242	31.305	31.329
	U	34.451	34.465	34.655
	V	37.072	37.076	37.212

Figure 6-9 First frame of paris.cif compressed by CSPIHT at 0.5 bpp

Figure 6-10 First frame of paris.cif compressed by CSPIHT-VL at 0.5 bpp

Figure 6-11 First frame of paris.cif compressed by CSPIHT-FSOT at 0.5 bpp

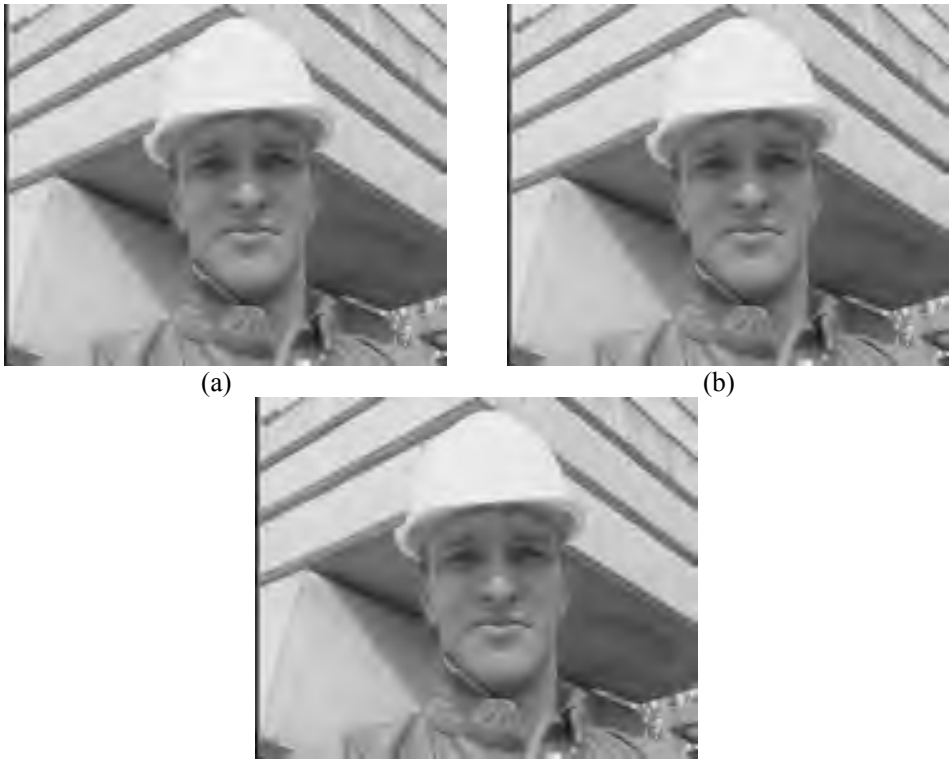
6.1.3 COMPRESSION RESULTS OF QCIF SIZED (176X144) IMAGES

Table 6-5 shows the result of compressing luminance data of first frame of foreman.qcif and suzie.qcif, using 3 DWT decomposition levels for SPIHT and SPIHT-VL, and 5 DWT decomposition levels for SPIHT-FSOT. Less DWT decomposition levels are used for SPIHT and SPIHT-VL due to the constraint of image dimension. Only one VL is used for SPIHT-VL because the first VL has dimension 11x9, which prevents more VL from being deployed.

It is observed that SPIHT-VL outperforms SPIHT significantly at low bit rate with approximately 2-4 dB improvement, which diminishes to around 0.15-0.4 dB at high bit rate. The performance of SPIHT-FSOT is always higher than that of SPIHT-VL, and the magnitude of improvement is approximately 1.2 dB at low bit rate and diminishes to around 0.1 dB at high bit rate. Figure 6-12 shows the compressed images.

Table 6-5 Results of compressing luminance data of QCIF images (176x144).

bpp	PSNR (dB)		
	SPIHT	SPIHT-VL (1 VL)	SPIHT-FSOT
First frame of foreman.qcif			
0.0625	19.757	21.677	22.889
0.125	24.012	24.715	25.322
0.250	27.324	27.617	27.853
0.5	30.831	31.071	31.151
1.0	35.921	36.069	36.141
First frame of suzie.qcif			
0.0625	22.717	26.662	27.890
0.125	27.637	29.960	30.488
0.250	31.695	33.006	33.262
0.5	35.869	36.505	36.695
1.0	40.661	41.025	41.158



(a) (b)

(c)

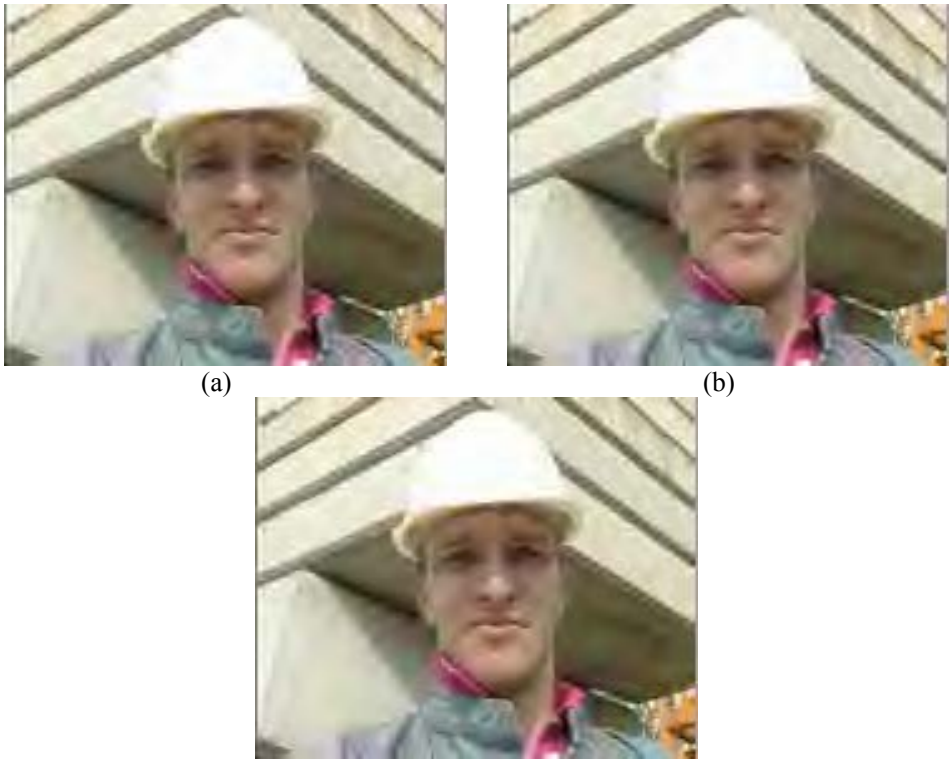
Figure 6-12 First frame of foreman.qcif compressed at 0.5 bpp by (a) SPIHT, (b) SPIHT-VL with 1 VL and (c) SPIHT-FSOT

Table 6-6 shows the result of compressing first frame of foreman.qcif and suzie.qcif, whose format are 4:2:0. 3 DWT decomposition levels are used in transforming the 3 color planes for CSPIHT and CSPIHT-VL, while 5 DWT decomposition levels are used in the case of CSPIHT-FSOT. Less DWT decomposition levels are used for CSPIHT and CSPIHT-VL due to constraint of image dimension. Only one VL is used for CSPIHT-VL because the first VL has dimension 11x9, which prevents more VL from being deployed.

