UNIVERSITY OF CALIFORNIA, SAN DIEGO

Image Coding Subject to Constraints

A dissertation submitted in partial satisfaction of the

requirements for the degree Doctor of Philosophy

in

Electrical Engineering (Communications Theory and Systems)

by

Tamás Frajka

2003

The dissertation of Tamás Frajka is approved. And it is acceptable in quality and form for publication on microfilm:

Chair

University of California, San Diego

2003

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This work took many years to complete and many people to help along the way. I would like to thank my advisor Professor Ken Zeger for his guidance. I want to thank Professors Pamela Cosman, William Hodgkiss, Venkat Rangan, and Bhaskar Rao for serving on my committee.

I would also like to thank my parents, my sister and her family, Eleanor, and my extended family. I'm grateful for their support, love, patience, and faith in me. Last, but not least, many thanks to my friends both here and abroad, who kept me company all this time.

The text of Chapter 3, in part, is a reprint of the material as it appears in: T. Frajka, P. G. Sherwood, and K. Zeger. Progressive Image Coding with Spatially Variable Resolution. *International Conference on Image Processing*, pp 53-56, Vol. 1, Santa Barbara, CA, Oct. 1997. The text of Chapter 4, in part, is a reprint of the material as it appears in: P. Cosman, T. Frajka, and K. Zeger. Image Compression for Memory Constrained Printers. *International Conference on Image Processing*, pp 109-113, Vol. 3, Chicago, IL, Oct. 1998. The text of Chapter 5, in part, is a reprint of the material as it appears in P. Cosman, T. Frajka, D. Schilling, and K. Zeger. Memory efficient quadtree wavelet coding for compound images. *33rd Asilomar Conference on Signals, Systems and Computers*, pp 1173-1177, Vol. 2, Pacific Grove, CA, Oct. 1999. The text of Chapter 6, in part, is a reprint of the material as it appears in T. Frajka and K. Zeger. Robust Packet Image Transmission by Wavelet Coefficient Dispersement. *International Conference on Acoustics, Speech and Signal Processing*, pp 1745-1748, Vol. 3, Salt Lake

City, UT, May 2001. The text of Chapter 7, in full, is a reprint of the material as it appears in T. Frajka and K. Zeger. Residual Image Coding for Stereo Image Compression. *Optical Engineering*, 42(1):182-189, Jan. 2003. The text of Chapter 8, in full, has been submitted for publication as: T. Frajka and K. Zeger. Disparity Estimation Window Size. *Optical Engineering*, Dec. 2002. The text of Chapter 9, in full, has been submitted for publication as: T. Frajka and K. Zeger. Downsampling Dependent Upsampling of Images. *Signal Processing: Image Communication*, Apr. 2003. With the exception of the second and third publication I was the primary researcher and the co-author Kenneth Zeger listed in these publications supervised the research which forms the basis for this dissertation.

VITA

1995	M.S., Budapest University of Technology and Economics
1996-2003	Research Assistant, University of California, San Diego
2003	Ph.D., University of California, San Diego

PUBLICATIONS

T. Frajka, P. G. Sherwood, and K. Zeger. Progressive Image Coding with Spatially Variable Resolution. *International Conference on Image Processing*, pp 53-56, Vol. 1, Santa Barbara, CA, Oct. 1997.

P. Cosman, T. Frajka, and K. Zeger. Image Compression for Memory Constrained Printers. *International Conference on Image Processing*, pp 109-113, Vol. 3, Chicago, IL, Oct. 1998.

P. Cosman, T. Frajka, D. Schilling, and K. Zeger. Memory efficient quadtree wavelet coding for compound images. *33rd Asilomar Conference on Signals, Systems and Computers*, pp 1173-1177, Vol. 2, Pacific Grove, CA, Oct. 1999.

T. Frajka and K. Zeger. Robust Packet Image Transmission by Wavelet Coefficient Dispersement. *International Conference on Acoustics, Speech and Signal Processing*, pp 1745-1748, Vol. 3, Salt Lake City, UT, May 2001.

T. Frajka and K. Zeger. Residual Image Coding for Stereo Image Compression. *Optical Engineering*, 42(1):182-189, Jan. 2003.

T. Frajka and K. Zeger. Disparity Estimation Window Size. *Optical Engineering*, (submitted Dec. 2002).

T. Frajka and K. Zeger. Downsampling Dependent Upsampling of Images. *Signal Processing: Image Communication,*(submitted Apr. 2003).

ABSTRACT OF THE DISSERTATION

Image Coding Subject to Constraints

by

Tamás Frajka

Doctor of Philosophy in Electrical Engineering

(Communications Theory and Systems)

University of California, San Diego, 2003

Professor Kenneth Zeger, Chair

An image in digital format is a rectangular array of nonnegative integers, representing the light or color intensity values in the image. Its compression or coding is concerned with trying to represent the image in as compact a form as possible. Modern image coding techniques must be able to deal with a wide variety of constraints, both from image sources and from physical system implementations.

This thesis focuses on image coding under certain special conditions. In particular, the coding of images is subject to constraints on fast recognition, memory constraints from physical implementation, special content of images, constraints from the transmission media, special image types, and fidelity constraints on image resizing.

We present a progressive coding technique for Region of Interest coding. We analyze image compression in terms of memory requirements and offer a low complexity approach to document image coding. We introduce a novel packet formation approach for image transmission over lossy packet networks. In stereoscopic image compression, we propose a method for improved coding of residual images and present an estimation algorithm for disparity window size selection. For the problem of image size conversion we introduce a solution that aims to identify the downsampling method used to improve the outcome of the upsampling operation.

Chapter 1

Introduction

Traditionally the primary concern of image coding has been to compress the image as well as possible while avoiding introducing visual distortion to the original image.

As in any source coding problem, we can distinguish between lossless and lossy encoding of images. While lossless compression techniques preserve the original image, they typically do not result in better than $3:1$ compression ratios. Some lossy techniques on the other hand are capable of producing images that are almost indistinguishable from the originals at a $20:1$ or higher compression ratio. The work presented in this proposal is in the area of lossy compression.

The advancements in communication technology and the popularity of the Internet have introduced further constraints into the coding of images. In certain scenarios, it is desirable to be able to recognize an image as quickly as possible during a download or to be able to better preserve certain areas in the image. This requires an image coder to display information important for recognition first as opposed to uniformly improving the entire image, or to better encode the desired area. Chapter 3 presents a possible solution to this problem.

Digital storage of images allows easy distribution and sharing. But it is also important to be able to obtain a paper copy of such images. Typical images may require large transmission bandwidths to printers that may cause bottlenecks in a network or slow transmission on low capacity links. It may be desirable to be able to send images in compressed form and then decode them at the printer. Hardware implementations of printers represent restrictions on image decoding that are typically not encountered in other applications. In Chapter 4 we analyze image compression methods from this point of view and propose a coding scheme that meets certain printing requirements.

Images to be compressed are not limited to natural images. More and more frequently images contain text, charts, line graphics, and drawings that introduce new characteristics to an image, mainly changing its mostly slow varying, low frequency nature. In Chapter 5 we propose coding techniques that outperform traditional image coders on compound images.

Communication of images may mean transmission over non-perfect channels that introduce errors to the transmitted bit-stream. These errors may cause catastrophic failure at the decoder, possibly rendering the decoded image unrecognizable. In Chapter 6 we introduce a solution for transmission over lossy packet networks that tries to mitigate the effect of the loss of image information on visual image quality.

In many applications depth rendering and improved depth perception can play an important role. Stereoscopic image pairs taken by two cameras can provide a simple solution for such problems. These image pairs contain significant redundancy which can be exploited in compression to improve performance. We present a stereo coding technique in Chapter 7.

An important building block of many compression mechanisms for stereoscopic images is disparity estimation, a method that aims to reduce the inherent redundancy of the image pair. In Chapter 8 we investigate the problem of disparity estimation and propose a technique to improve its efficiency.

The technological boom of the past few years led to an increased variety of display sizes and capabilities. Image size conversion has become an everyday necessity. Often images go through two or more steps of size conversion where earlier resizing steps can have an impact on the quality of the end result. The “signature” of a previous downsampling operation can be identified and that information can be used to choose a proper upsampling method as we show in Chapter 9.

Chapter 2 gives an overview of image coding and some of the techniques used in later chapters. Finally, in Chapter 10 we present some conclusions and propose some research ideas for future work.

Chapter 2

Image Coding Background

This chapter provides a general overview of transform based image coding and an introduction to the coding methods referred to later in this dissertation. Figure 2.8 shows a collection of original images frequently used in the image compression literature which we will also use in many of the later chapters.

In digital form, a gray scale image of size $X \times Y$ is represented by its intensity values, $I(i, j)$. A color image is given by its intensity values in the red, green, and blue color planes, $I_r(i, j)$, $I_g(i, j)$, and $I_b(i, j)$. In both cases $1 \leq i \leq X$ and $1 \leq j \leq Y$.

Initial efforts in image compression focused on spatial domain manipulation, that is, working directly with the pixel intensity values of the image using different forms of quantization. However, these schemes have not produced results very close to the theoretical rate-distortion limits for the source.

Interest later turned to transform coding of images using invertible, linear transforms (such as the Fourier Transform, the Discrete Cosine Transform, or the Discrete Wavelet Transform). These transforms map an image into a set of transform coefficients. For natural images, most of these coefficients have small magnitude values, permitting efficient quantization of the image transform coefficients.

In image compression the information rate is measured in bits-per-pixel (bpp). In an uncoded image 8 bits are typically used to represent each color. Thus for gray scale images, the uncoded rate is equivalent to 8 bpp, and for color images it is 24 bpp. The image quality is frequently measured by the Mean Squared Error (MSE), given as

$$MSE = \frac{1}{XY} \sum_{i=1}^{X} \sum_{j=1}^{Y} (I_1(i,j) - I_2(i,j))^2 \,,$$

where I_1 and I_2 are two images given by their intensity values, or by the Peak Signal-to-Noise Ratio (PSNR)

$$PSNR = 10 \log_{10} \frac{255^2}{MSE} \,.$$

In an image 255 is the largest intensity value, and thus the peak signal energy of an image is 255^2. While the PSNR gives a good indication of image quality it is not entirely accurate, as the human visual system's error perception is not perfectly described by the squared error model, and often images with lower PSNR are better perceptual representations of the original image.

A bit stream is said to be *progressive* if it can be decoded at more than one transmission rate to yield potentially coarser image qualities at lower transmission rates. Such a bit stream is also referred to as *SNR scalable*. A bit stream is said to be *embedded* in another if it is a prefix of the other bit stream. A bit stream is said to be *resolution scalable* if it contains distinct subsets representing the image at different resolutions. An image coder is *progressive* if the encoder output bit stream for every transmission rate is progressive.

2.1 JPEG standard

The Joint Photographic Experts Group (JPEG) standard is one of the most popular image compression algorithms in use today. The standard contains both lossy and lossless algorithms. It is based on the well known 2-dimensional Discrete Cosine Transform (DCT)

$$C(u,v) = \alpha(u)\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} I(x,y)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N}\,, \tag{2.1}$$

where N is the block size, and $\alpha(u)$ and $\alpha(v)$ are normalizing factors, given by

$$\alpha(u) = \begin{cases} \sqrt{1/N} & \text{for } u = 0 \\ \sqrt{2/N} & \text{for } u = 1, 2, \ldots, N-1 \end{cases} \tag{2.2}$$

The JPEG compression algorithm works in three steps: the image is first transformed, then the coefficients are quantized, and finally encoded with a variable-length lossless code.

For image transformation the image is subdivided into 8×8 blocks, and a DCT is performed on each block, processing them from left to right, top to bottom. Each coefficient in a block is uniform scalar quantized using some experimentally determined step size. The quantized values are then reordered according to a zigzag pattern shown in Figure 2.1. This reordering takes the coefficients in the order of increasing spatial frequency. This corresponds to the observation that intensity values change smoothly in most natural images. In the frequency domain most of the energy is compacted into the coefficients representing the lowest spatial frequencies. With an effective quantizer, only the first few coefficients in this order will carry significant information, creating long runs of zeros in the stream. The first coefficient in the zigzag scanning order (often

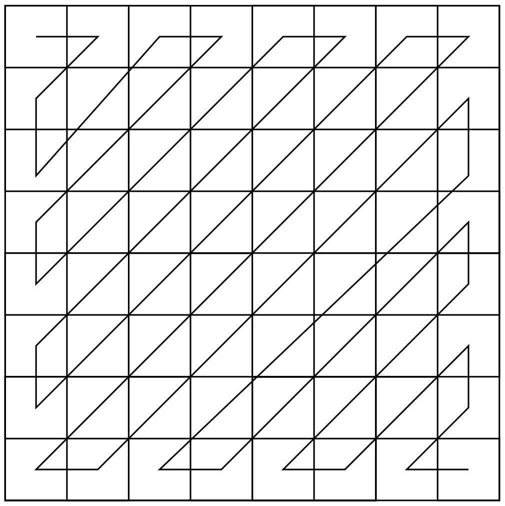


Figure 2.1: Zigzag ordering of coefficients of the 8×8 blocks in the JPEG algorithm.

referred to as the DC coefficient) is encoded separately. To take advantage of the slow varying nature of most natural images, the DC coefficient is predicted from the DC coefficient of the previous block and differentially encoded with a variable length code. The rest of the coefficients (referred to as AC coefficients) are run-length coded. More details can be found in [27, Chapter 6.].

JPEG produces relatively good performance at medium to high coding rates, but suffers from blocking artifacts at lower rates because of the block-based encoding. At low rates, in order to meet the target coding rate, the quantization table chosen is so coarse that only the DC coefficient is encoded without the higher frequency details. At the decoder this creates visually noticeable discontinuities at the block boundaries.

2.2 Wavelet Transform

Wavelet transforms provide good spatial and frequency localization that make them very suitable for image processing and compression.

Wavelets are functions generated from one single function $\Psi : \mathbb{R} \mapsto \mathbb{R}$, called the *mother wavelet*, by dilations and translations. The mother wavelet has to satisfy $\int \Psi(x)dx = 0$. Let $\Psi^{a,b}(t) = |a|^{-1/2}\Psi((t-b)/a)$ denote the dilation and translation of the mother wavelet. Then in wavelet analysis high frequency wavelets correspond to $a < 1$ (narrow width), while low frequency wavelets have $a > 1$. Thus frequency resolution can be traded off with spatial resolution and vice versa.

As with other transforms the goal is to represent any function f as a superposition of wavelets; in practice one prefers to write f as a discrete superposition

$$f = \sum_{m,n} c_{m,n}(f)\Psi_{m,n}$$

where $\Psi_{m,n} = \Psi^{a_0^m, nb_0a_0^m}$ with $a_0 > 1$ and $b_0 > 0$ fixed.

In multiresolution analysis one also has a scaling function $\Phi : \mathbb{R} \mapsto \mathbb{R}$ with its dilated and translated versions $\Phi_{m,n}(x) = 2^{-m/2}\Phi(2^{-m}x - n)$. For a fixed value of m the $\Phi_{m,n}$ are orthonormal. Let V_m denote the space spanned by the $\Phi_{m,n}$. These spaces then describe successive approximation spaces, each with resolution 2^m. The space W_m spanned by $\Psi_{m,n}$ is the orthogonal complement in V_{m-1} of V_m, for each m. Thus the coefficients

$$c_{m,n}(f) =< \Psi_{m,n}, f >= \int \Psi_{m,n}(x)f(x)dx$$

describe the information lost when going from the finer resolution, 2^{m-1}, to the coarser resolution, 2^m. From this one can get the following algorithm for the computation of the wavelet coefficients:

$$c_{m,n}(f) = \sum_k g_{2n-k}a_{m-1,k}(f)$$

$$a_{m,n}(f) = \sum_k h_{2n-k}a_{m-1,k}(f) \tag{2.3}$$

where $h_n = \sqrt{2}\int \Phi(x-n)\Phi(2x)dx$ and $g_l = (-1)^l h_{-l+1}$. If f is given in sampled

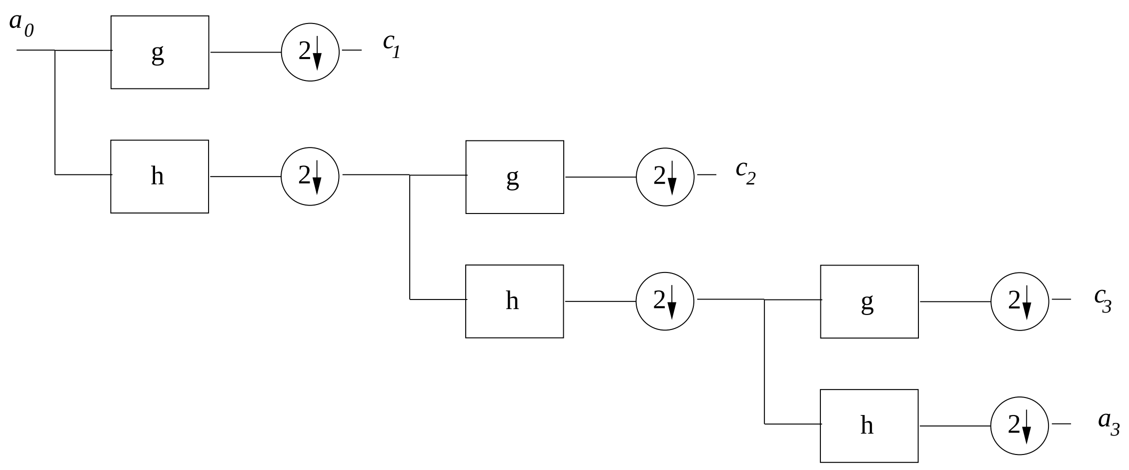


Figure 2.2: Decomposition filter bank for wavelet transforms.

form, then one can take these samples as the highest order resolution coefficients $a_{0,n}$ and describe a subband coding algorithm using the lowpass filter h and the highpass filter g. The reconstruction in the case of orthonormal wavelet bases is given by

$$a_{m-1,l}(f) = \sum_n \left[h_{2n-l} a_{m,n}(f) + g_{2n-l} c_{m,n}(f)\right].$$

These equations form the basis of the multiscale image decomposition filter structure depicted in Figure 2.2. Instead of using a two-dimensional filter for images, this one-dimensional filterbank is used in the horizontal and the vertical direction to create the two-dimensional wavelet transform. The resulting subband structure for a 2-level decomposition is shown in Figure 2.3. (In the notation LL_i, LH_i, etc. the first letter refers to the horizontal filtering (L:lowpass, H:highpass) at level i and the second letter to the vertical filtering of the coefficients in that subband.) The above octave band decomposition assumes that most of the energy is contained in the low frequency coefficients. For some images this iteration on the LL band may not provide the best energy compaction; a best basis selection algorithm ([18, 65]) could be used to optimize the decomposition structure.