CSPIHT-VL compresses much better than CSPIHT and the improvement diminishes as bit rate increases. CSPIHT-FSOT also outperforms CSPIHT-VL significantly at low bit rates. These observations are consistent with observations on compression of luminance data only. Figure 6-13 shows the compressed images.

Table 6-6 Results of compressing QCIF images (176x144) with 4:2:0 format.

Bpp	Color plane	PSNR (dB)		
		CSPIHT	CSPIHT-VL (1 VL)	CSPIHT-FSOT
First frame of foreman.qcif				
0.0625	Y	19.643	21.666	22.775
	U	26.995	26.995	32.440
	V	28.938	29.389	30.585
0.125	Y	23.412	24.502	25.099
	U	27.453	28.422	33.144
	V	28.866	30.818	31.981
0.25	Y	26.777	27.299	27.656
	U	33.602	33.674	34.355
	V	33.355	34.243	34.970
0.5	Y	30.167	30.544	30.841
	U	36.511	36.542	36.519
	V	37.308	37.302	37.470
1.0	Y	35.044	35.289	35.482
	U	38.603	38.667	38.785
	V	40.556	40.694	41.076
First frame of suzie.qcif				
0.0625	Y	22.493	26.423	27.662
	U	27.555	28.146	36.790
	V	28.434	29.441	35.757
0.125	Y	26.915	29.094	30.281
	U	28.146	35.089	37.941
	V	29.441	35.328	37.112
0.25	Y	30.834	32.217	33.024
	U	38.926	39.752	40.378
	V	38.154	38.584	39.188
0.5	Y	35.016	35.763	36.310
	U	41.975	43.006	42.554
	V	40.703	41.569	41.571
1.0	Y	39.736	40.245	40.563
	U	45.049	45.091	45.244
	V	44.918	44.916	45.132



(a) (b)

(c)

Figure 6-13 First frame of foreman.qcif compressed at 0.5 bpp by (a) CSPIHT, (b) CSPIHT-VL with 1 VL and (c) CSPIHT-FSOT

6.1.4 ANALYSIS

SPIHT-VL yields better performance over SPIHT by creating longer branches with VL, which reduces the bits required to sort the initial LIP and LIS. With longer branches, each initial set in LIS is also able to encapsulate more sets which leads to better compression efficiency.

The results show a strong correlation between the improvement of SPIHT-VL over SPIHT and the DWT decomposition levels performed. More improvement can be achieved if less DWT decomposition levels are performed, as

evident from Tables 6-1, 6-3 and 6-5. This correlation exists because the *ratio of LL band size to image size* is significantly higher when less DWT decomposition levels are performed. With 5 DWT decomposition levels, the ratio is 1:1024; with 3 DWT decomposition levels, the ratio is 1:64. Since SPIHT-VL achieves additional compression efficiencies by encapsulating the LL band, bits savings of SPIHT-VL is expected to be approximately proportional to the size of LL band.

Results in Table 6-1 also show that deploying more VL yields very little improvement. This is expected because the first VL exploits the following characteristics: (1) clustering of large coefficients within LL band and (2) large coefficients in adjacent LH, HL and HH bands corresponds to location of large coefficients in LL band. When an additional VL is deployed above the first VL, only the first characteristic is exploited. For example, if two VL are deployed, it is analogous to examining the LL band in blocks of 4x4: when a set on the second VL is significant, it will then be broken up into 4 sets of 2x2, which causes additional overhead with no direct PSNR contribution in encoding such information. In contrast, when a set on the first VL is significant, coefficients on the LL band are checked for significance, which improve PSNR. In essence, the advantage of having more than 1 VL is negligible and it is possible for compression performance to decline as more VL are deployed.

CSPIHT-VL is a direct extension of SPIHT-VL for compressing color images. The improvement of CSPIHT-VL over CSPIHT is similar to that of SPIHT-VL over SPIHT for the same reasons mentioned above.

SPIHT-FSOT yields better performance over SPIHT by overcoming the limitation of image dimensions and allows more DWT decomposition levels to be performed. There is a direct relationship between improvement of SPIHT-FSOT over SPIHT and the additional DWT decomposition levels that SPIHT-FSOT can perform over SPIHT. If the number of addition DWT decomposition levels is high, a significant improvement is also observed.

CSPIHT-FSOT is an extension of SPIHT-FSOT to encode color images, hence improvements of CSPIHT-FSOT over CSPIHT are observed to be similar to that of SPIHT-FSOT over SPIHT.

In essence, using VL allows the exploitation of additional compression efficiency in situations where less DWT levels are performed. Performing more levels of DWT decomposition allows better exploitation of correlation between subbands and improves compression performance. In general, 5 levels of DWT decomposition are normally performed, as additional levels after 5 usually yield insignificant improvement. In this respect, it is clear that FSOT is superior to VL such that FSOT actually enables more DWT decomposition levels to be performed, which can be clearly observed from the results in Tables 6-1 through 6-6.

6.2 COMPARING SPIHT-FSOT WITH JPEG2000

In Section 6.1, it is clear that SPIHT-FSOT has the best performance and is able to encode image of arbitrary dimensions. In this section, we will compare SPIHT-FSOT with JPEG2000 as one of its core design consideration is that it must be able to encode image with arbitrary dimensions. In sub-section 6.2.1 the numerical results is presented, followed by an analysis in section 6.2.2.

6.2.1 NUMERICAL RESULTS

Our results are obtained using non-standard images that are smaller in dimensions than the standard 512×512 images like Lena and Barbara. The Daubechies 9/7-tap biorthogonal filter bank [14] and whole-sample symmetric extension [15] are used to perform DWT in our experiments. The JPEG2000 compression results are obtained using the jj2000 version 5.1 Java software available at http://www.jj2000.org/.

Table 6.7 shows the compression performance on the luminance data of the *first frame* of various QCIF (176x144) video sequences. Compression results of SPIHT-FSOT using five DWT decomposition levels are compared with the compression results of JPEG2000 using five DWT decomposition levels.

Table 6.8 shows compression performance on luminance data of the *first frame* of various CIF (352x288) video sequences. Compression results of SPIHT-

FSOT using five DWT decomposition levels are compared with the compression results of JPEG2000 using five DWT decomposition levels.

To ensure a fair comparison, we compare SPIHT-FSOT using arithmetic coding to JPEG2000, which draws much of its compression efficiencies from its tightly integrated arithmetic coder[3][4]. Using FSOT, coefficients cluster in block sizes of 2x2, 3x2, 2x3 and 3x3, hence the arithmetic coder used for SPIHT-FSOT is different from that of SPIHT. The arithmetic coder used with the original SPIHT uses several adaptive models, each with 2^m symbols, $m \in \{1,2,3,4\}$, to code the information in groups of four pixels [5]. For the arithmetic coder of SPIHT-FSOT, we use more adaptive models that contain up to 2^n symbols, $n \in \{1,2,3,4,5,6,7,8,9\}$, which can code information for group sizes of up to 9 coefficients. Figure 6-14 to Figure 6-19 show the compressed images.

Table 6-7 Compression comparison of SPIHT-FSOT and JPEG200 on CIF size images

bpp	PSNR (dB)	
	SPIHT-FSOT	JPEG2000
First frame of paris.cif		
0.0625	19.846	19.656
0.125	21.625	21.401
0.250	23.846	24.117
0.500	27.823	28.220
1.000	33.551	33.766
First frame of tempete.qcif		
0.0625	21.771	21.404
0.125	23.195	22.931
0.250	25.403	24.991
0.500	28.268	28.148
1.000	41.835	40.408

Figure 6-14 First frame (luminance) of paris.cif compressed by SPIHT-FSOT with arithmetic coding at 0.5 bpp

Figure 6-15 First frame (luminance) of paris.cif compressed by JPEG2000 at 0.5 bpp

Figure 6-16 First frame (luminance) of tempete.cif compressed by SPIHT-FSOT with arithmetic coding at 0.5 bpp

Figure 6-17 First frame (luminance) of tempete.cif compressed by JPEG2000 at 0.5 bpp

Table 6-8 Compression comparison of SPIHT-FSOT and JPEG200 on QCIF size images

bpp	PSNR (dB)	
	SPIHT-FSOT	JPEG2000
First frame of foreman.qcif		
0.0625	23.179	18.669
0.125	25.706	23.713
0.250	28.230	27.336
0.500	31.638	30.931
1.000	36.888	36.159
First frame of suzie.qcif		
0.0625	28.082	22.712
0.125	30.749	28.369
0.250	33.656	32.225
0.500	37.172	35.873
1.000	41.835	40.408

(a) (b)

Figure 6-18 First frame (luminance) of foreman.qcif compressed at 0.5bpp by (a) SPIHT-FSOT with arithmetic coding and (b) JPEG2000

(a) (b)

Figure 6-19 First frame (luminance) of suzie.qcif compressed at 0.5bpp by (a) SPIHT-FSOT with arithmetic coding and (b) JPEG2000

6.2.2 ANALYSIS

When compressing standard test images like *Len*a (512x512) and *Bike* (2560x2048), JPEG2000 performs marginally better than SPIHT with arithmetic coding using five DWT decomposition levels [3][4]. However, an important design consideration of JPEG2000 is that it must be able to encode images of any dimension. In particular, we study the compression performance of JPEG2000 on small images.

For CIF size images, results in Table 6.7 show that SPIHT-FSOT marginally out-performs JPEG2000 at lower bit rates. At higher rates the improvement may become less significant, or JPEG2000 may yield marginally better compression over SPIHT-FSOT. For QCIF images, the results in Table 6.8 show that SPIHT-FSOT consistently out-performs JPEG2000 at all bit rates, and it is clear that improvement of SPIHT-FSOT over JPEG2000 is significantly higher at lower bit rates. From Table 6.7 and Table 6.8, it is clear that SPIHT-FSOT is superior in encoding small images, especially at low bit rates.

In general, an arithmetic coder which uses adaptive context modeling extensively will require more information to be passed through the coder, so that the histograms of the models will become more updated to reflect the actual statistics, hence yielding better compression efficiency. A disadvantage is that sufficient symbols must first be coded for the models to reflect actual statistics reasonably well. At a given bit rate, less symbols will be coded by the arithmetic coder if the image is small, hence JPEG2000 is less effective in compressing small

images. This is evident from Table 6.7 and Table 6.8, which show results on CIF size images and QCIF size images respectively.

SPIHT-FSOT draws their compression efficiencies from the sorting algorithm, refining algorithm and FSOT structure, which work well with both small and large images. Arithmetic coding is optional and can be added to exploit additional compression efficiency, especially at higher bit rates.

In essence, SPIHT and SPIHT-FSOT are both able to outperform JPEG2000 in encoding smaller images since they share the same algorithm. However, SPIHT's rigid SOT requirements make it impossible to perform at least 5 DWT decomposition levels in most cases. As such, SPIHT can only outperform JPEG2000 in very special cases where the small image's width and height comprise of length 64, 128, 256, etc. SPIHT-FSOT overcomes this constraint and hence is superior to JPEG2000 in encoding smaller images in general.

7 VIDEO COMPRESSION PERFORMANCE

A video coding system that uses CSPIHT, CSPIHT-VL or CSPIHT-FSOT as its quantization technique is referred as a CSPIHT-based video coding system (CVCS). In this chapter, we compare the video compression performance of CVCS with H.263. In section 7.1, the overview of a CVCS is introduced, and the performance results are presented in section 7.2 while analysis is presented in section 7.3.

7.1 CSPIHT-BASED VIDEO CODING SYSTEM

In Chapter 2, the separate components that make up a video system are discussed. In this chapter, the full video model and additional implementation details are provided. CVCS has a hybrid model that utilizes both intra-frame and inter-frame coding. Figure 7-1 shows the full block diagram of CVCS.

Most blocks that constitute CVCS are also commonly used in other video CODEC. However, CVCS differentiates itself from other video codecs by the use of a CSPIHT-based coder as the quantization block.

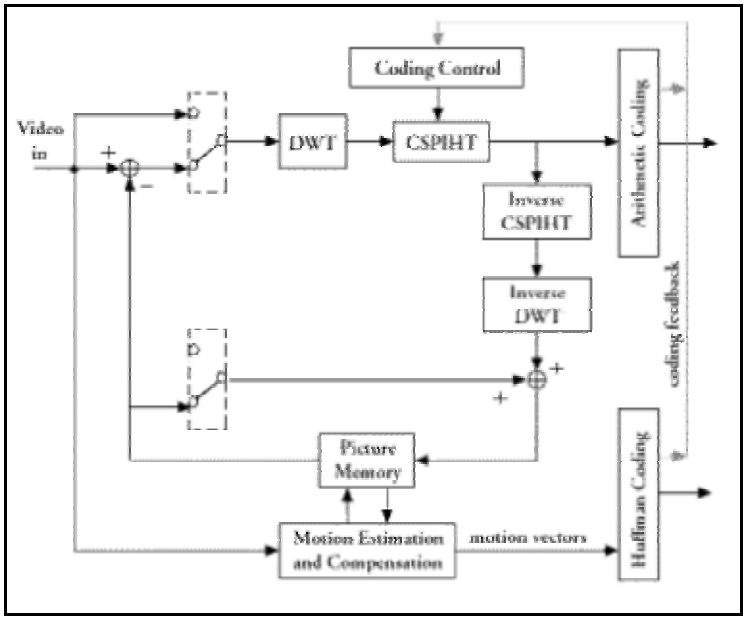


Figure 7-1 CVCS video model

As illustrated, there are two switches in the diagram. While operating in intra-frame mode, both switches will be in the top position; while coding in inter-frame mode, both switches will be in the bottom position. Figure 7-2 traces the path taken during intra-frame coding while Figure 7-3 shows that of inter-frame coding. Unused paths are deliberately removed from the figures and intermediate data are labeled to improve clarity.