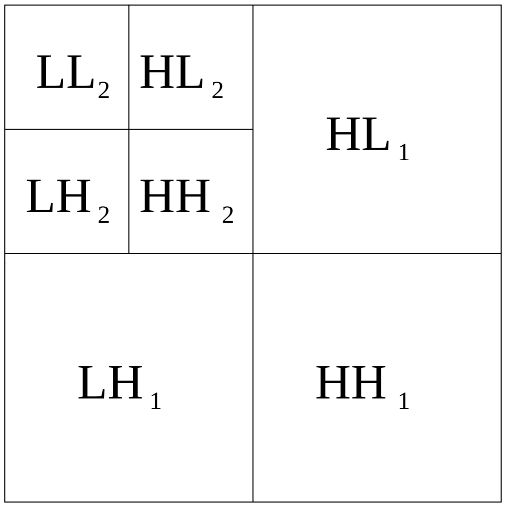


Figure 2.3: The 2-level wavelet decomposition with subband notation.

Most natural images are smooth and slowly varying. One would expect an exact reconstruction subband coding scheme to rely on orthonormal bases with a reasonably smooth mother wavelet. The filters should also be short to facilitate fast computation. Since the filters are used in a pyramidal filter structure, the linear phase property would allow their cascading without phase compensation. Unfortunately, there do not exist any nontrivial orthonormal linear phase FIR filters with the exact reconstruction property. The only symmetric, exact reconstruction filters are the Haar filters with $h_0 = h_1 = \sqrt{2}$, $g_0 = -g_1 = \sqrt{2}$, and $h_n = g_n = 0$ for all other n. The lowpass filter performs the averaging operation, and the highpass filter computes the difference.

If one relaxes the orthonormality condition and uses biorthogonal wavelet bases, then the linear phase property can still be preserved. The signal decomposition will remain the same as in (2.3). However, the reconstruction is different:

$$a_{m-1,l}(f) = \sum_n [\tilde{h}_{2n-l} a_{m,n}(f) + \tilde{g}_{2n-l} c_{m,n}(f)]$$

where the *reconstruction filters* $\tilde{h}, \tilde{g}$ may differ from the *analysis filters* h, g. Exact reconstruction is guaranteed if

$$\tilde{g}_n = (-1)^n h_{-n+1}$$

$$g_n = (-1)^n \tilde{h}_{-n+1}$$

$$\sum_n h_n \tilde{h}_{n+2k} = \delta_{k,0}.$$

The most commonly used wavelet filters in image compression are the biorthogonal, so-called "9-7" filters (named after the number of filter taps in the low and high frequency filters) introduced in [3]. For more on wavelet filter design and filters see [87].

2.3 Wavelet Based Coders

Initial efforts in the coding of wavelet coefficients focused on traditional quantization techniques. (For an overview see [20]). Many of these techniques led to improvements over DCT-based methods, but the gains were mostly due to the use of the wavelet transform, and not the coding of wavelet coefficients.

2.3.1 Set Partitioning in Hierarchical Trees

A conceptually new coding technique, the Embedded Zerotree Wavelet (EZW) coder was introduced by Shapiro [72], that was later refined by Said and Pearlman [68]. Both EZW and Said and Pearlman's Set Partitioning in Hierarchical Trees (SPIHT) algorithm use implicit quantization by encoding the coefficients by bit planes starting from the most significant bit. This enables progressive decoding as more bits arrive to the decoder. The challenge is to efficiently code the bit planes. Given the subband structure of the wavelet transform, the algorithms make use of the self-similarity across different scales as shown for the wavelet transform of the *Lena* image in part (b) of Figure 2.4. The trees describing these spatial similarities are called *zerotrees*, depicted in part (a) of Figure 2.4.

The coefficient in the LL band forms the root of the tree while the corresponding higher frequency coefficients are its *descendants*. To each coefficient in a lower frequency band corresponds four coefficients in the next higher frequency band in the zerotree structure. Given a wavelet coefficient $W(i,j)$, its four *children* are given as $\{W(2i+k, 2j+l),\ k=0,1;\ l=0,1\}$. Coefficients in the highest frequency bands (where $i \geq \lfloor X/2 \rfloor$ or $j \geq \lfloor Y/2 \rfloor$) have no children. For each coefficient, its descendants can be determined by recursively seeking the children of its children and then their children, etc.

The coding is done in two passes for each bit plane. The first pass identifies the significant coefficients, i.e. coefficients whose most significant bit lies in the given bit plane. The second pass refines the coefficients already found significant at previous passes. After each iteration the algorithms move on to the next less significant bit plane. The coding relies on the fact that most images have their energy in the LL band and if a coefficient is not significant on a given bit plane its descendants in the zerotree are also likely to be insignificant. Thus entire regions of wavelet coefficients can be deemed insignificant by a single bit in this framework. To further reduce redundancy, the code bits are encoded using context-based adaptive arithmetic coding ([89]). SPIHT is one of the best image coders for general image compression today.

2.3.2 MultiGrid Embedding

MultiGrid Embedding (MGE) coding by Lan and Tewfik [43] is an alternative to the zerotree structure of EZW and SPIHT. Both MGE and SPIHT perform bit plane coding; the difference is in the identification of significant coefficients for each bit plane. MGE

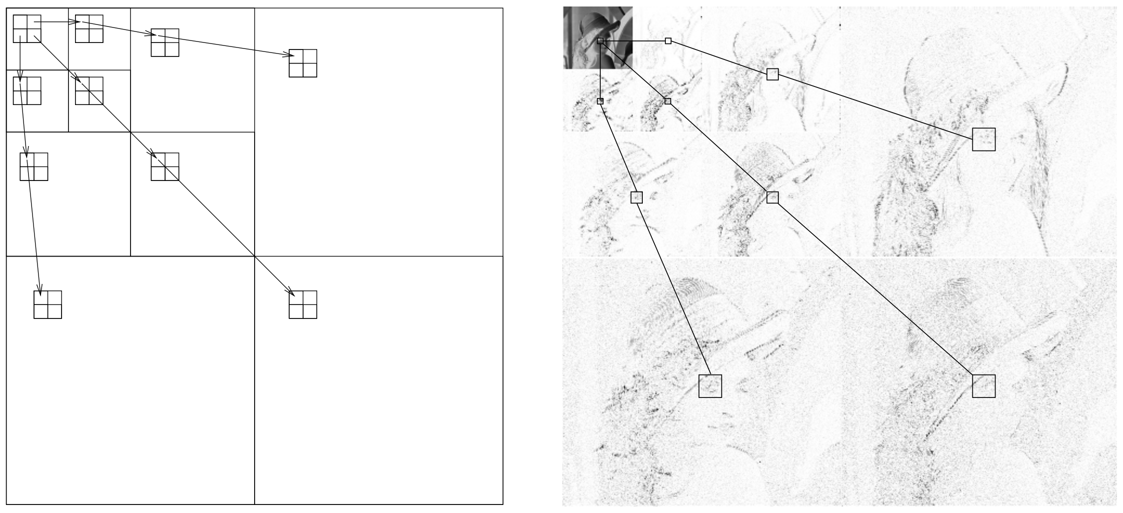

(a) (b)

Figure 2.4: (a) Subband structure of a 3-level wavelet decomposition, (b) spatial dependence trees of a 3-level decomposition.

does not rely on the spatial similarities between subbands to identify significant wavelet coefficients. Instead it uses a quadtree style decomposition. At each step of the decomposition the block in question is split into four blocks at the midpoint along each side. The splitting strategy is known to both the encoder and decoder and it does not require any further communication.

Starting with the entire wavelet domain image, a quadtree search is conducted to find the significant coefficients on the most significant bit plane. For each image block a single bit describes if it contains any significant information. The quadtree decomposition is continued until it reaches the individual coefficient level. The insignificant blocks are revisited on subsequent passes parsing less significant bit planes. The performance of the algorithm is similar to that of SPIHT, as shown in Table 2.1. MGE outperforms SPIHT on images with significant high frequency information where this search method reaches the high frequency coefficients faster than the zerotree description. These results indicate that the strength of the zerotree and MGE structures lie in efficiently identifying large areas of insignificant wavelet coefficients.

Table 2.1: Comparison of SPIHT and MGE coding methods in terms of PSNR over a wide range of rates and image characteristics.

		PSNR (dB)							
		Lena		*Peppers*		*Baboon*		*Barbara*	
		SPIHT	MGE	SPIHT	MGE	SPIHT	MGE	SPIHT	MGE
	0.1	30.19	30.19	30.64	30.55	21.65	21.72	24.25	24.47
	0.25	34.09	34.12	34.18	34.06	23.55	23.58	27.56	27.97
Rate	0.5	37.19	37.27	36.47	36.44	25.92	26.01	31.38	31.90
(bpp)	0.75	39.03	39.07	37.80	37.72	27.67	27.85	34.24	34.64
	1	40.39	40.47	38.89	38.88	29.44	29.58	36.40	36.90

2.3.3 EBCOT

Embedded Block Coding with Optimized Truncation (EBCOT) by Taubman ([79]) is the building block of the new JPEG 2000 [50] standard for still image coding. Based on the wavelet transform, the coding is performed on individual nonoverlapping blocks in the wavelet domain that generate independent bit-streams for each portion of the image. Within a block the quadtree technique is used for identifying significant coefficients. The coder is very flexible and can support a wide array of applications, such as resolution or SNR scalable compression. To achieve finer embedding, the EBCOT authors introduced fractional bit planes. EBCOT outperforms SPIHT and MGE on most natural images.

2.3.4 Quadtree-Guided Encoding

In [81], Teng and Neuhoff used the quadtree decomposition for the compression of images in the spatial domain. [82] is an extension of that work to wavelet domain coding. Their method uses one or two levels of wavelet decomposition. The encoding

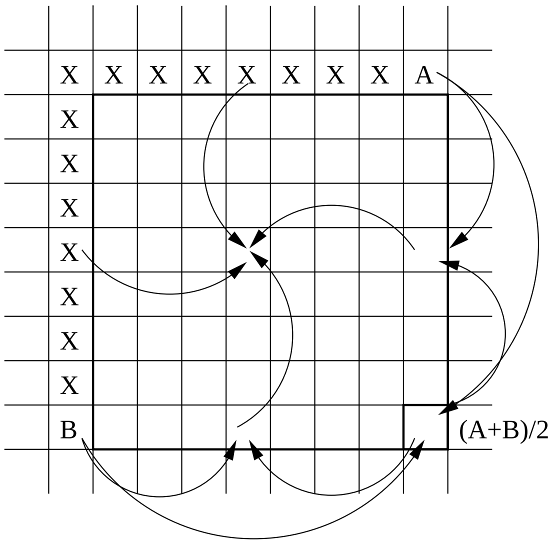


Figure 2.5: Quadtree encoding: the lower right corner pixel (the foot) is first predicted from pixels A and B, which in turn is used in the linear prediction of other coefficients, until the whole block is interpolated.

of the LL band and the outer bands are different, and in fact the LL band encoding strongly influences the outer band encoding. The LL band is divided into $k \times k$ blocks (typically k is 8 or 16) which are processed in raster scan order. For each block, the lower right corner (called the foot) is predicted from the pixel just above and to the left of the block in the same line or row, as shown in Figure 2.5. The prediction error is quantized and added to the prediction. Using this foot value, as well as the reconstructed values of neighboring pixels from previous adjacent blocks, the rest of the pixels are predicted using two and four point linear interpolation. A quality test then checks how well the interpolation approximates the real pixel values. If the block passes the test, the quantized prediction error for the foot is transmitted using a variable length code. If not, the block is split into four sub-blocks, and the same procedure is repeated for those.

The coding of the outer band relies on the assumption that if a block is difficult to predict linearly in the LL band, then the corresponding coefficients in the higher frequency bands are significant. A block is declared difficult to predict if the interpolation

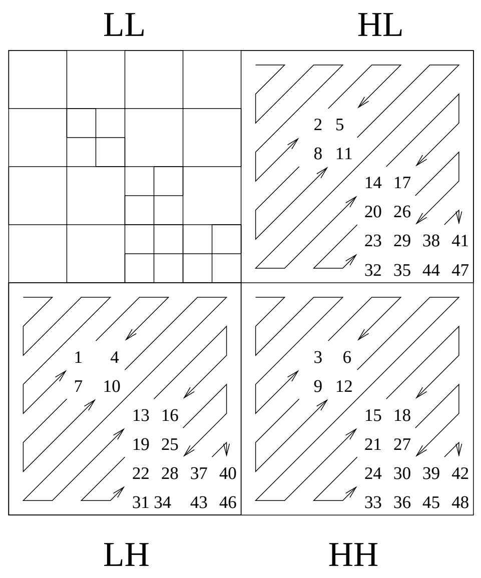


Figure 2.6: Quadtree encoding showing the block division in the LL band, the significant coefficients in the outer band and the scanning order (which applies to the significant coefficients only.)

fails the quality test on a 2×2 block. The high frequency band coefficients corresponding to those highly subdivided LL blocks are quantized with a symmetric scalar quantizer; for all the blocks where the interpolation passed the quality test, the corresponding high frequency coefficients are not encoded (therefore quantized to zero). This is depicted in Figure 2.6. Finally, those quantized coefficients are runlength encoded in a zig-zag pattern similar to JPEG with the modification that coefficients in the same position in each outer band are coded one after the other, as the numbering in Figure 2.6 indicates.

2.4 Embedded Zerotree DCT (EZDCT)

One of the drawbacks of JPEG encoding is the inflexibility of the rate control. In JPEG, rate control consists of choosing a *quality factor* which serves as a scaling factor of the standard quantization tables. The quality factor ranges between 0 and 100. The finite choice of quality factor limits the available rates to a finite set as well. In addition, the

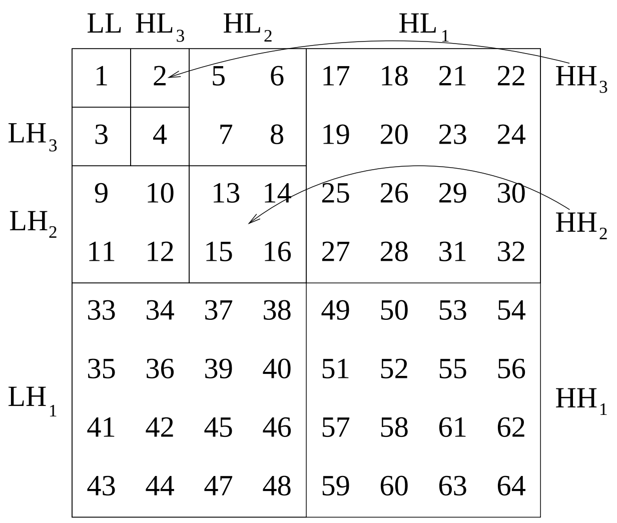


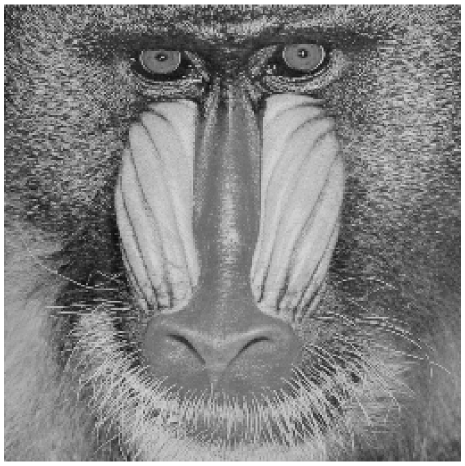
Figure 2.7: Wavelet-style 3-level decomposition applied to an 8×8 block of DCT coefficients.

target rate is found by iteratively changing the scaling of the quantization tables.

Xiong et al. in [92] proposed to apply the zerotree structure to the DCT transformed image. The 8×8 DCT block is treated as a 3-level decomposition as shown in Figure 2.7. The structure shown in the figure also represents the zerotree dependency among these 64 coefficients. Coefficient 1 has three children (2,3, and 4). The parent of coefficient c is $(\lfloor (c-1)/4 \rfloor + 1)$ for $2 \leq c \leq 64$. The four children of coefficient p are $\{4p-3, 4p-2, 4p-1, 4p\}$ for $2 \leq p \leq 16$. Coefficients with the same index from all 8×8 blocks are grouped together. Embedded zerotree quantization is applied to the above structure with the coefficients being scanned starting from the DC coefficients.

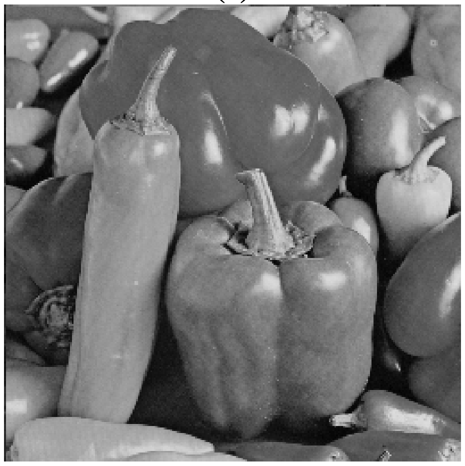
This method outperforms standard JPEG coding. It better allocates the available rate among the different frequencies. The embedded, progressive nature of the coder allows encoding to stop at any desired rate. While the improvement is significant over JPEG coding, it still suffers from blocking artifacts at low coding rates which, is inherent to the block-based nature of DCT.

(a)



(b)

(c)



(d)

(f)

Figure 2.8: The original images: (a) *Lena*, (b) *Baboon*, (c) *Crowd*, (d) *Peppers*, and (f) *Barbara*.

Chapter 3

Coding with Spatially Variable Resolution

A modification of the SPIHT algorithm is presented for Region of Interest (RoI) coding that maintains its progressive (and embedded) property while adding the capability of a spatially adaptive allocation of bits. The bit allocation is based on a general importance function which defines the relative interest of the viewer to the various parts of the image. The algorithm is flexible enough to allow arbitrary changes of “importance” during transmission by sending that information through some feedback channel to the encoder.

RoI coding can be applied to transmission of images to facilitate fast recognition (by focusing the bit rate on coding important features of the image first) or to better preserve certain areas of the image (such as medical image compression [78]). In order to fulfill the fast recognition requirement the coder has to be able to progressively update the image as more bits arrive allowing the user to terminate transmission when it is determined the image does not meet the search criteria.

In schemes using scalar quantization the RoI can be emphasized by choosing finer quantizers in the preferred regions ([48]). With progressive coders that code the wavelet coefficients by bit planes, this finer coding can be achieved by magnifying the corresponding coefficients ([73, 67, 6, 51]) or by stopping the coding of coefficients outside the RoI after some refinement is accomplished ([56]). The RoI can be given offline by either human selection or by some feature extractor such as in [48] or it can be communicated interactively during the transmission. The proposed algorithm can be applied with both offline and online RoI selection.

3.1 Region Masking

The goal of region masking is to achieve improved perceptual quality in certain "important" regions (perhaps at the expense of reduced quality in other less important locations) while still using the same overall bit rate.

Our proposed modification of the SPIHT algorithm achieves this by idling those coefficients that do not correspond directly to the specified regions. Thus for a few iterations of sorting and refinement passes all the bandwidth is assigned to important areas. For proper communication, both the encoder and the decoder must maintain the same set of active coefficients. In our proposed method only the region description has to be communicated to the decoder and everything else can be computed locally.

To characterize the relative importance of each pixel location in an image, we define an *importance function* i that assigns a value $i(x, y) \in [0, 1]$ to each pixel (x, y) in the spatial domain, such that its sum over the pixels in the image is 1 (i.e. i is a probability mass function). A large value of $i(x, y)$ is used to indicate that higher resolution is desired for pixel (x, y). Initially a uniform importance function is used, and then its value is updated each time a change in preference occurs.

The quantization and coding in SPIHT occurs in the wavelet domain, and hence it is necessary to convert the spatial domain importance description into a bit resolution description for each coordinate (u, v) in the wavelet domain. We define a mapping I, that assigns to each pair (u, v) in the wavelet domain, a number $I(u, v) \in [0, 1]$. The mapping I is derived deterministically from the importance function, i. For a given coefficient with coordinates (u, v) at level l, the value of $I(u, v)$ is chosen to be the average value of $i(x, y)$ taken over the set of all pixels (x, y) in the $2^l \times 2^l$ block in the spatial domain, that are mapped to (u, v) via the wavelet transform. Other selections for $I(u, v)$ could include the maximum, median, minimum of the $i(x, y)$. Due to the nature of the subband decomposition each subband will contain coefficient(s) corresponding to the selected region(s). Since SPIHT does not allocate bits uniformly to each wavelet coefficient, the relative importance of the pixels is not matched exactly by the relative transmission rate assigned to them.

To relate the notion of "importance" to wavelet domain resolution, we delay the processing of wavelet domain pixels associated with unimportant regions in the SPIHT algorithm (i.e. no bits are spent coding the delayed pixels). The amount of delay, $t_d(u, v)$, (measured in the number of sorting/refinement pass cycles) is determined based on the $I(u, v)$ values for each coefficient as

$$t_d(u, v) = \left\lfloor 4 \frac{I_{max} - I(u, v)}{I_{max} - I_{min}} \right\rfloor ,$$

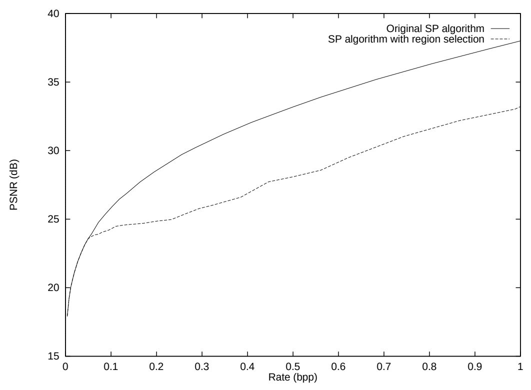
where I_{max} and I_{min} are the maximum and minimum of the wavelet domain importance values, respectively. If the importance function changes during an iteration, the delay values are re-evaluated.

If the importance function changes during encoding, that new information must be provided to the decoder. The importance information needs to be placed in the bit stream carefully to avoid confusion with image data. It can be placed either after a constant number of bits each time or at points where the sorting or the refinement pass ends. The encoder can use 1 bit at each such point to indicate whether the next n bytes contain control information or more image information. In the case where this change is prompted by a request from someone downloading an image, the control information can simply be some acknowledgment of receiving the request. (Because the feedback medium to the encoder may not be reliable the acknowledgment needs to indicate to which request it is referring.) It is important that the encoder not start to use the new importance function before that control information (or the acknowledgment) is sent to the decoder to keep the two sides consistent.

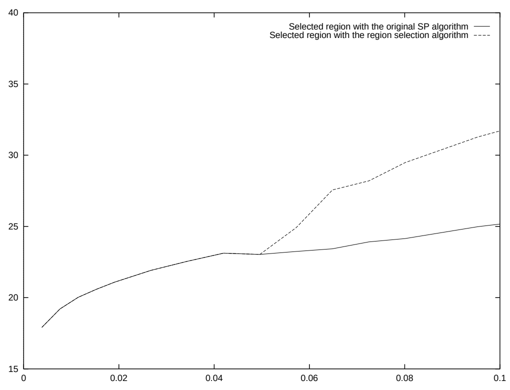
3.2 Simulation Results

In this simulation we assumed the following situation: someone is downloading an image and during transmission can select regions using a mouse. A single mouse click is translated into adding a two dimensional Gaussian lobe centered at the mouse click with variances $\sigma_x = \sigma_y = 50$ to the importance function. This creates a smoother transition in image quality around the RoI when compared with a step-function like characterization of importance. Since the natural synchronization points (end of sorting/refinement pass) are getting placed farther apart in the bit stream as the algorithm progresses, we chose to use synchronization points after a predetermined number of information bits for faster feedback.

Figure 3.2 shows 3 snapshots of both SPIHT and of the proposed algorithm. The SPIHT images are shown in the left column (images (a), (c) and (e)). The region in the mid-right part of the image was selected at a bit rate of 0.05 bpp. It can be seen on images (a) and (b) that shortly after the selection (at 0.11 bpp) the head in the selected region is recognizable with the region selection, while still blurry on the other image.



(a)



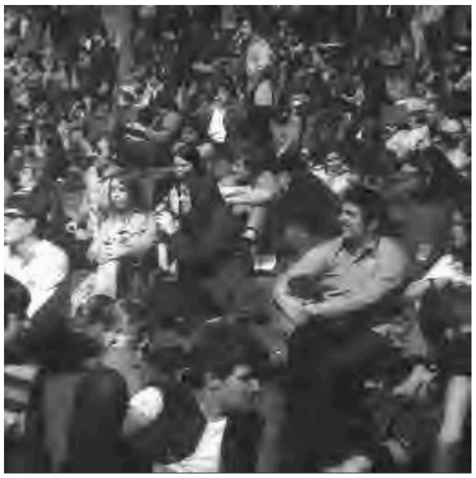
(b)

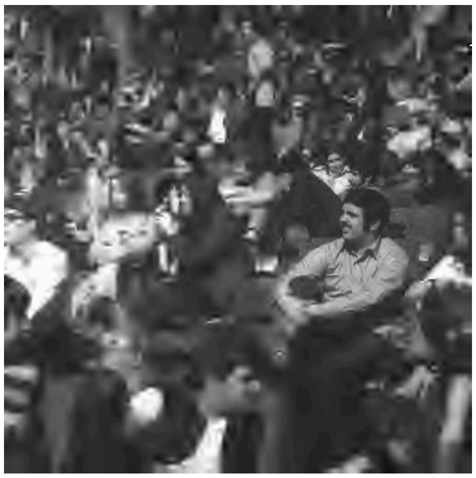
Figure 3.1: The PSNR comparison between SPIHT and the algorithm with region selection for (a) the entire image, (b) for the region selected in the mid-right portion at 0.05 bpp.

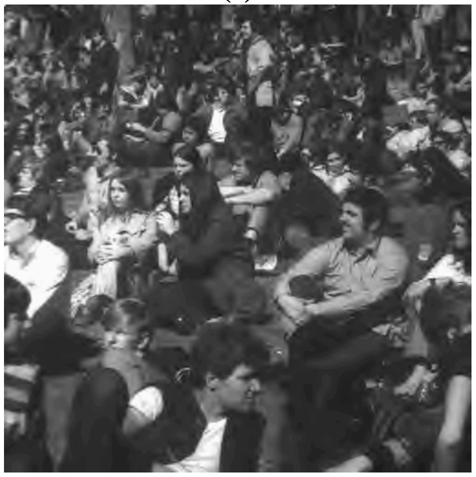
The next pair of images shows the situation at 0.23 bpp shortly after the region around the dark haired girl in the middle is selected. Finally the last pair shows the result at 1 bpp. At this rate the real difference between the two images is the resolution/image quality toward the sides of the image. On the image with the region selection, (f), the regions around the image edges are still a little blurry, because less bandwidth was allocated to these parts.

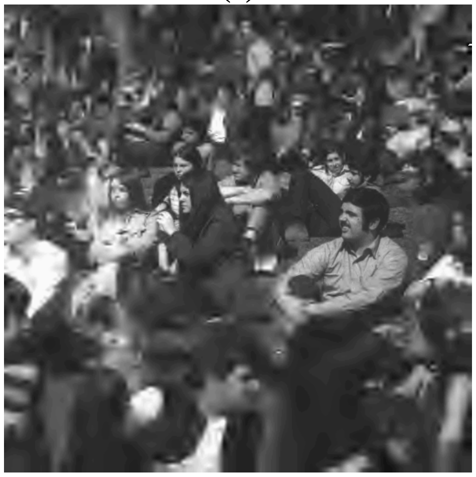
In applications where early recognition is important, the coding of coefficients in the RoI can be terminated once the refinement reaches a stage where further bits could only yield unnoticeable improvements in image quality. This strategy would allow the image to achieve a more uniform image quality at the target rate.

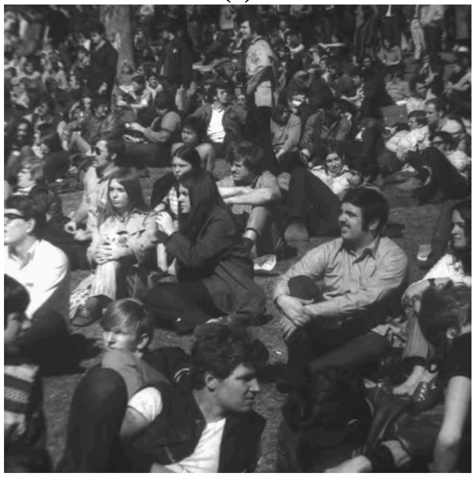
A more quantitative comparison is shown in Figure 3.1. Part (a) compares the PSNR of the same image encoded with SPIHT and with the regions selected as described above. The PSNR with the region selection on the overall image is about 5 dB inferior to that of SPIHT. But as shown in part (b) for only the selected region in the mid right section, the PSNR in that region improves very fast to about a 6.3 dB improvement (at 0.1 bpp) over SPIHT in that same region.


(a)


(b)


(c)


(d)


(e)

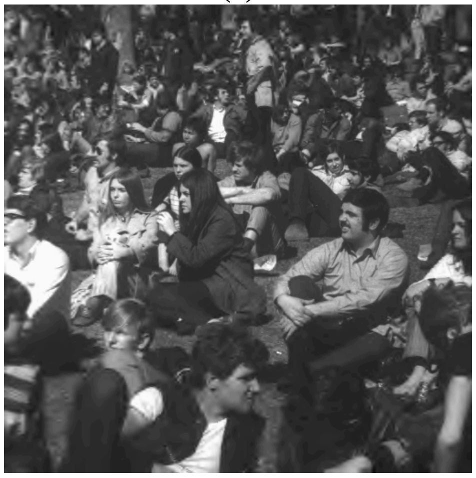

(f)

Figure 3.2: The sequence of images at different rates with SPIHT (left) and the algorithm with region selection (right), (a)-(b) 0.11 bpp, (c)-(d) 0.23 bpp, (e)-(f) 1 bpp.

3.3 Summary

In this chapter we described a flexible mechanism to handle RoI coding. We proposed a modification of the SPIHT coder which was chosen for its good compression performance and progressive nature that is necessary in fast recognition applications.

The importance function we introduced coupled with the encoding method allows for flexibility in changing the RoI in the image during an interactive session. Methods that rely on scaling of coefficients in the wavelet domain run into difficulties as the “importance” preferences change. In our proposed method coefficients that have been delayed can be activated and vice versa at each update of the importance function.

Our results demonstrated significant improvement within the RoI soon after the selection has been made. The trade-off for improved quality inside the RoI is some degradation elsewhere in the image. This can be mitigated by delaying the coefficients inside the RoI once their refinement reaches a certain quality.

This chapter, in part, is a reprint of the material as it appears in: T. Frajka, P. G. Sherwood, and K. Zeger. Progressive Image Coding with Spatially Variable Resolution. *International Conference on Image Processing*, pp 53-56, Vol. 1, Santa Barbara, CA, Oct. 1997. The dissertation author was the primary investigator of this paper.

Chapter 4

Image Coding Subject to Memory Constraints

Limited data transmission rates between a computer and a color printer can be the bottleneck in producing rapid prints of digitized images. This could result from low transmission rate connections, network congestion to centralized printing, especially when high quality, high resolution images are concerned.

Image compression techniques can greatly speed up the printing process if the images can be decompressed on-board the printer, but there are two difficult constraints that must be met that are not typically both found in other image compression problems: (1) *the amount of memory on-board the printer is limited, and (2) the decompression algorithm must progress from the top of the paper to the bottom of the paper, in the order that the paper emerges from the printer.* Typically, there is also a complexity constraint for the decoder. These constraints are derived from the practical issues of designing competitive low-cost color printers for home use and for businesses.

Wavelet-zerotree-based compression algorithms such as EZW and SPIHT have excellent distortion-rate performance, but do not meet the memory constraints and sequential processing needs of printers; they require a complete image to be reconstructed before any particular part of the image can be printed by a printer. Their progressive property is not useful here; unlike a video monitor, low cost paper printers cannot return to and change earlier rows, once the paper has begun exiting the printer. The full quality decoding for the top row of an image must be completed before any printing can begin. This constraint is already satisfied with baseline sequential JPEG, in which the image is processed in blocks of 8×8 pixels across each row of blocks starting from the upper left corner of the image. However, many wavelet-based algorithms have much better distortion vs. rate performances than the JPEG technique and also yield visually better image quality.

In [61] the image is processed in independent strips with a small number of horizontal rows and each strip is compressed using the EZW algorithm. While this technique yields lower memory complexity, its performance is significantly poorer than compressing the full image. The degradation in performance is mainly due to the discontinuities at the strip boundaries and the smaller context available to the adaptive arithmetic coding engine. In [19, 21] the wavelet transform is carried out over the entire image, but the coefficients are reordered to allow decoding from the top to the bottom. Chrysafis and Ortega [17] introduced a line-by-line wavelet coding scheme where both the transform and the encoding/decoding is limited to working with a minimum set of lines of wavelet coefficients. Instead of using a zerotree structure their method relies on context modeling to achieve good compression performance.

We present a method for ordering the wavelet coefficient information in a compressed bit stream to allow a decoded image to be sequentially decompressed. In addition, we use a hybrid filtering scheme that uses different horizontal and vertical filters, each with different depths of wavelet decomposition. We provide formulas for computing the memory requirements due to the choice of filters, wavelet decomposition levels, and image coding method. We also incorporate a modified version of the quadtree-guided wavelet encoder in order to reduce the memory requirement imposed by the zerotree structure.

4.1 Memory Requirements

The memory requirements at the decoder are determined by certain properties of the wavelet transform and the image coder.

Only a small subset of the wavelet coefficients is required in the inverse transform to obtain any given output row. This is illustrated in Figure 4.1. This process can be described in terms of line-by-line printing in the spatial domain of the image. We seek the *minimum set* of wavelet coefficients that must be received by the decoder in order to print out a given single horizontal row in the reconstructed image. After the printer has received that minimal set, and used it to print out one particular row in the image, we then seek the *minimum set of additional wavelet coefficients* that must be transmitted by the encoder so that the following row can be printed. We also determine how much of the currently stored set of coefficients can be purged from memory. Coefficients that can be expunged are those which are not needed to print out a future row which has not yet been printed. This process is iterated for each row until the entire image of size $M \times N$ is printed.