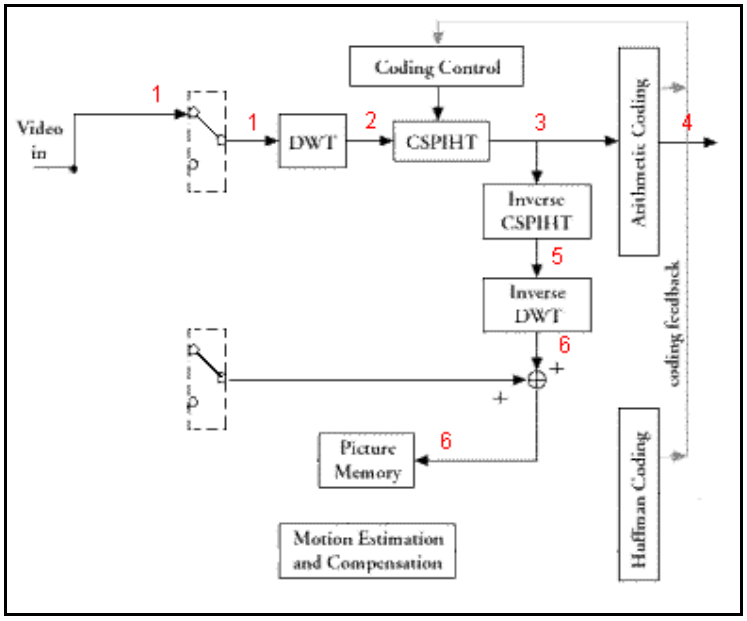


Figure 7-2 CVCS intra-frame coding
(1) Raw video data (2) wavelet coefficients (3) CSPIHT coded bit stream (4) Arithmetic coded bit stream (5) Reconstructed Wavelet coefficients (6) Reconstructed Image

It can be easily seen in both figures, CVCS finishes coding a frame by storing a copy of the reconstructed image into *Picture Memory*. The encoder must duplicate the same image that is transmitted to the decoder because inter-frame coding should minimize error in the image reconstructed by the decoder.

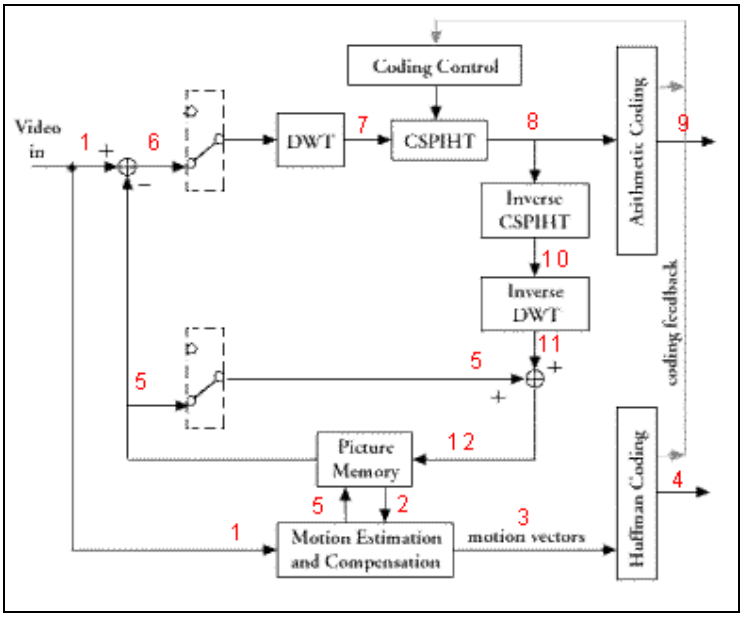


Figure 7-3 CVCS inter-frame coding

(1) Raw video data (2) Previous frame data (3) Motion vectors (4) Huffman coded motion vectors (5) Motion compensated image (6) Residual Signal (7) Wavelet coefficients (8) CSPIHT coded bit stream (9) Arithmetic coded bit stream (10) Reconstructed wavelet coefficients (11) Reconstructed error signal (12) Reconstructed image

7.2 NUMERICAL RESULTS

In our work, the Daubechies 9/7-tap biorthogonal filter bank [14] and whole-sample symmetric extension [15] are used to perform DWT in obtaining the results, which are binary coded without using arithmetic coding. 3 DWT decomposition levels are performed for CSPIHT and CSPIHT-VL, while 5 DWT decomposition levels are performed for CSPIHT-FSOT. Only 1 Virtual Level (VL) is used in CSPIHT-VL because the LL band dimension is 22x18, which only allows the deployment of 1 VL. Figures 7-4 to 7-11 show compression performance on foreman.qcif and suzie.qcif at 30 frames per second (fps) at different bitrates.

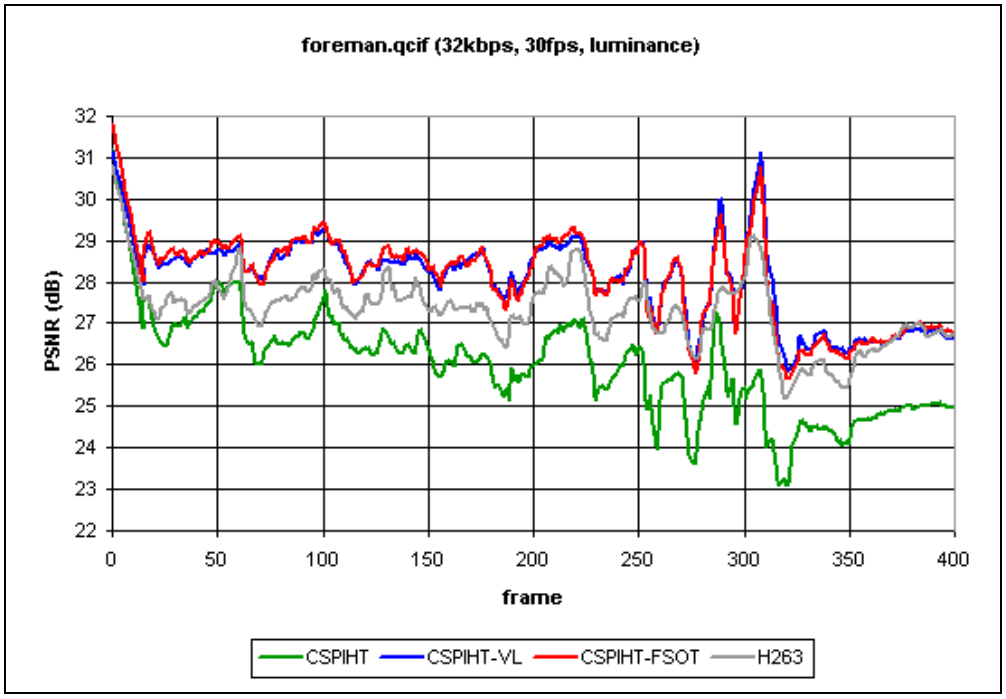


Figure 7-4 Compression performance (luminance) of foreman.qcif at 32kbps and 30fps.

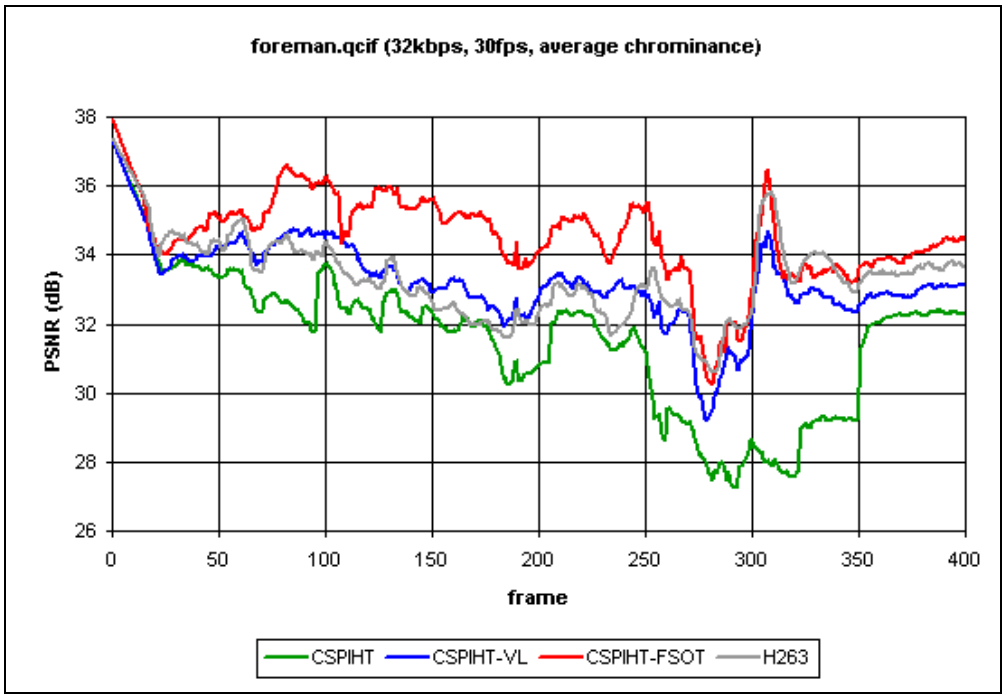


Figure 7-5 Compression performance (average chrominance) of foreman.qcif at 32kbps and 30fps.

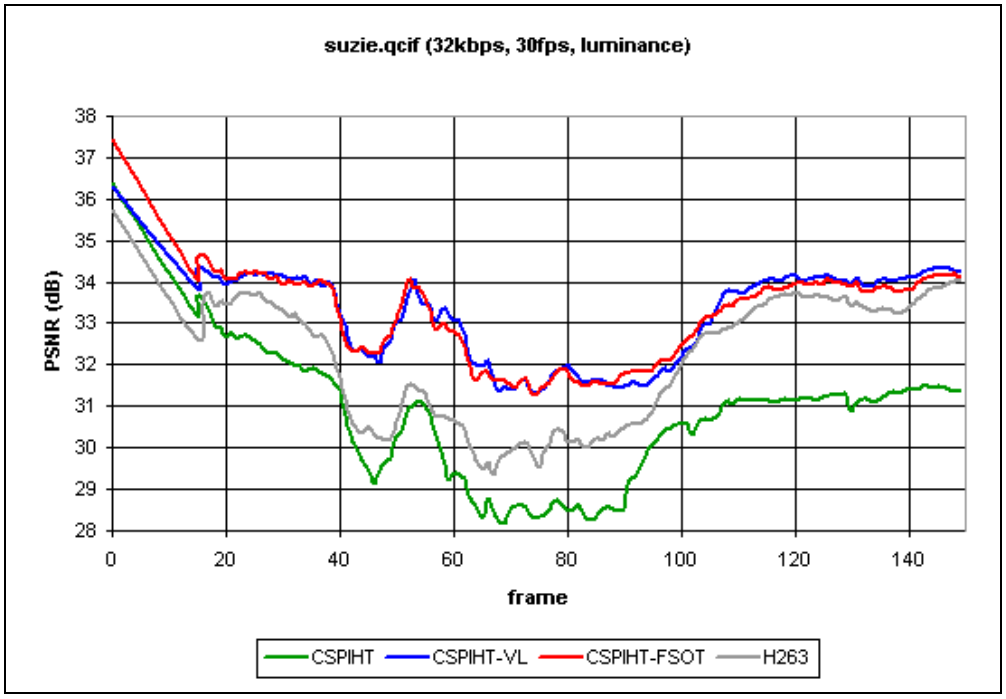


Figure 7-6 Compression performance (luminance) of suzie.qcif at 32kbps and 30fps.

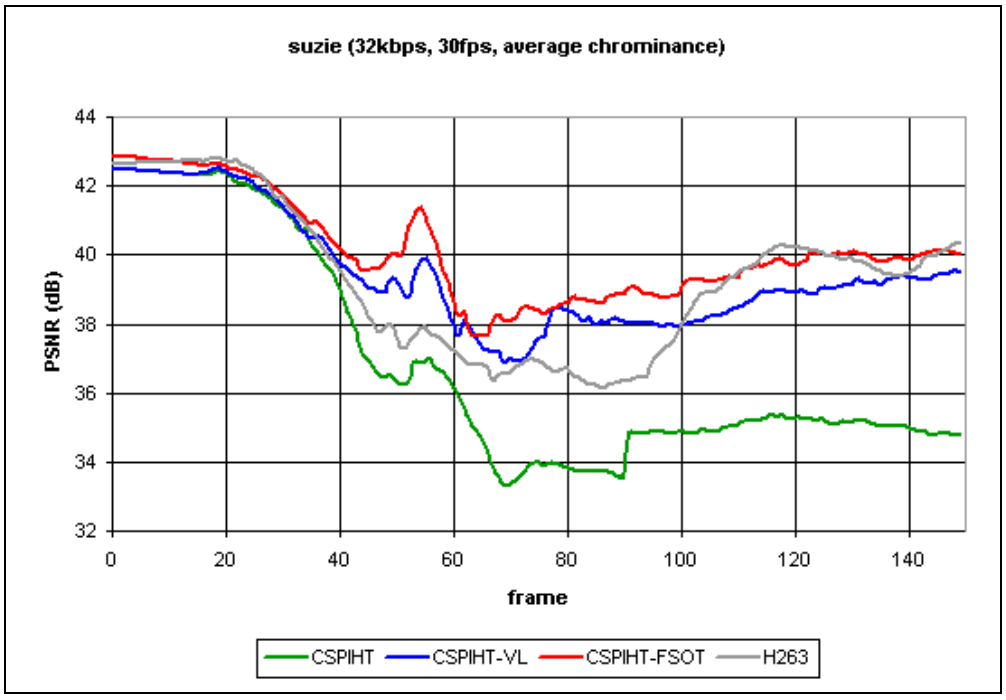


Figure 7-7 Compression performance (average chrominance) of suzie.qcif at 32kbps and 30fps.