In the case of wavelet based coders three major factors determine this minimum set of wavelet coefficients: the length of the wavelet filters, the number of decomposition levels and the coding algorithm.

The filter length becomes an important factor in the vertical inverse transform. To be able to recover any single line one needs to receive as many lines in the first level of decomposition in the wavelet domain as there are taps on the filter. Some of these lines

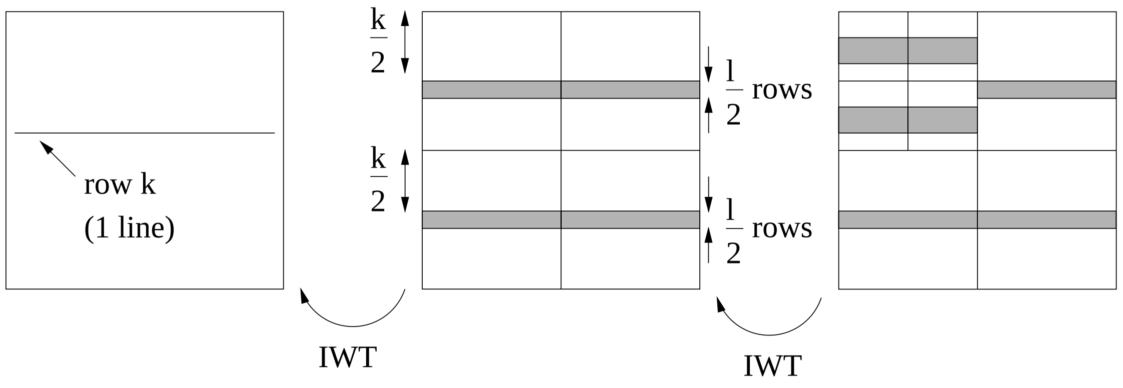


Figure 4.1: On the left, the printer is ready to print out row k of the output image. The middle picture shows strips of wavelet coefficient rows that are required at the first level of decomposition. On the right are strips required at the second level of decomposition.

are in the vertically lowpass filtered band, and some are in the highpass filtered band. Thus shorter filters would help reduce the memory requirements. Unfortunately the use of shorter filters often results in a decrease in image quality. Filters cannot be too short because that would lead to less smoothness as observed in [3]. Smooth reconstruction filters are desired for good visual image quality.

As noted above, some of the lines that feed the filter taps come from the LL band. In the case where more levels of wavelet decomposition are done, one has to do several levels of inverse wavelet transforms to obtain the lines used at the last level of the inverse filtering. Generally at each level the same filters are used, thus contributing the same number of rows at each level that are necessary for a single row at the next higher level (although the rows get shorter horizontally in the lower frequency subbands of the decomposition). The memory requirement can be determined recursively for each level given the number of rows at the next higher level.

In addition to the memory requirements induced by the nature of the wavelet transform, the order in which the wavelet coefficients are coded also influences the memory needs. Because zerotree-based coders tend to send quantization information that pertains to large groups of coefficients at one time, the decoder receives and must store information about many coefficients that are not needed to reproduce the first line of the original image. Methods that use context modeling to encode wavelet coefficients would require that the decoder store the coefficients that form the context. In [46] Liu and Moulin showed, using information theoretic arguments, that intrascale modeling (using coefficients from the same level of wavelet decomposition) results in higher mutual information than interscale modeling (exploiting parent-child relationships). Using only coefficients from the same subband in forming the context can greatly reduce the memory requirement. However, their results indicate that using intrascale modeling alone is suboptimal; the gains in memory complexity are achieved at the price of compression performance.

4.2 Bit stream Re-ordering

Bit stream re-ordering is a simple way of reducing the memory requirement of otherwise memory-complex compression methods. It allows the decoder to receive information about coefficients in the top to bottom order required for line-by-line printing.

The re-ordering of the SPIHT bit stream can be thought of in terms of our work in Chapter 3. In our low-memory printer application, the RoI can be seen as the actual row(s) to be printed. Bits are sent in an order such that the decoder receives a succession of “minimal sets” of coefficients corresponding to successive printing rows. In order for the decoder to correctly determine where the information concerning the region of interest ends in the bit stream, a header (usually small) is required; the encoder describes the threshold T_n at which the encoding of the original algorithm (not re-ordered) terminates for the region. The decoder then evaluates, as it decodes each row in the re-ordered bit stream, whether the received bits correspond to a threshold of T_n. Once it passes that threshold, the decoder deduces that the data corresponds to the next row.

While this method only incurs a small overhead penalty, it can only stop coding each minimal set of rows when all corresponding coefficients have been refined to the given threshold T_n. This limits the achievable compression ratios to a small set of coding rates. By transmitting the coordinates of the coefficient where the coding ends at threshold T_n, the encoding can stop at any arbitrary point in the sorting/refinement cycles.

Given L levels of wavelet decomposition, using the zerotree structure, the memory requirement is

$$\begin{aligned} S_{zt}(k) &= kN2^{-L} + \sum_{l=1}^{L} 3kN2^{-l}2^{L-l} = kN2^{-L} + 3kN2^{L}\sum_{l=1}^{L} 2^{-2l} \\ &= kN2^{-L} + kN2^{L}\left(1 - 4\left(\frac{1}{4}\right)^{L+1}\right), \end{aligned} \tag{4.1}$$

where k is the number of rows needed from the LL band. $S()$ gives the number of wavelet coefficients instead of the number of rows. It is more accurate than a row count as the rows get halved at each new level of decomposition.

4.3 Quadtree-Guided Encoding

The quadtree-guided wavelet compression technique can achieve a substantial reduction in the memory requirements because it uses only a single level of wavelet decomposition with encoding relying on $k \times k$ blocks of wavelet coefficients instead of the zerotree structure.

To make it more suitable for the printer problem, the zig-zag encoding order in the outer bands is changed to a strictly horizontal ordering for the runlength coding which is shown in Figure 4.2. The original scanning order is depicted in Figure 2.6. (For the images tested, the modification of the outer band scanning order resulted in a

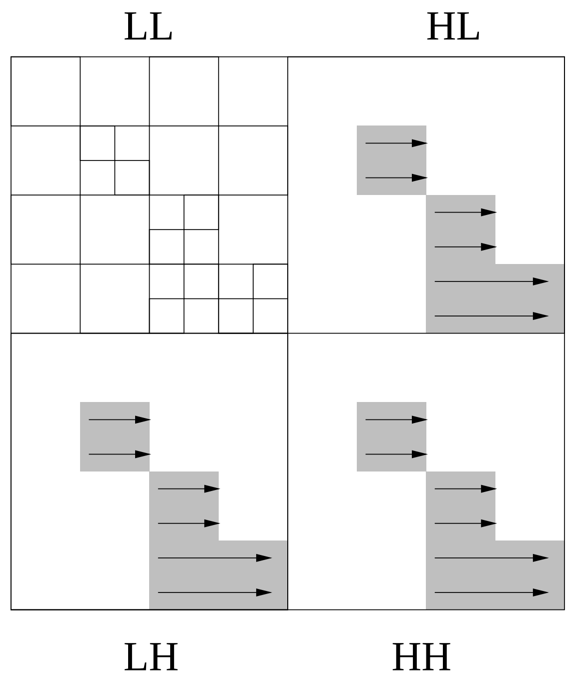


Figure 4.2: Modification of scanning order of outer band coefficients to horizontal scanning order to match the line-by-line processing order of a low-cost printer application.

3-8% decrease in outer band rate compared to the zig-zag ordering.) Using this type of encoding one only needs to store

$$S_{qge}(k) = 4kN/2, \tag{4.2}$$

coefficients, where k is the initial block size of the quadtree coding, which is also the number of rows required from the LL band for this technique.

4.4 Hybrid Filtering

More wavelet coefficients are required to produce a single output row when either the filter length increases, or when the number of decomposition levels increases.

Let $r_L(l)$ and $r_H(l)$ denote the number of rows of coefficients required from level l of the decomposition in the vertical low and high frequency bands, respectively, to reconstruct 1 row of the output image . Let the 0^{th} level denote the spatial domain image and let $r_L(0) = 1$, $r_H(0) = 0$. Then the following recursive formulas give the values of $r_L(l)$ and $r_H(l)$ as a function of the maximum of the low and highpass filter lengths at level l, k_l.

$$r_L(l) = \left\lceil \frac{k_l}{2} \right\rceil + \left\lfloor \frac{r_L(l-1)-1}{2} \right\rfloor \tag{4.3}$$

$$r_H(l) = \left\lfloor \frac{k_l}{2} \right\rfloor + \left\lceil \frac{r_L(l-1)-1}{2} \right\rceil \tag{4.4}$$

Using (4.3) and (4.4) the number of coefficients needed to recover 1 line in the spatial domain is

$$S_f = r_L(L)N2^{-L} + \sum_{l=1}^{L} (r_L(l) + 2r_H(l))N2^{-l}, \tag{4.5}$$

where L is the maximum number of decomposition levels. The first term in (4.5) represents the LL band, the first term of the sum represents the HL bands, while the last term of the sum represents the LH and HH bands.

With a single level of filtering with the 2-tap Haar filters, $S_f = 2N$, the same scenario with the 9-7 filters yields $S_f = 9N$, a 4.5 fold increase.

Filtering in the horizontal direction has no effect on the memory requirements, and thus one can maintain in the horizontal direction the full 6 levels of decomposition using the 9-7 filters. In the vertical direction, we can, for example, perform 2 levels of decomposition with the 9-7 filters, and 4 levels of filtering with the Haar filters. The total number of levels of decomposition does not have to be the same in the two directions. There exist many different possibilities, each presenting its own trade-off between distortion, rate, and buffering requirements.

Table 4.1: The values of r_L and r_H from (4.3) and (4.4), respectively, for the 9-7 filters up to 6 levels of decomposition.

	wavelet decomposition level					
	$l = 1$	$l = 2$	$l = 3$	$l = 4$	$l = 5$	$l = 6$
$r_L(l)$	5	7	8	8	8	8
$r_H(l)$	4	6	7	8	8	8

4.5 Results

In our experiments we used 512×512 24-bit color images. To gain a better understanding of memory requirements we compare different coding algorithms and filter choices in terms of the percentage of all coefficients, $(S()/(MN)) * 100$ %, instead of actual coefficient count.

Table 4.1 shows the values of r_L and r_H with $r_L(0) = 1$ for up to 6 levels of the wavelet decomposition using the 9-7 filters. For the Haar filters these values are easily computed based on the inverse filtering operation as $r_L(l+1) = r_H(l+1) = r_L(l)$. In the inverse transform, Haar filters take one coefficient from both the low- and high-frequency bands to return one coefficient at the next level.

With a single level of filtering with a 2-tap filter, only 0.39% of the wavelet coefficient array is required to produce an output row (512 pixels). If one replaces the Haar filters with the 9-7 filters, the memory needed grows to 1.7%.

With full 6 levels of decomposition using the Haar filters, only 0.57% of the coefficient array is involved in producing a single output row. However, the decrease in SPIHT performance is very significant when Haar filters are substituted for the 9-7 biorthogonal filters. The decrease in SPIHT performance is also very significant when the number of decomposition levels drops below 4. If 9-7 filters were used instead with 6 levels of decomposition, 3.2% of all coefficients would be needed. These numbers reflect only the memory use induced by the filter choice, they do not take into account further constraints imposed by the encoding method.

The memory constraint imposed by the encoding method shows that the zerotree structure requires larger memory storage of coefficients than the quadtree-guided coding. With taking only zerotrees that are rooted in the first row of the LL band ($k = 1$ in (4.1)) and 6 levels of decomposition for the zerotree structure, the memory requirement is 12.5% of all coefficients. A 5 level decomposition would reduce this amount to 6.25%. Using the quadtree-guided coding with 8×8 blocks of coefficients gives $k = 8$ in (4.2) and memory requirement of 3.1%.

To obtain the full picture one needs to consider constraints arising from all three factors together. The quadtree-guided coding uses only 1 level of decomposition. Even using 9-7 filters, one only needs 5 rows of coefficients from the LL band to recover a single output row. That is still less than the 8 rows imposed by the encoding method, thus the memory requirement is dominated by that factor and stands at 3.1%. For a zerotree based method such as SPIHT with 6 levels of decomposition the 9-7 filters choice would dictate $r_L(6) = 8$. Substituting that value for k in (4.1) gives 100% of all coefficients as memory requirement. Replacing the 9-7 filters with the Haar filters with same 6 levels imposes $r_L(6) = 1$ and a 12.5% memory requirement. From Table 4.1 it follows that if a 6-level decomposition is used, any combination of 9-7 and Haar filtering with more than 2 levels of 9-7 filtering requires the full wavelet domain image to be stored at the decoder. ($r_L(6) = 8$ would follow from any such scenario.)

PSNR results are obtained for the 24-bit color *Lena* image compressed to 0.24 bpp (i.e. $100 : 1$ compression ratio), for both quadtree-based and zerotree-based systems. For zerotree-based systems the horizontal filtering is maintained at 6 levels of decomposition using the 9-7 filters, as in the SPIHT algorithm. The vertical filtering also uses 6 levels of decomposition, but anywhere from 0 to all 6 of those filtering rounds

Table 4.2: Comparison of different methods for memory requirement and compression performance.

Method	Memory Requirement	PSNR (dB)
SPIHT original	100%	30.77
SPIHT w/ Haar filtering	12.5%	29.60
QGE	3.1%	29.52

can be done by 9-7 filters; the remainder use the Haar filter. In the case of quadtree coding, only 1 level of filtering is done using 9-7 filters. The trade-off between memory requirement and compression performance is shown in Table 4.2. The original SPIHT algorithm produces the best result, but requires 100% of the coefficient array in decoder memory. With 6 levels of vertical Haar filtering and the bit stream re-ordering, a 1.1 dB drop in quality is the "penalty" for having to store only 12.5% of wavelet coefficients in decoder memory. The quadtree-based scheme, on the other hand, uses only 3.1% of buffering, but with 1.2 dB drop in PSNR. It is thus comparable in PSNR to the zerotree method with all 6 levels of Haar filters vertically, but it uses much less memory and the visual quality is more pleasing, too. (See Figure 4.3.)

4.6 Summary

In this chapter we examined the relationship between different aspects of an image compression technique and its memory complexity. Certain applications may limit the available memory at the decoder that the coding method must take into account.

We gave formulas for the number of coefficients required by the choice of wavelet filters, the number of decomposition levels, and the coding method. Our experimental results indicated that memory requirement is traded off for compression performance. The modified quadtree-guided coding achieves similar results as the re-ordered SPIHT

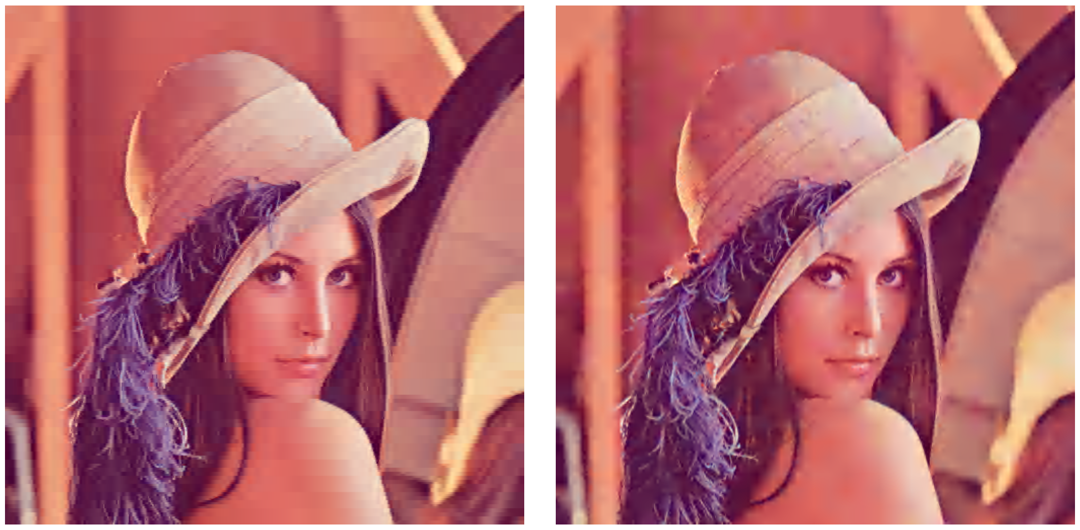


Figure 4.3: Comparison of (a) SPIHT coding using 6 levels of Haar filtering and (b) quadtree-guided encoding. Lena image at 0.24 bpp.

coding method with 6 levels of Haar filtering with improved visual quality, with 75% lower memory complexity. They both lag over 1 dB behind the 9-7 filter implementation of SPIHT which requires 100% storage.

We also showed that hybrid filtering can reduce the amount of memory needed. But if it is coupled with zerotree-style coding reduction can only be achieved with fewer than 3 levels of 9-7 filtering.