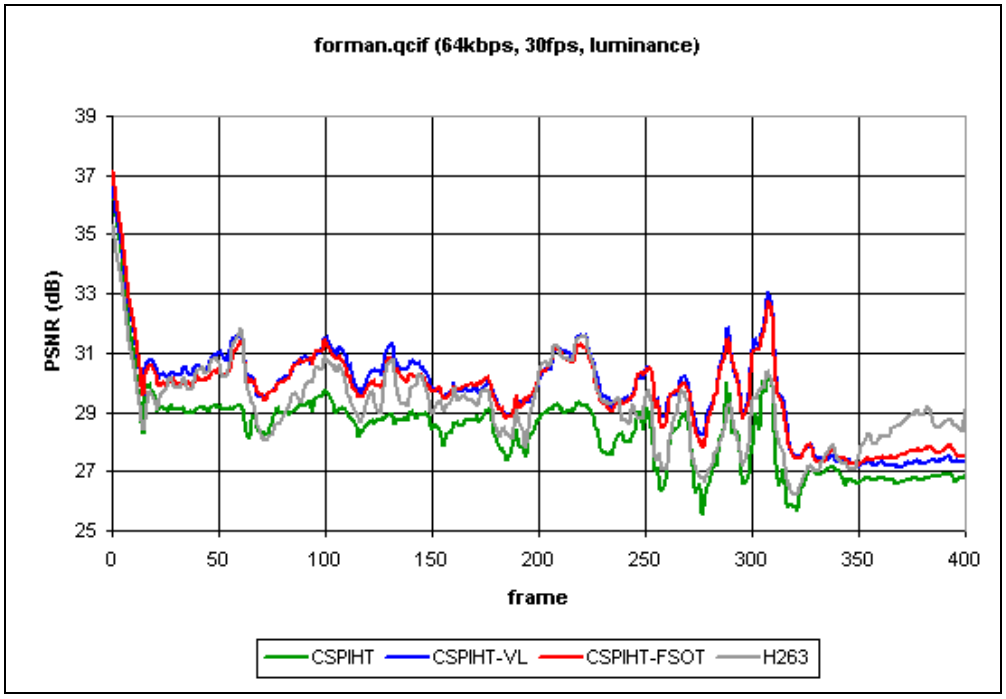


Figure 7-8 Compression performance (luminance) of foreman.qcif at 64kbps and 30fps.

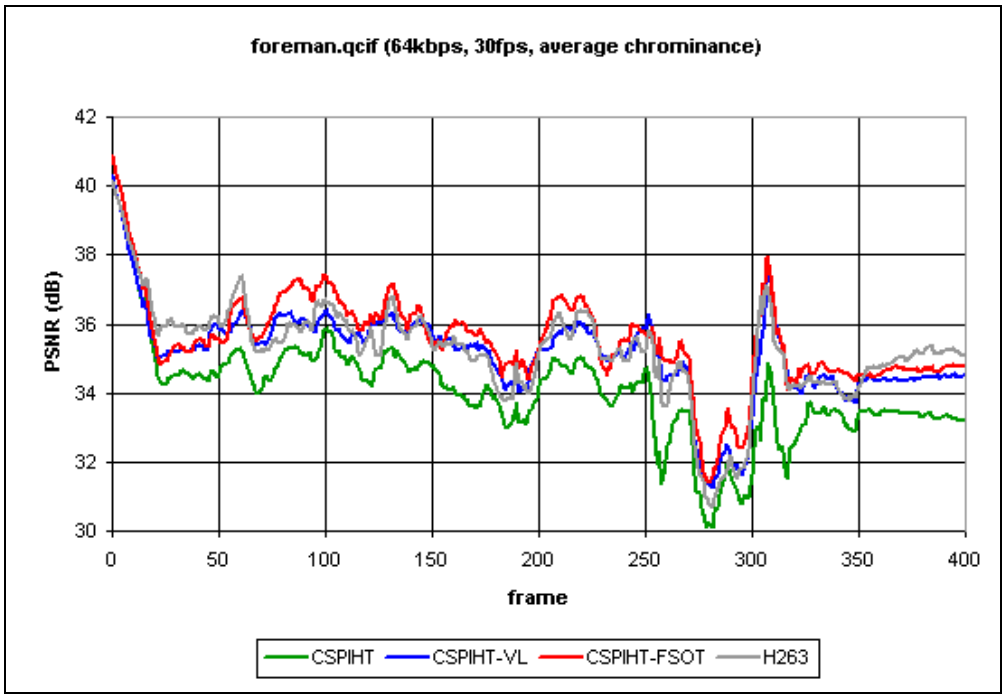


Figure 7-9 Compression performance (average chrominance) of foreman.qcif at 64kbps and 30fps.

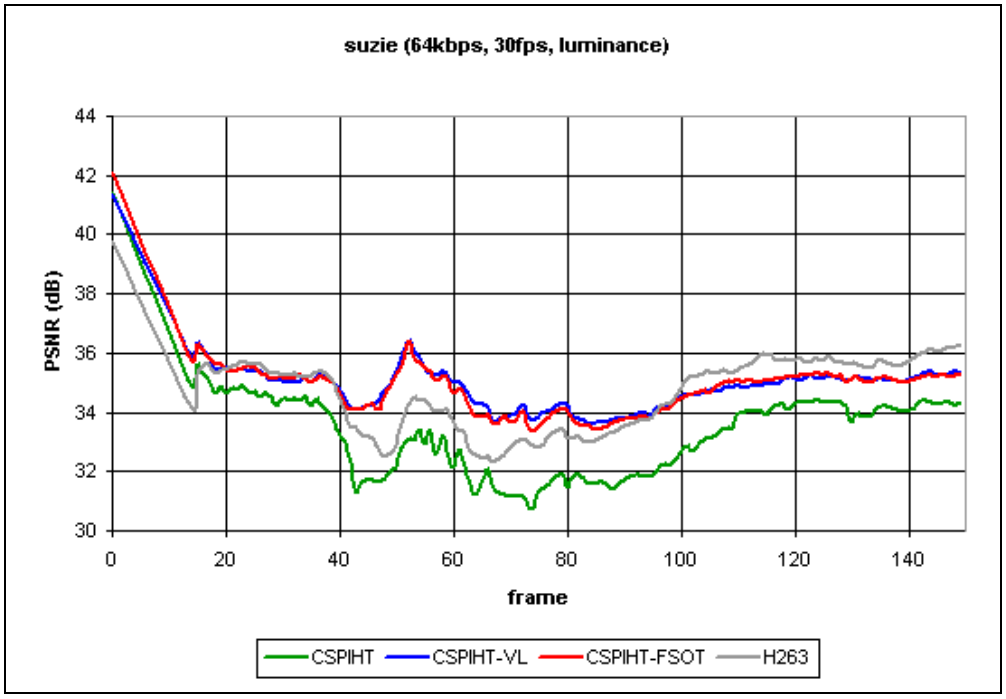


Figure 7-10 Compression performance (luminance) of suzie.qcif at 64kbps and 30fps.