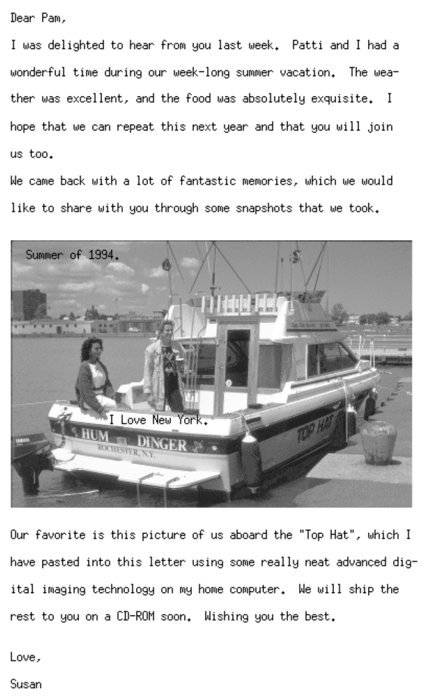
This chapter, in part, is a reprint of the material as it appears in: P. Cosman, T. Frajka, and K. Zeger. Image Compression for Memory Constrained Printers. *International Conference on Image Processing*, pp 109-113, Vol. 3, Chicago, IL, Oct. 1998.

Chapter 5

Image Coding for Compound Images

We present two variations on a quadtree wavelet-based coder designed for improved performance on compound images. We segment the image to identify blocks containing text, which are then treated specially. One coder operates entirely in the wavelet domain, applying separate coding parameters to text and non-text blocks. In the second variation, the coder combines wavelet-domain processing of non-text blocks with spatial domain processing of text blocks. Both variations provide improved performance over standard wavelet methods when applied to compound images.

Traditional image coding focused on natural images. With the latest developments in communications images that contain both text, drawing, charts and natural images became more significant. Such images are often referred to as compound images or document images. The images used in our simulations are show in Figure 5.1. Text in an image can be far more visually important to a human viewer than might be deduced from summing the energy of the text pixels themselves. Distortion in the edges of text characters, caused by lossy compression of the image, can be more annoying than the same type of distortion in other areas of the image. Unfortunately, wavelet-based image coders suffer from just this deficiency when used on compound images. They often focus on improving low frequency information while allowing high frequency edges, such as the sharp edges of text characters, to blur.



Dear Pam,

I was delighted to hear from you last week. Patti and I had a wonderful time during our week-long summer vacation. The weather was excellent, and the food was absolutely exquisite. I hope that we can repeat this next year and that you will join us too.

We came back with a lot of fantastic memories, which we would like to share with you through some snapshots that we took.

Summer of 1994.

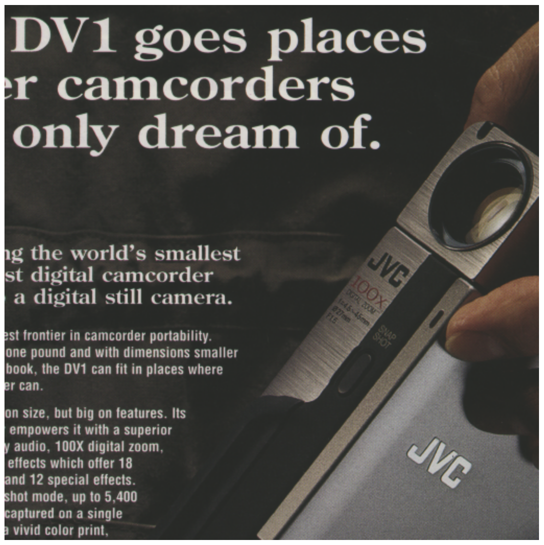
I Love New York.

Our favorite is this picture of us aboard the "Top Hat", which I have pasted into this letter using some really neat advanced digital imaging technology on my home computer. We will ship the rest to you on a CD-ROM soon. Wishing you the best.

Love,

Susan

(a)



(b)

Figure 5.1: Original compound images (a) *cmpnd1* greyscale image, (b) *camcorder* color image.

Zeng et al. in [95] propose an adaptive wavelet transform that uses different filters in different parts of an image depending on the entropy characteristics of the image region. In [41] filters are designed specifically to reduce the blurring and ringing effects around sharp edges of text. Their filter design aims at limiting the spatial domain undershoot and overshoot in the side lobes of the basis functions. The visually disturbing side effects can be avoided using filters with high spatial localization property. Another proposed solution ([42]) introduces a metric based on quantized DCT data to classify image blocks as having low or high activity. In a JPEG-compliant fashion blocks with high activity are more finely quantized to improve the coding of text regions in an image.

A conceptually different technique relies on the Mixed Raster Content (MRC) ITU standard [33] that represents the image as three different layers: full color continuous tone *background* layer, full-color or limited color *foreground* layer, and a binary selector layer. The selector layer determines for each pixel whether that belongs to the text area (foreground) or the smooth image (background). In this framework the images are segmented into foreground and background layers prior to coding. [16, 32, 22] present such coding schemes with the emphasis being on finding close to optimal segmenters.

Haffner et al. presents DjVu ([29]) that uses the same MRC context for document images that are typically of high resolution (300 dpi or better). They use a variation of the JBIG2 bi-level compression algorithm (dubbed JB2) to encode the foreground map using soft pattern matching. The background coding is done with the IW44 wavelet image coder [11]. This method is aiming at encoding the background image without wasting bits on pixels that are removed as text information. It is achieved by an iterative algorithm that changes the wavelet coefficients in successive projections. These are used in conjunction with the ZP-coder [10] adaptive arithmetic coder for improved compression efficiency.

The filling in of removed text information is investigated by de Queiroz in [23]. The spatial and DCT domain iterative algorithms try to fill in the data in a rate-distortion optimal way.

5.1 Text Segmentation

In this chapter the coder variations we describe rely on the segmentation of images into text and non-text regions. Our focus is on the coding of the segmented image, not the segmentation itself. Any block-based segmenter may be used for this purpose. Our implementation uses a relatively simple procedure based on decision trees.

Training images are divided into 8×8 blocks. For each block, 11 parameters are computed from the 64 pixels in the block. Among these are the row variance, column variance, 3rd and 4th moments, and DCT coefficient energy. The CART (Classification and Regression Trees) algorithm by Breiman et al. [14] is then used to construct a binary decision tree based on the parameters computed from the training images. Each leaf node of the tree represents either a text or non-text outcome.

At each stage in growing the tree, CART considers which node to split next by considering every parameter at each of the current leaf nodes. The node, parameter, and parameter decision threshold which yield the most accurate partition of the training data are determined, and that leaf node is split. CART grows a large tree, then employs optimal pruning to reduce it to the desired size.

Given the 8×8 block size, one bit per 64 pixels would be required to describe the segmentation map. This information is arithmetically encoded using a causal context of four neighbor blocks; it typically adds less than 0.01 bpp to the overall compressed bit rate.

5.2 Wavelet-Domain Coding of Text Blocks

The quadtree wavelet coder presented in Section 2.3.4 performs well on natural images. As discussed in Chapter 4 a modified version of this coder requires less memory at the decoder than many other wavelet coders.

We present a modification of the above coder to allow separate parameters to be used on text and non-text blocks. The encoder first classifies each 8×8 image block as either text or non-text, and sends this segmentation map to the decoder. A one-level wavelet decomposition is performed and the coding is similar to that in Section 2.3.4.

In our approach, an LL-band block is considered “text” if all four corresponding spatial blocks were classified as text. Because of the 1-level wavelet decomposition, every 8×8 block in the LL band corresponds to four 8×8 blocks in the spatial domain. Four parameters are selected for each LL-band block depending on its classification as text or non-text:

- The wavelet transform filters. For text blocks, short filters such as Haar filters are chosen to improve response to sharp edges. At the boundaries between text and non-text blocks the respective filters extend beyond the given block. A possible solution is to describe some coefficients twice to provide data for the filters beyond the block boundary. While this solution guarantees smoother transition between blocks with different filters, it is redundant. Instead we propose to reflect the coefficients to the other side of the boundaries between text and non-text blocks to provide the right type of pixels for the filter taps. This is similar to the boundary effect at the image edges where the filters expand beyond the actual image and does not result in any redundancy of the encoding.

- The quantizer for the foot prediction error. Since the algorithm is trying to predict the entire block using the quantized foot and coefficients from neighboring blocks, finer quantization is used in text blocks to achieve a better prediction for those blocks.

- The block error threshold. A lower threshold is chosen for text blocks forcing finer prediction in those blocks as the algorithm subdivides blocks where the prediction error falls below the lowered threshold.

- The quadtree block shape. Text blocks are subdivided into horizontally oriented rectangular sub-blocks (8×4, 4×2 or 2×1) which were found empirically to perform better than square sub-blocks in text areas.

The overall effect of these selections is that more bits are invested in text blocks. For any given rate the quality in the text region is improved at the expense of the quality in the non-text regions.

5.2.1 Results

Our simulation results show that treating coefficients corresponding to text blocks differently from non-text blocks can improve (perceptual) image quality. Figure 5.2 shows the comparison between our proposed method, regular quadtree coding and the EBCOT algorithm for a portion of the *cmpnd1* greyscale image at 0.24 bpp. Our proposed method using improved text region coding displayed here uses Haar filters in the text regions with horizontal quadtree block shape and finer quantization. The PSNR values shown are computed over the entire image. Our coder produces sharper text and more detailed background for the image than the other two coders. As these images demonstrate, PSNR is not a good measure of image quality when it comes to compound images. The overall PSNR in the EBCOT-coded image is higher because in the image coded with our proposed algorithm there is a slight seepage of gray in the white background above the photograph due to the predictive nature of the coder. While this error contributes to the MSE, it is visually not degrading to the image.

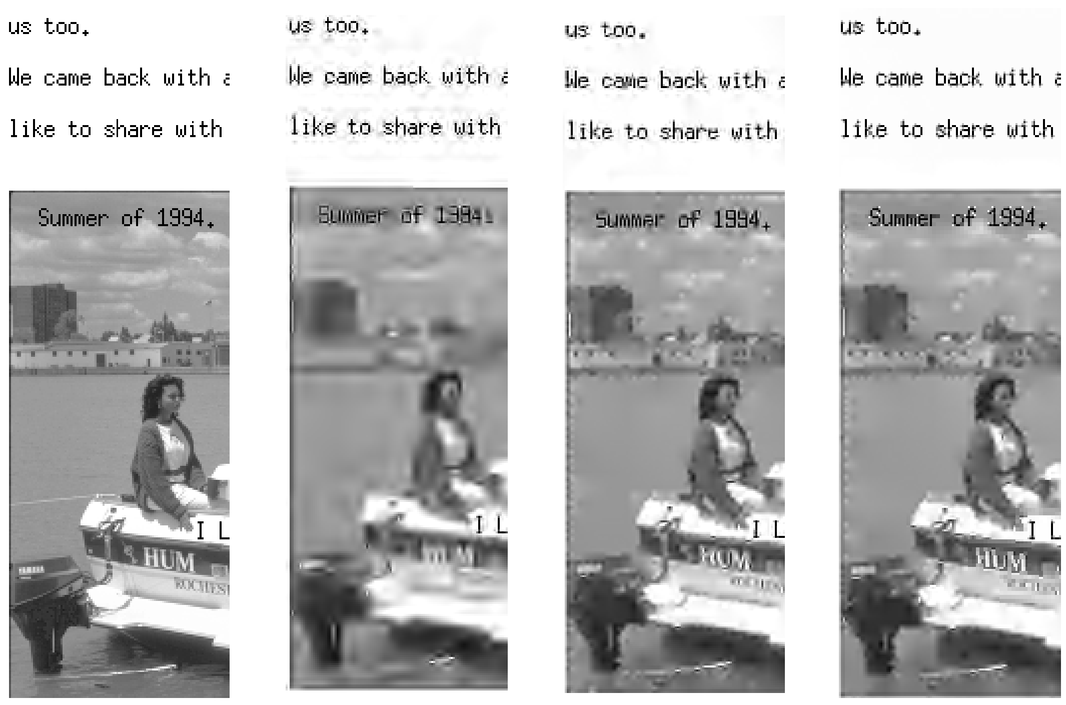


Figure 5.2: Portions of the *cmpnd1* images, (a) original, (b) EBCOT coded at 0.24 bpp, 26.48 dB, (c) Quadtree coded at 0.24 bpp, 25.08 dB, (d) Quadtree coded with improved text region coding at 0.24 bpp, 25.98 dB.

5.3 Combined Wavelet-Domain and Spatial-Domain Coding

Because of the sharp transition at edges and small feature size, text image data can benefit from processing in the spatial domain rather than in the wavelet domain. In this section we propose an approach which combines wavelet-based coding of non-text regions with spatial domain coding of regions containing text. The algorithm operates at the pixel level; that is, each individual text pixel is coded in the image domain, and all remaining pixels are coded in the wavelet transform domain.

Algorithm Summary: The image is first segmented into text and non-text blocks. The pixels belonging to text blocks are then labeled as either foreground or background (non-text), whereby the encoder chooses for each block which color – light or dark – will be called foreground, and which will be called background. The intensities for the foreground and background colors are then determined for each block. In a segmentation improvement step, some blocks which were previously identified as text blocks can be reclassified as non-text blocks, based on their foreground and background colors as well as those of their neighbor blocks. Following this step, the block segmentation map, the binary foreground/background map for the text blocks, and the quantized foreground intensity values are arithmetically encoded and transmitted.

At this point, both encoder and decoder possess knowledge of the text pixel locations and colors. It is therefore possible to “remove” the text (foreground) pixels to a large degree from the original image, by interpolating their intensities from neighboring non-text pixels. Such a step has the advantage of smoothing the original image, so that it can be more efficiently coded by the quadtree wavelet algorithm. After the text pixels have been removed by interpolation, the remaining image is wavelet-transformed and quadtree coded as discussed in Section 5.2. In the following sections we describe the steps of this algorithm in greater detail.

5.3.1 Text Block Color Classification

Within each text block X, the two main colors are identified with Lloyd-Max optimization ([47]) on a two-level scalar quantizer. If the two main colors are closer to each other than some threshold, the block is classified as unimodal - a block that only contains different shades of the same color - and is assigned to be text or non-text based on the foreground and background colors of the neighboring blocks. Otherwise the colors are designated as either foreground or background, based on information about the North and West neighbor blocks. Three cases are considered:

- If both the North and West neighbors are non-text blocks, their average pixel intensity $\overline{I}$ is determined. The color in X which is farthest from $\overline{I}$ is designated as foreground (text), and the other as background.

- If both the North and West neighbors are text blocks, the average of their foreground colors is determined, $\overline{f} = (f_N + f_W)/2$. The color in X which is closest to $\overline{f}$ is designated as foreground, and the other as background.

- If one of the North and West neighbors is a text block and the other non-text, the color in X which is closest to the foreground color in the neighboring text block is designated as foreground, and the other as background. However, if the foreground colors of the neighboring text block and of X are far apart, the color in X which is farther from the background color of the non-text neighbor is designated as foreground.

After the foreground and background colors have been designated, each pixel in block X is assigned to either foreground or background. To avoid distorting the text, only pixels with intensity within a given threshold from the foreground color are assigned to the foreground; the rest are treated as background.

5.3.2 Map and Foreground Color Coding

The block-level text segmentation map is transmitted as described in Section 5.1. The foreground/background map for each text block is then arithmetically coded with a causal context of neighboring pixels. Only pixels within text blocks are coded. The foreground color for a text block X is coded in a predictive coding fashion from the foreground values of the North and West neighbors, exploiting the fact that foreground values usually do not change significantly between neighboring blocks.

5.3.3 Interpolation and Residual Coding of Text Pixels

We now describe the procedure for removing text pixels from the image, in order to allow more efficient wavelet-based coding of the background pixels. The steps of this procedure are illustrated in Figure 5.3 for only a one-dimensional cross section along the non-text to text transition. The original image is shown in profile **A**, with a background area on the left and text on the right. In profile **B**, the text pixel values have been replaced by the block foreground color and in profile **C** they are replaced by values interpolated from neighboring background pixels. The image corresponding to profile **C** could be encoded by the wavelet quadtree coder, and combined by the decoder with the text pixel values sent from profile **B** to reconstruct the image. This reconstructed image would be an approximation of profile **B**. The sharp text edges thus obtained can cause the edges of small text characters to appear jagged, however. Instead, we include several additional steps to allow the residual error to be corrected, reducing any jagged appearance.

Profile **D** shows the residual, or difference between the original image and block foreground color in text areas. This residual is added to profile **C** to yield profile **E**, which is then encoded by the wavelet quadtree coder.

At the decoder, a lossy version of **E** is reconstructed, as shown in profile **F**. The decoder also receives the locations of text pixels and the quantized foreground color for each block, which it uses to produce profile **G**. Given the text pixel locations, the decoder interpolates the values of text pixels from neighboring background pixels to obtain profile **H**, which is an approximation of the encoder's profile **C**. The residual **I** is reconstructed as the difference between profiles **F** and **H**. By adding this residual to **G**, the decoder generates the output image profile **J**.