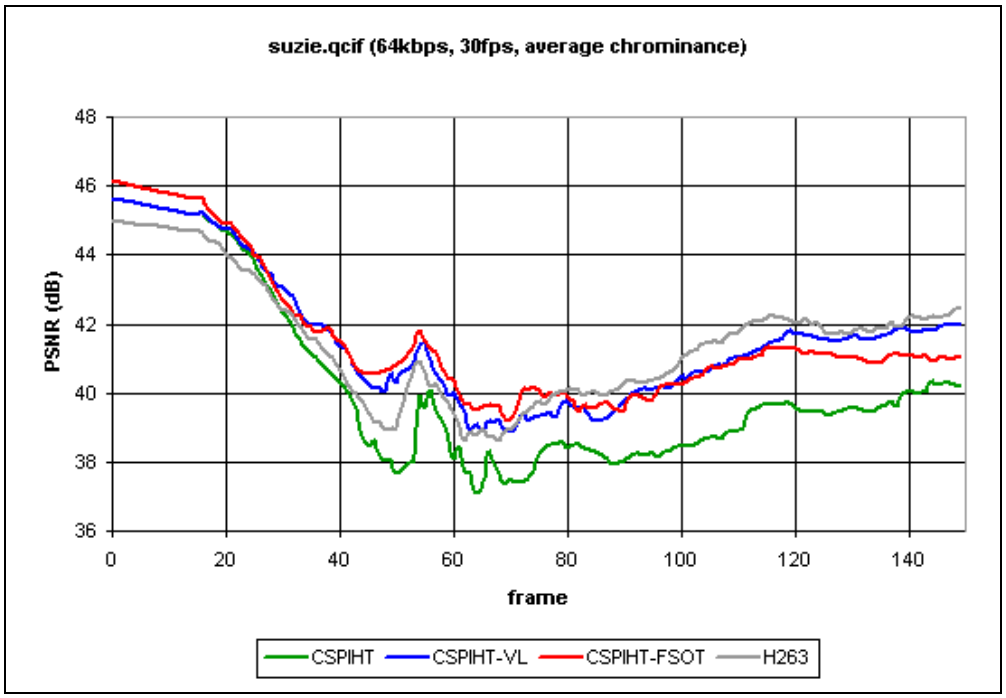


Figure 7-11 Compression performance (average chrominance) of suzie.qcif at 64kbps and 30fps.

7.3 ANALYSIS

In this section, we analyze the coding performance of the video coders. We first compare the CVCS using different CSPIHT-based coders, followed by comparing the best performing CVCS to H.263.

7.3.1 COMPARING CVCS USING CSPIHT, CSPIHT-VL AND CSPIHT-FSOT

At all bitrates, CVCS using CSPIHT-VL and CSPIHT-FSOT always outperform that using CSPIHT. From Figures 7-4, 7-5, 7-6 and 7-7, it is also shown that at lower bitrates, using CSPIHT-FSOT produces better compression efficiency over using CSPIHT-VL, especially in the compression of chrominance data. At higher bitrates, the efficiencies of using CSPIHT-FSOT over using CSPIHT-VL diminishes, which is evident from Figures 7-8, 7-9, 7-10 and 7-11. These observations are consistent with observations in Chapter 6, where the relative merits of VL and FSOT are discussed with regards to image compression. It is clear that CVCS using CSPIHT-FSOT gives the best performance.

7.3.2 COMPARING CVCS USING CSPIHT-FSOT WITH H.263

In Section 7.3.1, it is shown that CVCS gives best performance when using CSPIHT-FSOT. In this section, we compare the performance of CVCS using CSPIHT-FSOT with H.263. From Figures 7-4, 7-5, 7-6 and 7-7, it can be

easily observed that at low bit rates, compression performance of CVCS using CSPIHT-FSOT generally outperforms H.263. At higher bit rates, it is observed that the improvement of CVCS using CSPIHT-FSOT over H.263 is much reduced. It is also observed that at all bitrates, intra-frame coding performance of CVCS using CSPIHT-FSOT is always better.

CSPIHT-based coders exploit the self-similarity across subbands and decaying spectrum density from LL band to higher bands, which are characteristics of DWT maps of natural images. Residual signals, however, exhibit very different characteristics. As such, CVCS loses its compression efficiency over H.263 after prolonged inter-frame coding. These can be observed in all the figures starting from Figures 7-4 to Figure 7-11, where the compression improvement of CVCS using CSPIHT-FSOT diminishes as more frames are coded.

8 CONCLUSIONS

SPIHT is an image compression technique that exploits decaying spectrum density from LL to higher bands and self-similarities of the DWT map, by using a SOT structure. SPIHT maintains three lists: the LIP, LIS and LSP. LIP is initialized to hold all nodes in the LL band of DWT map while LIS excludes those that have no descendants.

The algorithm iterates through the sorting and refinement passes for each bit plane. Both LIP and LIS are sorted during the sorting pass. LIP is first traversed and the significance of each pixel is coded. If a pixel is significant, its sign will be coded and the pixel is moved to LSP. LIS, which contains both Type A sets and Type B sets, is then traversed. If a Type A set is significant, appropriate pixels within the set are checked for significance and handled accordingly, while the set itself will be added to the end of LIS as a Type B set. If a Type B set is significant, it will be broken up into smaller subsets and added to the end of LIS as Type A. Lastly, LSP is traversed in the refinement pass to refine the value of each significant pixel. CSPIHT, which shares the same compression algorithm with SPIHT, spawns a SOT that links up three DWT maps of different color planes to compress color images.

SPIHT uses SOT structures that require all subbands of the DWT map to have even width and even height. SOT of CSPIHT also requires all subbands of luminance plane to have even width and even height, allowing only the LL bands and the adjacent HL, LH, HH bands of U and V planes to have the same odd width and/or height. Due to restriction of SOT structure, both SPIHT and CSPIHT

are impractical to be used in real world applications, where images and video come in various dimensions.

Two enhancements, namely VL and FSOT, for SPIHT and CSPIHT have been proposed and their improvements in image and video compression are discussed and analyzed. Both VL and FSOT exploit the compression efficiencies that are lost due to lesser DWT decomposition levels performed (as a result of image dimension constrain) if the original SOT is deployed.

By using a Virtual Level, it is possible to yield better performance by creating longer branches with VL(s), which reduces the bits required to sort the initial LIP and LIS. With these longer branches, each initial set in LIS is also able to encapsulate more sets which leads to better compression efficiency. Since the initial LIP and LIS sets are disjoint in the original SPIHT algorithm, the correlation between large coefficients in LL band and the corresponding spatial locations in other bands is not exploited. With the use of Virtual Levels, coefficients on LL band and the corresponding coefficients in higher bands will be grouped under the same set, hence improving compression efficiency.

In essence, FSOT is superior compared to VL, as FSOT (1) has better compression efficiency than VL, and (2) also allows the compression of images with arbitrary dimensions. Consider the case where the image has a odd width or odd height. A SOT cannot be used and hence both SPIHT and CSPIHT will not be able to encode such an image. Since VL can only be deployed on top of a SOT, both SPIHT-VL and CSPIHT-VL are also unable to encode such an image. FSOT

overcomes the restriction on image dimensions and allows the compression of images with arbitrary dimensions.