The significance of adding the residual to the background image lies in the ability of the method to losslessly encode the compound image. Without this step the decoded

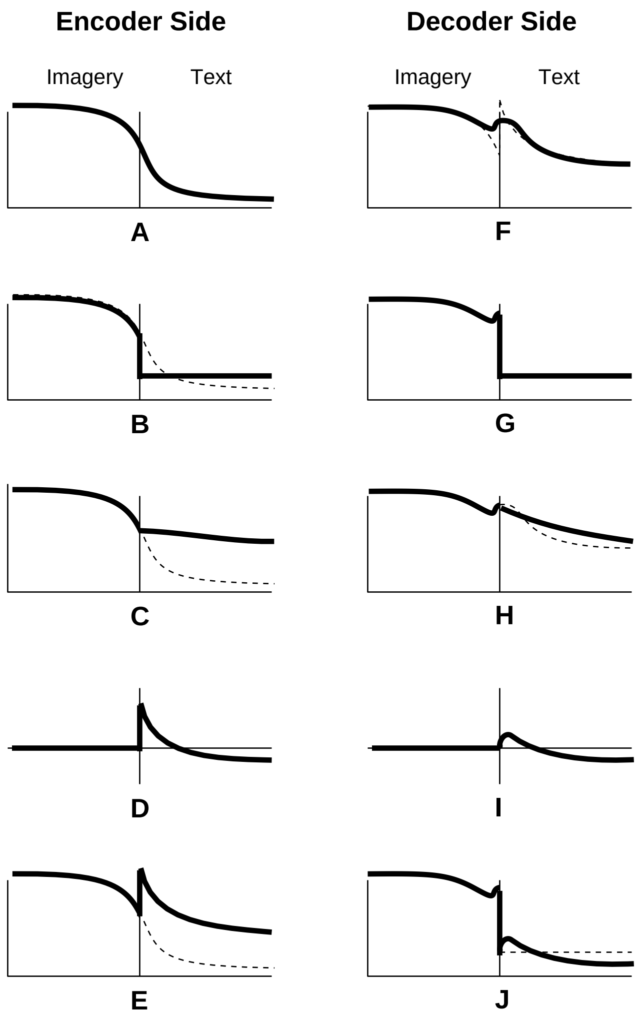
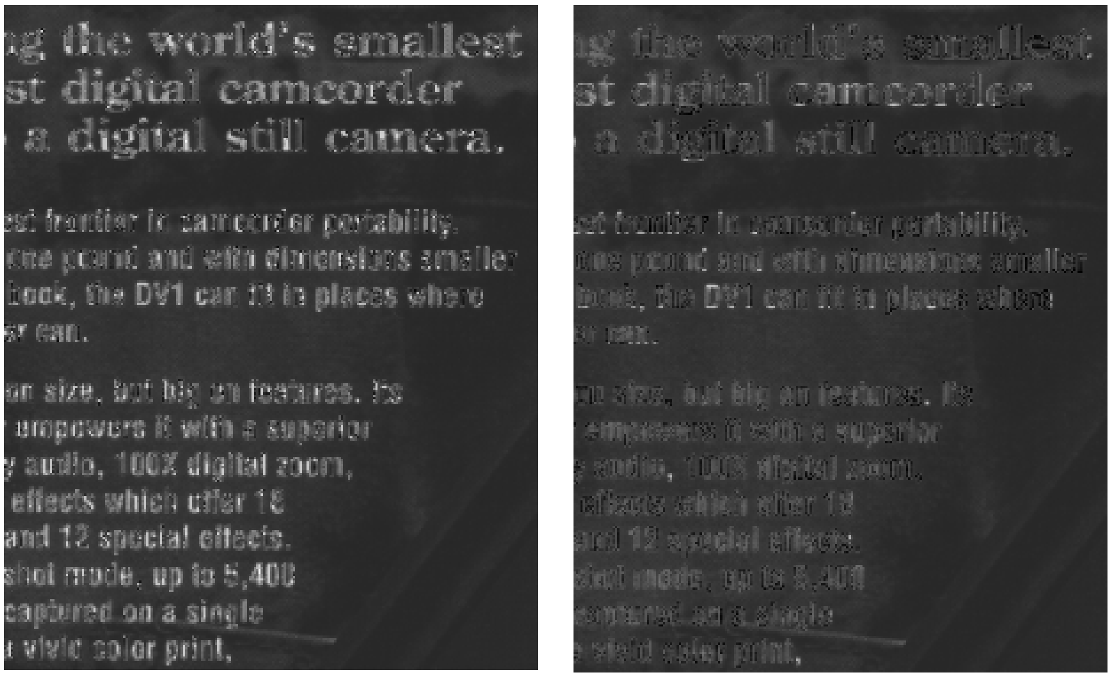


Figure 5.3: Procedure for removing text pixels from the image to be wavelet-encoded. See description in Section 5.3.2.

image would always be lossy no matter how high the coding rate is. The foreground color for each block is represented by a single value which inevitably distorts some of the intensities in that block. Since the description of the segmentation and the coding of the foreground colors is lossless, only the background image can improve when a larger bit budget is available. Adding the residuals to the background guarantees that at high enough coding rate their values can be recovered.



(a) (b)

Figure 5.4: Comparison of the background in the text portion of the *camcorder* image (a) with binary text coding, (b) with ternary text coding.

5.3.4 Ternary Coding of Text Blocks

The procedure described thus far performs well, but further improvement is possible. In many compound images, especially the ones obtained as a result of scanning documents, the transition from foreground to background color is not sharp; there are some intermediate levels present as well. The discontinuities in the background image introduced by adding the residual can be reduced by more finely quantizing the foreground/background map. Specifically, instead of computing a binary map, we compute a ternary map – i.e., by adding a third “transition” level between foreground and background. This results in smaller residual peaks at the edges of text. The encoder determines the foreground/transition/background map for each text block in the image using a 3-level Lloyd-Max quantizer. This map is transmitted using arithmetic coding with a ternary alphabet. The encoder transmits a block’s predictive quantized foreground and

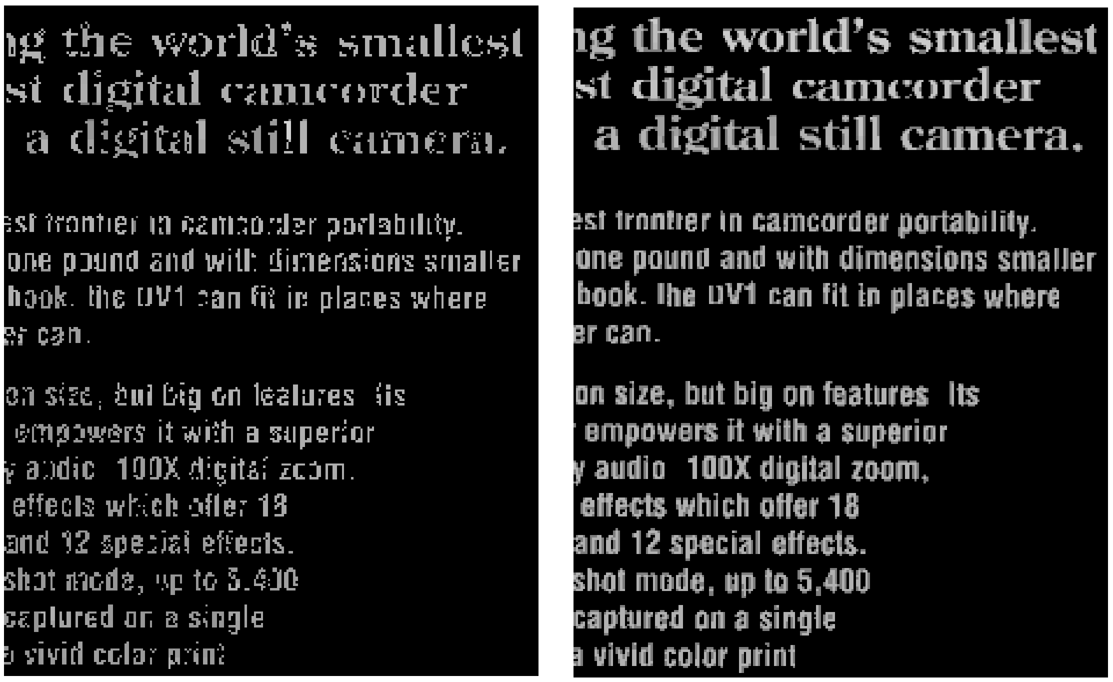


Figure 5.5: Comparison of the foreground in the text portion of the *camcorder* image (a) with binary text coding, (b) with ternary text coding.

intermediate color as before. As with unimodal blocks in the binary case, the algorithm only uses a ternary description of a text block if three distinctive levels can be identified in a block. Allowing binary and unimodal blocks also improve the coding rate for the arithmetic coder.

The residual peaks are smaller than in the binary case, however. This can be seen for the text portion of the *camcorder* image in Figure 5.4. The foreground quality is also improved as shown in Figure 5.5. Having a smoother image allows the wavelet transform to concentrate most of the energy in the lower frequency bands and the image can be coded with less distortion at the same rate.

5.3.5 Results

In Figure 5.6 we compare results of our combined quadtree wavelet and spatial domain coding approach for color images with the color versions of SPIHT and EBCOT. The original color image was sampled at 100 dpi from a print source. The results are presented at a $100:1$ compression ratio. We used the ternary version of our combined wavelet/spatial-domain approach. In the image compressed by our algorithm, text is significantly sharper and clearer than in the SPIHT and EBCOT images. Separate coding of text and image data not only better preserved the text, but the spatial domain coding proved to be more efficient as well, leaving more bandwidth to code the image data. Note that the background edges around the letters “JVC” appear sharper than the same areas in the SPIHT and EBCOT images. In this case the improved visual image quality is reflected by an improvement in PSNR as well, with a gain of 0.95 dB over the SPIHT encoded image, and 4.93 dB over the EBCOT encoded image.

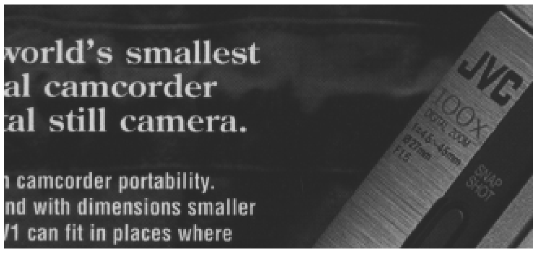
5.4 Summary

In this work we proposed two different algorithms for the compression of document images. One method works entirely in the wavelet domain and aims at encoding the text portions of the image more accurately at the expense of the non-text areas. We introduced an effective method for changing filters within an image at any arbitrary point.

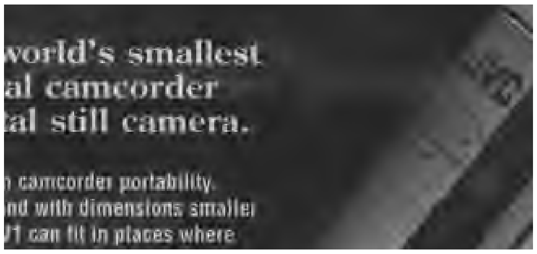
The other technique processes text pixels in the spatial domain while background pixels are wavelet transformed and coded in the frequency domain. To improve wavelet coding efficiency, the background image is smoothed to fill in the gaps where the text pixels have been removed before transform. For the foreground map we used both a binary and a ternary encoding. The ternary coding is especially effective when the document image is a result of scanning where transition between text and non-text blocks is more gradual. To enable lossless reconstruction at high rates the foreground representation error is superimposed on the background image.

Both these techniques outperformed state-of-the-art wavelet-based image coding methods in terms of image quality both in the text and the non-text portions of the images.

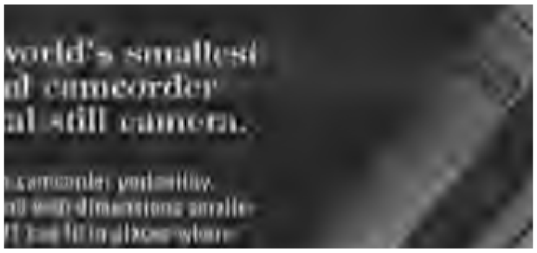
This chapter, in part, is a reprint of the material as it appears in P. Cosman, T. Frajka, D. Schilling, and K. Zeger. Memory efficient quadtree wavelet coding for compound images. *33rd Asilomar Conference on Signals, Systems and Computers*, pp 1173-1177, Vol. 2, Pacific Grove, CA, Oct. 1999.



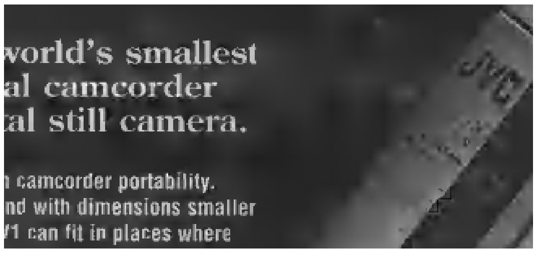
(a)



(b)



(c)



(d)

Figure 5.6: Portion of the color image *camcorder*, reproduced here in black and white (a) Original image, 100dpi, (b) SPIHT at 0.24 bpp, 27.07 dB, (c) EBCOT at 0.24 bpp, 23.09 dB, (d) Combined wavelet/spatial coding approach, 0.24 bpp, 28.02 dB.

Chapter 6

Image Coding Subject to Channel Constraints

In this chapter we propose a new packetization scheme based on the MultiGrid Embedding (MGE) coding. In zerotree based packetization methods, packets contain one or more zerotrees. Any loss of data wipes out the entire corresponding spatial location in the image. Using our proposed method, the effect of the packet loss is not concentrated on a given spatial location but spread over different locations and frequencies.

With the increased interest in multimedia communications transmission of images has become a fact of life. Image compression used to focus on optimizing the image data for storage on media that, for the most part, act as noiseless channels. Modern communication systems experience a wide variety of channel conditions from individual bit errors to packet losses. In block based coders such as JPEG, the effect of the error can be limited to the given block. State of the art coders use wavelet transform and perform global coding of coefficients. These coders can experience catastrophic failures in case of bit errors or packet losses.

In order to maximize compression efficiency, wavelet based coders such as EZW or SPIHT rely on state information and encoding decisions conveyed by a single bit. In such cases, any error can lead to a breakdown at the decoder. In a packet network, a lost packet causes complete loss of synchronization at the receiving end, rendering all future packets useless for decoding.

One could use unequal amounts of Forward Error Control (FEC) coding to protect the image data and combine that with packetization as done in [54]. The amount of protection depends on the noisiness of the channel. In the case of the Internet, the noise characteristics may change very rapidly even during the transfer of an image. A pre-determined protection scheme may not work as the channel changes, or if it is tailored to the maximum expected error it may reduce the source coding rate too severely. An alternate approach is to make the image coder more robust to prevent catastrophic failures. Rogers and Cosman [67] propose a packetization scheme (Packetized Zerotree Wavelet (PZW)) that groups individual zerotrees into packets by changing the SPIHT encoding order. Instead of sending the information about all zerotrees in the order of magnitude, they collect descriptions corresponding to each zerotree separately. Thus any packet loss will only have an effect on the spatial locations of the image that the given zerotrees correspond to. Sherwood and Zeger [74] use a macroscopic multistage compression method to enhance the robustness of the packetized coder. The image is encoded in stages, where the input to the next stage is the residual of the original image with the output of the current stage. Each stage is coded independent of the others making it possible to decode future stages even when information in past stages is corrupted. Each stage can be assigned a different level of error protection. In their work, only two stages are used, with the first stage image data being protected by FEC and the second stage sent without additional error protection.

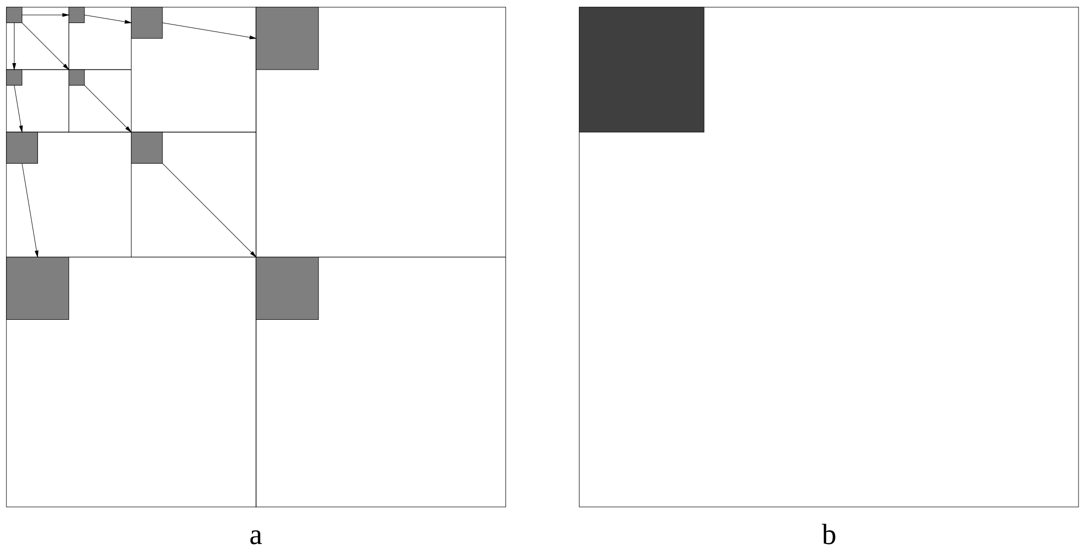


Figure 6.1: (a) The zerotree structure in the wavelet domain and (b) the corresponding spatial location in the image.

6.1 Spread Spatial Location Packetization

In a zerotree based coder, coefficients in the same zerotree correspond to the same spatial location in the image as is shown in Figure 6.1. When packets are formed of several zerotrees, the loss of such a packet results in the loss of information at the given spatial locations, a blank spot on the image. This property is the consequence of the zerotree structure, the coding of higher frequency coefficients at the same spatial location depends on the coding of lower frequency coefficients at the same location. The description of each zerotree has to be kept in the same packet to make the packets independently decodable. Otherwise the loss of the packet containing the description of the lower frequency coefficients renders the received description of higher frequency coefficients on the same zerotree useless.

MGE is a flexible alternative to the zerotree structure. Without the zerotree dependence, the coding can proceed in almost any arbitrary order. In the original MGE algorithm, the quadtree identification process for the significant coefficients starts with the full wavelet domain image. The coding dependencies only exist within the quadtree structure. Thus, by choosing the appropriate initial block size for the quadtree identification process, different spatial locations can be coded separately. With an l level wavelet decomposition, $2^k \times 2^k$ sized non-overlapping regions of the image can be independently coded by using $2^{k-1} \times 2^{k-1}$ sized blocks at the first level, $2^{k-2} \times 2^{k-2}$ sized blocks at the second level and $2^{k-l} \times 2^{k-l}$ sized blocks at the l^{th} level as the initial blocks for the significance check. On one hand, this change will decrease compression efficiency as many blocks that would normally be coded after a single significance check at a given bit plane will have to be individually compared. On the other hand, this change makes it possible to code different spatial locations and frequency contents independently.