FSOT also extends SOT elegantly such that, if all subbands have even width and even height, the structure of FSOT and structure of SOT will be exactly the same. Lastly, it is worth mentioning that FSOT does not require any modification on the algorithm of SPIHT (and CSPIHT).

Image compression results show that SPIHT-FSOT outperforms JPEG2000 in compressing small images. This is because JPEG2000 draws most of its compression efficiencies from its tightly integrated arithmetic coder, which requires sufficient symbols to pass through it so that its histogram will reflect the actual statistics reasonable well, but for a small image, the amount of symbols passed into the coder is significantly lesser than that of a large image at the same bit rate. The compression efficiencies of SPIHT-FSOT are mainly derived from its sorting algorithm, refining algorithm and FSOT structure, which work well with both small and large images. Arithmetic coding is optional and can be added to exploit additional compression efficiency, especially at higher bit rates.

Video compression results show that CVCS that uses CSPIHT-FSOT is in general superior to H.263, especially at low bit rates. SPIHT-based coders are, however, not effective in encoding residual signal, as such, CVCS loses its compression efficiency over H.263 after prolonged inter-frame coding.

8.1 APPLICATION OF FSOT

SPIHT is a very robust algorithm and is widely used in different areas. FSOT will be useful to most if not all of these areas by enabling the compression of images with arbitrary dimensions. For example, Modified SPIHT [16], Improved SPIHT [17] and Virtual SPIHT [18],[19] are some existing modifications on the SPIHT algorithm and structure of SOT. These modifications are all designed to work within the limitations of the existing SOT structure. FSOT can be incorporated into these SPIHT-base coders to enable the compression of image with arbitrary size.

Currently, a CSPIHT-3D video coder [20], similar to 3-D SPIHT [20], is also under-development. It is observed that in these coders, only 3 DWT decomposition levels are performed when compressing QCIF video sequences. FSOT can be easily extended into 3D to fit into the 3D block and allows more DWT decomposition levels to be performed, hence improving the compression efficiencies.

8.2 FUTURE WORK

Future research should focus on ways to better integrate inter-frame coding with the existing DWT plus CSPIHT-based coder combination. CVCS is less efficient in inter-frame coding because abrupt transitions are observed in the residual signal generated by motion estimation/compensation. Since the residual signal is then intra-frame coded, compression overhead increases because DWT is

more suitable for smooth signal. A better algorithm is needed to boost CVCS performance.

In [12], [13], [22], [23], the performance obtained by making modifications to the motion estimation/compensation techniques and the CSPIHT algorithm are investigated.

REFERENCE

[1] R. J. Clarke, "Image coding and the coding standards," *IEEE Conf. on Image Processing and its Applications,* vol. 1, pp. 1-6, 14-17 Jul 1997.

[2] M.W. Marcellin and A. Bilgin, "JPEG2000: highly scalable image compression," *IEEE Conf. On Information Technology*, pp. 268-272, Apr. 2001

[3] F. Ono, E. Ordentlich, G. Seroussi, D. Taubman, I. Ueno and M. Weinberger, "Embedded block coding in JPEG2000," *International Conf. Image Processing.* Vol. 2, pp. 33-36, 2000.

[4] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Trans. Image Processing.* Vol. 9, pp. 1158-1170, July 2000.

[5] A. Said and W.A. Pearlman, "A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees*", IEEE Trans.Circuits Syst. Video Technol.*, vol. 6, pp. 243-250, June 1996.

[6] W. K. Pratt, "Karhunen-Loeve transform coding of images," *Proc. 1970 IRRR Int. Symp. Inform. Theory*, 1970.

[7] Z. Xiong, K. Ramchandran, M. T. Orchard and Y. Q. Zhang, "A comparative study of DCT and wavelet based coding", *IEEE Symposium on Circuits and Systems*, vol. 4, pp. 273-276, 31 May-3 Jun 1998

[8] S. G. Mallat, "A Theory for Multiresolution Signal Decomposition : The Wavelets Representation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, pp. 674-693, June 1989.

[9] K. R. Rao and J. J. Hwang, "Techniques and Standards for Image, Video, and Audio Coding," *Prentice Hall*, July 1996.

[10] J.M. Shapiro, "Embedded Image Coding Using Zerotrees of Wavelets*", IEEE Trans. Signal Processing,* vol. 41, pp. 3445-3462, Dec. 1993.

[11] R. M. Rao and A. S. Bopardikar, "Wavelet Transforms: Introduction to Theory and Applications", *Addison-Wesley*, 1998.

[12] L.F.Zhao, A.A.Kassim, "A Fast Wavelet Color Image Coder for Real-time video coding", *Proc. Of IEEE Asia Pacific Conf. On Comm.,* pp. 654-658.

[13] L.F.Zhao and A.A.Kassim, "A New Class Color Image Codec for Scalable Video Coding *IEEE Trans. CSVT*, Aug., 1999.

[14] M. Antonini, M. Barlaud, P. Mathieu and I. Daubechies, "Image Coding Using Wavelet Transform," *IEEE Trans. Image Proc.,* Vol. 1, pp. 205-220, Jul. 1992.

[15] C. M. Brislawn, "Preservation of Subband Symmetry in Multirate Signal Processing," *IEEE Trans. Signal Processing,* Vol. 43, pp. 3046-3050, 1995.

[16] U. Bayazit and W. A. Pearlman, "Algorithmic modifications to SPIHT," *IEEE Conf. On Image Processing*, Vol. 3, pp. 800-803, 2001.

[17] Jian Zhu and S. Lawson, "Improvements to SPIHT for lossy image coding," *IEEE Conf. On Electronics, Circuits and Systems,* Vol. 3, pp. 1363-1366, 2001.

[18] E. Khan and M. Ghanbari, "Embedded color image coding with virtual SPIHT," *IEEE Conf. On Acoustics, Speech, and Signal Processing*, Vol. 4, pp. 3529-3532, 2002.

[19] E. Khan and M. Ghanbari, "Very low bit rate video coding using virtual SPIHT," *IEEE Electronics Letters,* Vol. 37, pp. 40-42, 4 Jan 2001.

[20] B. Kim, W.A. Pearlman and Z. Xiong, **"**Low bit-rate scalable video coding with 3-D set partitioning in hierarchical trees (3-D SPIHT)", *IEEE Trans. Circuits and Systems of Video Technology,* Vol. 10, pp. 1374-1387, 8 Dec. 2000.

[20] A. A. Kassim and E. H. Tan, "Video codec based on the 3D color set partitioning in hierarchical trees", *paper in preparation.*

[22] W. S. Lee and A. A. Kassim, "Low Bit-rate Video Coding Using Color Set Partitioning In Hierarchical Trees Scheme", *7th IEEE Singapore International Conference on Communication Systems*, Nov 2000.

[23] A. .A. Kassim and W. S. Lee, "Performance of the Color Set Partitioing in Hierarchical Trees (C-SPIHT) in Video Coding", *Council of Scientific Society Presidents Journal Special Issue on Multimedia Communication Services*, 2001.