We define a collection of wavelet coefficient blocks *random zerotree* that satisfies the following properties:

- Each random zerotree contains one block from each frequency band of the appropriate size ($2^{k-l} \times 2^{k-l}$ on level l, for $l = 1 \ldots L$) that corresponds to the same spatial domain size.

- Each block represents a different spatial location of the image to spread any possible information loss over different parts of the image.

- The intersection of any two random zerotrees is the empty set.

- The union of all random zerotrees covers the full wavelet domain image.

Figure 6.2 shows a random zerotree and the effect of a loss of such a tree on the spatial domain data. Each affected spatial location is missing a different frequency component making the visual loss less noticeable. (The different shades of grey signify the differing contributions of different frequency bands to the overall image quality with darker being more significant.)

In addition to the trade-off between flexibility and compression performance,

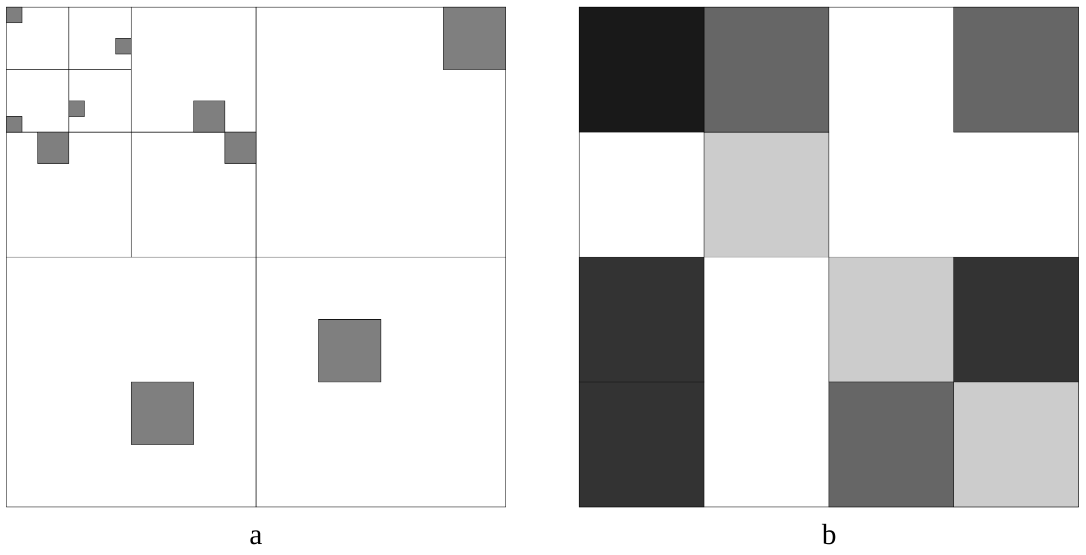


Figure 6.2: (a)A *random zerotree* in the wavelet domain and (b) the corresponding spatial locations in the image.

one also needs to consider the effect of block size on the description length of random zerotrees. The description length clearly depends on the target coding rate. Small initial block sizes yield short descriptions that allow flexible packet formation, but degrade compression performance. Large initial block sizes however may generate so much information for each random zerotree that it becomes impossible to fit all the information into the payload of a single packet. This violates the constraint that each packet be independently decodable.

For the same reason, the adaptive arithmetic coder must gather its distribution estimates independently for each packet. By not being able to use information learned at other locations in the coding process its compression efficiency decreases.

The coding proceeds in two phases. The first phase is a “dry run” when the encoder collects the rate information for each random zerotree for the given target bit rate. The second phase is the actual packet formation. The encoder tries to pack any packet as fully as possible, even allowing the sum of the rates to exceed the packet size by some margin if without adding the last random zerotree the packet is under

Table 6.1: Comparison of the proposed method and the PZW algorithm ([67]) for the *Lena* and *Peppers* images at a bit rate of 0.209 bpp.

		PSNR (dB)			
Image	Algorithm	no loss	1 % loss	10 % loss	20 % loss
Lena	PZW	32.19	31.33	26.29	24.63
	This work	31.90	30.91	26.24	23.70
Peppers	PZW	31.75	30.85	26.38	23.31
	This work	32.09	30.99	26.09	23.49

filled. Thus some random zerotrees will have more rate assigned to them than in the non-packetized coding, while others will have a lower rate description. The assignment of wavelet coefficients to random zerotrees is determined at random, but ensuring that the properties of the random zerotrees are satisfied. The packing is carried out in this order which is known both to the encoder and decoder, and is used for all images. The overhead information is limited to the sequence number of the starting random zerotree and the number of random zerotrees in the packet. Since the LL-band information has the most significant effect on the image quality, the packing algorithm avoids placing random zerotrees with neighboring LL-band blocks in the same packet.

6.2 Experimental results

To demonstrate the performance of the proposed scheme we used the 512×512 greyscale *Lena*, *Baboon*, and *Peppers* images. The images were transformed with a 4-level wavelet transform using the 9-7 filters. The random zerotree assignments were generated in advance and used for all test images. There are a total of 256 random zerotrees, corresponding to a block size of 2×2 in the LL band. A packet size of 384 bits was chosen. In the channel model, lost packets do not arrive at the decoder; it must be able to perform independent decoding on each packet.

The PSNR results for the *Lena* and *Peppers* images are shown in Table 6.1. The PSNR values were obtained using the average MSE of 10,000 independent runs for each loss scenario. The performance of our method is comparable to that of the PZW algorithm over the different loss scenarios. It is a well known fact that PSNR values are sometimes poor indicators of the underlying image quality. While the above two methods yield similar PSNR results the corresponding image quality is quite different. The amount of information lost in both methods is about the same (as measured by the PSNR), but its distribution is different in the image.

Figures 6.3 and 6.4 show the visual comparison between a zerotree style packetization approach (like PZW) and the proposed method for packet loss rates of 1%, 10% and 20% for the *Lena* and *Baboon* images. The packets loss scenario was forcing the loss of typically the same LL-band blocks in both packetization schemes. As can be seen the images generated with the proposed method does not exhibit regions that are completely wiped out due to the packet loss, rather disperse the loss through different frequencies of different spatial blocks resulting in a visually less disturbing image.

The robust packetization method sacrifices the progressivity of the underlying coder. While within the packet the coding is progressive, any further progressive refreshing is tied to the reception of the next packet. Furthermore, certain portions of the image improve in an uneven fashion as opposed to the gradual uniform improvement associated with progressive image coders.

6.3 Summary

We presented a packetization scheme that spreads the information within the packet among frequency bands and spatial locations to avoid complete loss of image informa-

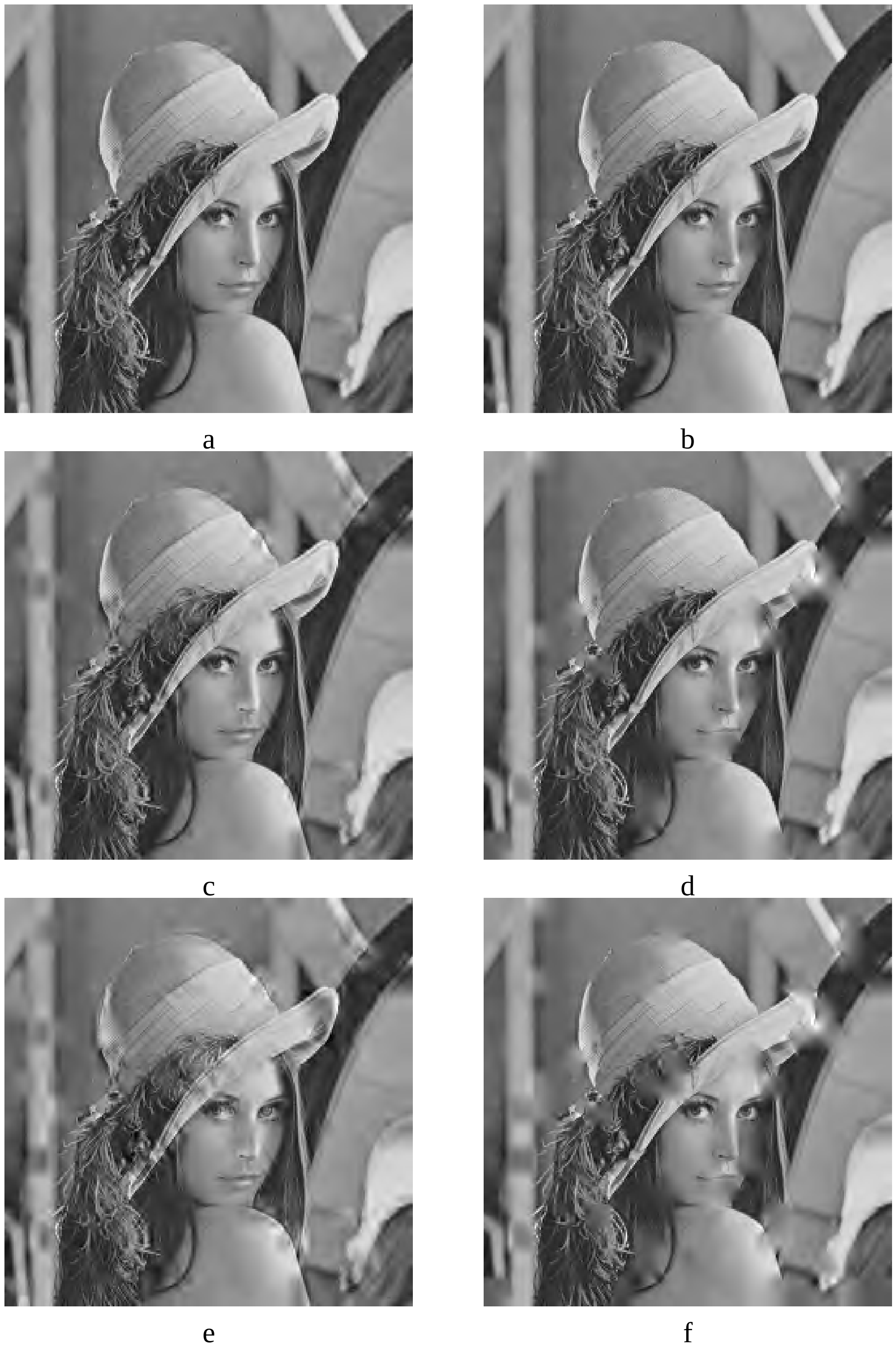


Figure 6.3: Comparison of proposed method (left column) and zerotree style packetization (right column) for the *Lena* image at 0.208 bpp, (a)-(b) 1% loss, (c)-(d) 10% loss, (e)-(f) 20% loss.

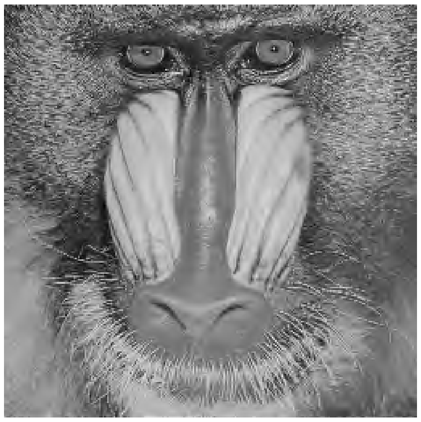
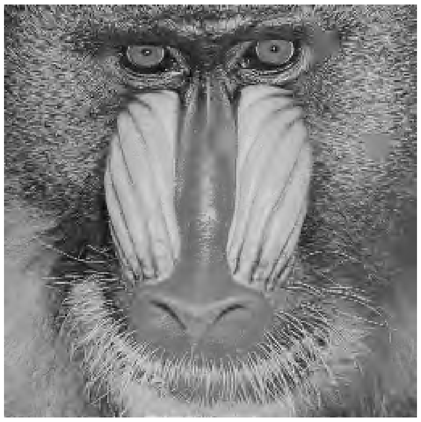
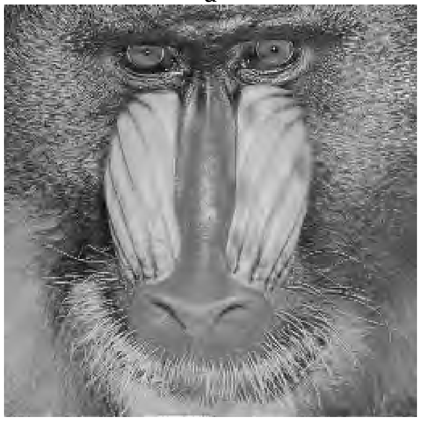
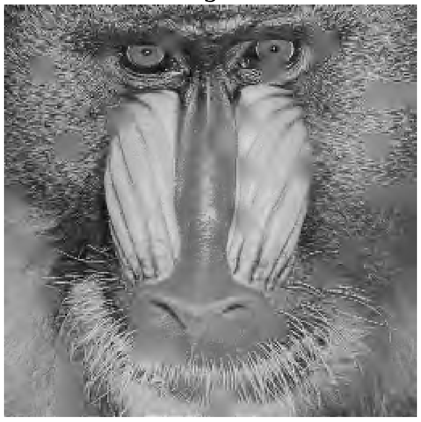
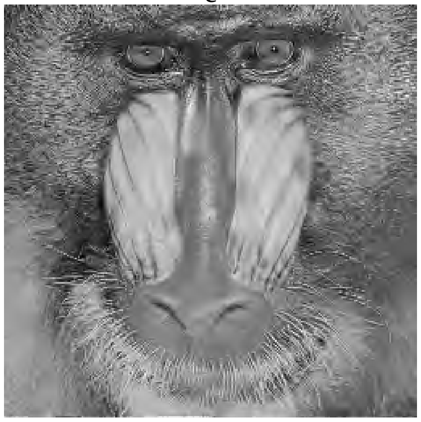
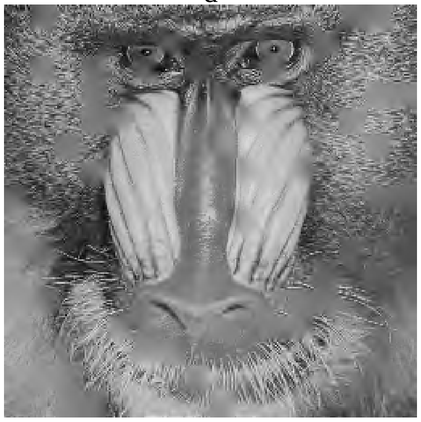

a

b

c

d

e

f

Figure 6.4: Comparison of proposed method (left column) and zerotree style packetization (right column) for the *Baboon* image at 0.4 bpp, (a)-(b) 1% loss, (c)-(d) 10% loss, (e)-(f) 20% loss.

tion at any given spatial block. Our experimental results indicate that this proposed method produces better quality images over noisy channels than previously published techniques.

The coder is flexible in the coding of the information due to the MGE structure that is not limited by the zerotree interband dependencies. The coder loses progressivity at the expense of more robust performance on packet erasure channels.

This chapter, in part, is a reprint of the material as it appears in T. Frajka and K. Zeger. Robust Packet Image Transmission by Wavelet Coefficient Dispersement. *International Conference on Acoustics, Speech and Signal Processing*, pp 1745-1748, Vol. 3, Salt Lake City, UT, May 2001. The dissertation author was the primary investigator of this paper.

Chapter 7

Residual Image Coding for Stereo Image Compression

One main focus of research in stereo image coding has been disparity estimation, a technique used to reduce the coding rate by taking advantage of the redundancy in a stereo image pair. Significantly less effort has been put into the coding of the residual image. These images display characteristics that are different from that of natural images. In this chapter we propose a new method for the coding of residual images that takes into account the properties of residual images. Particular attention is paid to the effects of occlusion and the correlation properties of residual images that result from block-based disparity estimation. The embedded, progressive nature of our coder allows one to stop decoding at any time. We demonstrate that it is possible to achieve good results with a computationally simple method.

7.1 Introduction

Human depth perception in part relies on the difference in the images which the left and right eyes send to the brain. By presenting the appropriate image of a stereo pair to the left and right eyes, the viewer perceives scenes in three dimensions instead of as a 2-dimensional image. Such binocular visual information is useful in many fields, such as telepresence style video conferencing, telemedicine, remote sensing, and computer vision.

These applications require the storage or transmission of the stereo pair. Since the images seen with the left and right eye differ only in small areas, techniques that try to exploit the dependency can yield better performance than independent coding of the image pair.

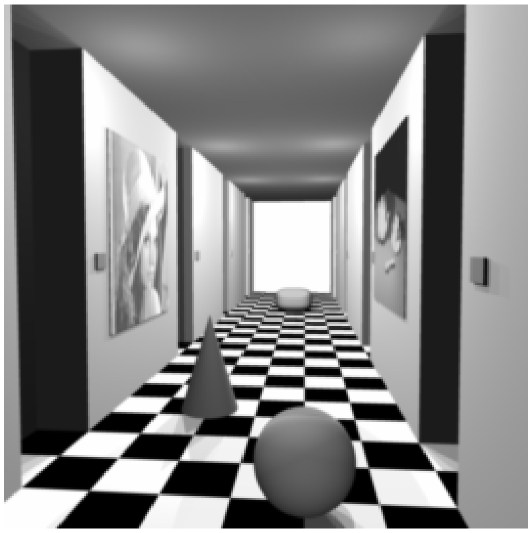
Most successful techniques rely on disparity compensation to achieve good performance. Disparity compensation is similar to motion compensation for video compression. It can be carried out in the spatial domain [35, 39, 40, 45, 84] , or in the transform domain [36]. Disparity compensation can be a computationally complex process. In [66] a wavelet transform based method is used for stereo image coding that does not rely on disparity compensation. While this technique is more simple, it sacrifices compression performance for reduced complexity.

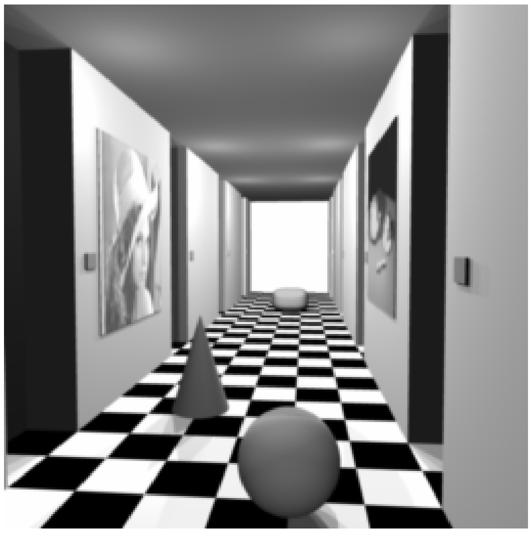
Many of the above works use discrete cosine transform (DCT) based coding of the images which uses a rate allocation method to divide the available bandwidth between the two images. For each target bit rate a new optimization has to be performed to find the optimal balance. Embedded coding techniques based on the wavelet transform [72, 68] provide improved performance for still images when compared with DCT-based methods. An embedded bit stream can be truncated at any point to obtain the best reconstruction for the given bit rate. An embedded stereo image coding scheme is proposed in [12] that achieves good performance without having to use rate allocation.

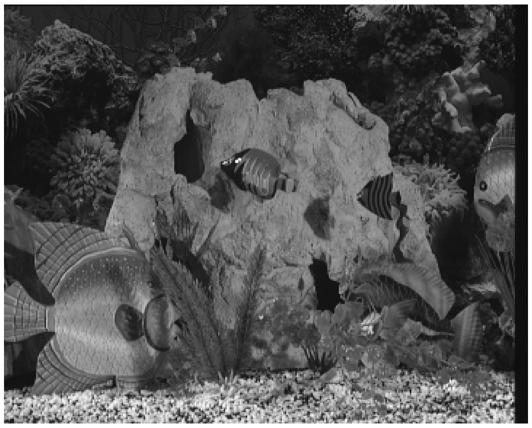
With disparity compensation, one image is used as a reference image, and the other is predicted using the reference image. The gain over independent coding comes from compressing the residual image that is obtained as the difference of the original and predicted image. Little attention has been paid to the coding of the residual image. Moellenhoff and Maier [53] examined the properties of disparity compensated residual images and proposed some DCT and wavelet techniques for their improved encoding.

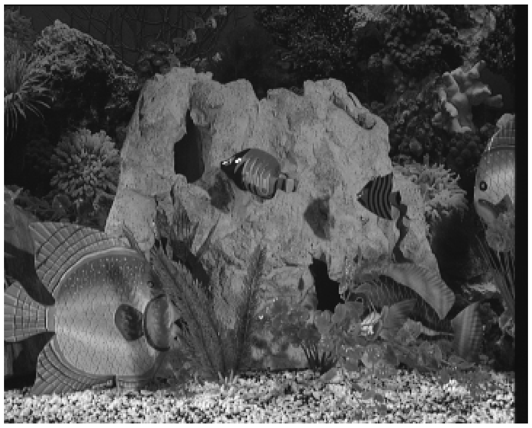
In this chapter, we propose a progressive coding technique for the compression of stereo images. The emphasis of our work is on the coding of the residual image. These images exhibit properties different from natural images. Our coding techniques make use of these differences. We show that the correlation across block boundaries in the residual image is diminished, suggesting that the coding of these blocks individually might be preferable. We propose to use transforms that take into account the correlation properties of the residual image as well as the block-based nature of the disparity estimator used in our coder. Occluded blocks are often difficult to estimate. Residuals of occluded blocks therefore are different from blocks that are well matched by the disparity estimation (DE) process. In our technique, we treat these two types of blocks differently. The image transform we propose uses a DCT on the blocks that are well matched by DE, and uses Haar-filtering on the occluded blocks, resulting in a mixed image transform. MultiGrid Embedding [43] is used as the embedded image coder. It provides similar performance to that of zerotree-based techniques with increased flexibility. While the components of our proposed method are low complexity, they yield some significant improvements over other methods in the coding of stereo images.

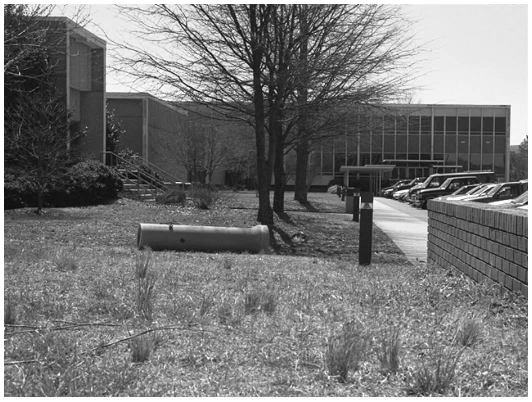
The outline of the chapter is as follows. Section 7.2 gives an overview of stereo image coding. Our contribution is in Section 7.3, with experimental results provided in Section 7.4. Finally, the conclusion is given in Section 7.5.


a


b


c


d


e

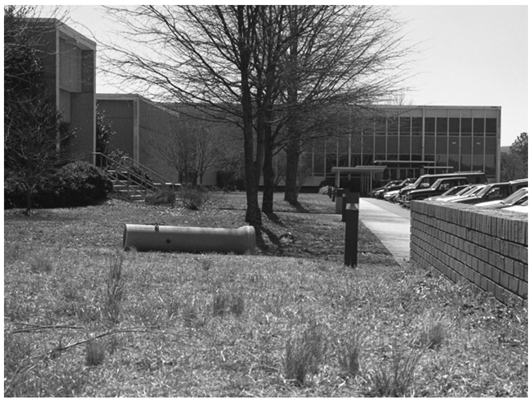

f

Figure 7.1: Original images of the (a) and (b) *Room*, (c) and (d) *Aqua*, and (e) and (f) *Outdoors* stereo pairs.

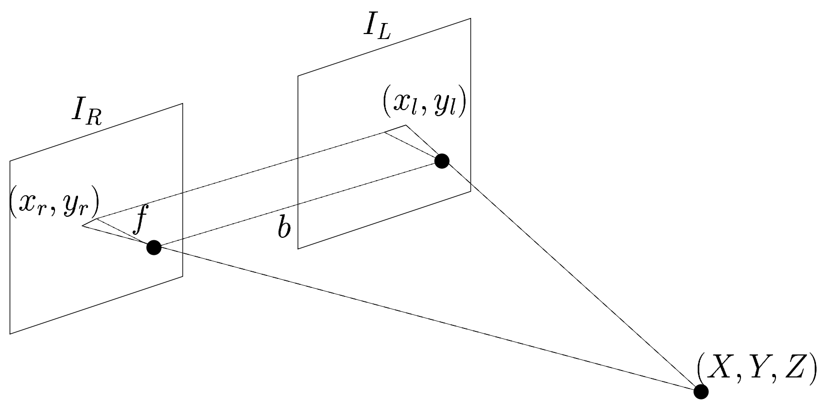


Figure 7.2: Stereo camera system.

7.2 Stereo Image Coding

Stereoscopic image pairs represent a view of the same scene from two slightly different positions. When the images are presented to the respective eye the human observer perceives the depth in the scene as in 3 dimensions. One can obtain stereo pairs by taking pictures with two cameras that are placed in parallel 2 to 3 inches apart. Figure 7.1 shows the original images of three different stereo pairs. The synthetic *Room* image represents such applications as video games or virtual reality, while the two natural scenes provide different distance scenarios which exhibit different disparity properties. The left and right images of the *Room* pair differ mainly in the left edge of the left image where a piece of the wall is visible that cannot be found in the right image. Certain areas of the floor tile are differently covered by the cone and ball in these images. In the *Aqua* pair the differences occur at the left and right edges of the images as well as around the rock in the middle. Because of the larger distance between the objects and the camera of the *Outdoors* pair, they exhibit the smallest differences, mostly around the image edges.

Because of the different perspective, the same point in the object will be mapped to different coordinates in the left and right images as shown in Figure 7.2. Let (x_l, y_l) and (x_r, y_r) denote the coordinates of an object point in the left and right images, respectively. The *disparity* is the difference between these coordinates, $\mathbf{d} = (x_l - x_r, y_l - y_r)$. If the cameras are placed in parallel, then $y_l - y_r = 0$, and the disparity is limited to the horizontal direction. Let b denote the separation between the two cameras, f the focal length, and Z the depth of the object (or the distance of the object from the camera) as shown in Figure 7.2. If all of these parameters are known, one can compute the disparity of each object as [2, 93]

$$|d| = \frac{bf}{Z}. \tag{7.1}$$

This equation agrees with the intuition that objects closer to the camera will exhibit larger disparity than objects farther away.

In order to exploit the dependency between the left and right images most compression schemes use a conditional coder structure as depicted in Figure 7.3. One image of the pair serves as a reference image, I_{ref}, and the other, I_{pred}, is predicted from the reference image. The encoder describes to the decoder the reference image, the residual image (i.e. the difference of the predicted image and its estimate), I_{diff}, and the disparity vectors used to obtain the estimate image. At the decoder the reconstructed reference image, $\hat{I}_{ref}$, and the decoded disparity vector field are used by the disparity compensation process to arrive at the estimate $\tilde{I}_{pred}$ of the predicted image. The reconstructed predicted image, $\hat{I}_{pred}$, is obtained by adding the reconstructed difference image, $\hat{I}_{diff}$, to $\tilde{I}_{pred}$. Most earlier methods used a DCT-based encoding for both the reference image and the residual image. Recently, progressive, wavelet based techniques have been introduced [12, 57] to replace the DCT-based encoding, and Perkins [60] showed that in general the conditional coder structure is sub-optimal in the rate-distortion sense.

If the parameters in (7.1) are known ahead of time then, in principle, for each pixel one could compute the disparity and use that in the prediction. Unfortunately these parameters are seldom available at the time of the encoding. Using disparity estimation, one could try to obtain approximate values for each pixel of the image. Since this process would be quite complex if performed for each pixel individually, it is usually carried out for groups of pixels instead. One such grouping is the use of $k \times l$ non-overlapping blocks.

The search for the matching block is carried out in a limited search window. Given the reference image, the optimal match could be any $k \times l$ block of the image. This exhaustive search is computationally complex. From the parallel camera axis assumption, one can restrict the search to horizontal displacements only. (A small vertical displacement can also be allowed to compensate for the inaccuracy of practical camera systems.) From the camera setup it is clear that the disparity for objects in the left image with respect to the right image is positive and vice versa. This observation helps further limit the scope of the search.

Other techniques aimed at reducing the computational cost of full search DE include dynamic programming [35], hierarchical DE [45], and adaptive directional, limited search algorithms [39].

Let D denote a distortion measure between image blocks and let $I[i, i', j, j']$ denote a $(i' - i) \times (j' - j)$ block with upper left coordinates (i, j) and lower right coordinates (i', j'). Then the *disparity estimate* for a block with upper left coordinates (i, j), assuming only horizontal displacement, is defined as

$$\tilde{d}_{ij}(w) = \operatorname*{argmin}_{d \leq w} D(I_{pred}[i, i+k, j, j+l], I_{ref}[i+d, i+k+d, j, j+l]). \qquad (7.2)$$

where w is the disparity window size within which the search is performed.

The quantity $\tilde{d}_{ij}(w)$ is the horizontal amount by which a $k \times l$ block in one image has to be shifted in order to most closely match in similarity a given $k \times l$ block

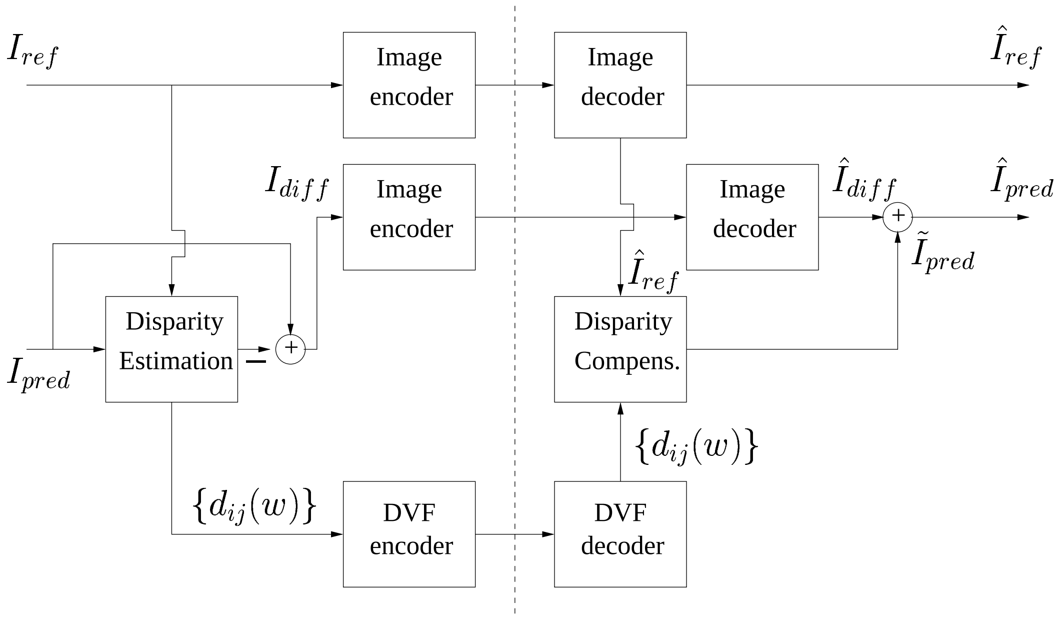


Figure 7.3: Stereo image coding system based on a conditional coder structure. The left side of the dashed line is the encoder and the right side is the decoder.

in another image. The two most often used similarity measures in block matching are the maximum absolute difference and the mean-squared error. Using the mean-squared error, (7.2) becomes

$$\tilde{d}_{ij}(w) = \operatorname*{argmin}_{d \leq w} \sum_{m=i}^{i+k} \sum_{n=j}^{j+l} (I_{pred}(m,n) - I_{ref}(m+d,n))^2 \tag{7.3}$$

where $I(m,n)$ is the pixel intensity value at coordinate (m,n).

The disparity estimation process works well for blocks that are present in both images. However, occlusion may result if certain image information is present only in one of the images. Occlusion can happen for two main reasons: finite viewing area and depth discontinuity. Finite viewing area occurs on the left side of the left image and the right side of the right image where each eye can see objects that the other eye cannot. Depth discontinuity is due to overlapping objects in the image; certain portions can be hidden from one eye on which the other eye has direct sight.

Another cause of mismatch is photometric variations. This phenomenon is due to the variation of the reflected light that reaches the left and right lenses. A simple, global solution to this problem is histogram modification, as proposed in [24].

At the price of increased complexity, several methods were introduced that try to improve DE for both occlusion and photometric variations: subspace projection [8], sequential orthogonal subspace updating [70], and overlapped block DE [91].

For any positive number w, define the *disparity vector field* (DVF) of an $X \times Y$ image to be the matrix of integers

$$\{\tilde{d}_{ij}(w)\}$$

where $0 \leq i \leq X-1$, $0 \leq j \leq Y-1$, and i and j are divisible by k and l, respectively.

The reconstructed predicted image depends on the quality of the reconstructed reference image, $\hat{I}_{ref}$, the disparity vector field, and the reconstructed difference image, $\hat{I}_{diff}$. The image predicted from the reconstructed reference image using the DVF can be written as

$$\hat{I}_{pred} = \tilde{I}_{pred} + \hat{I}_{diff}$$

where $\tilde{I}_{pred}$ depends on $\hat{I}_{ref}$ and $\{\tilde{d}_{ij}(w)\}$.

Then given the compressed reference image, $\hat{I}_{ref}$, of size $X \times Y$, the predicted image estimate is

$$\begin{aligned} \tilde{I}_{pred} \quad &= \quad \Big\{ \hat{I}_{ref}[i + \tilde{d}_{ij}(w), i + \tilde{d}_{ij}(w) + k, j, j + l] : \\ & \qquad 0 \leq i \leq X-1, \quad 0 \leq j \leq Y-1, k|i, l|j \Big\} \end{aligned}$$

and the *disparity estimation distortion* is defined as

$$\tilde{D}_{pred}(\{\tilde{d}_{ij}(w)\}, \hat{I}_{ref}) = ||I_{pred} - \tilde{I}_{pred}||^2 = \sum_{m,n} \Big(I_{pred}(m,n) - \tilde{I}_{pred}(m,n) \Big)^2 .$$

The *total rate* is the sum of the rates of coding the reference image, the disparity vector field, and the residual image, namely

$$R_T = R_{ref} + R_{dvf} + R_{diff}.$$

The *distortion* is defined as the average of the distortions of the two images:

$$\begin{aligned} D_T &= (D_{ref} + D_{pred})/2 \\ &= (||I_{ref} - \hat{I}_{ref}||^2 + ||I_{pred} - \hat{I}_{pred}||^2)/2. \end{aligned}$$

The overall rate-distortion performance depends on the rate allocation between R_{ref}, R_{dvf}, and R_{diff}.

In practice, without the DVF no real stereo effect can be achieved. It is often assumed that the minimum coding rate is at least R_{dvf} and that the DVF is encoded first, leaving the rate allocation to the reference and difference images for the remaining available rate. The most widely used technique for the encoding of the DVF is DPCM followed by entropy coding [97, 40, 35, 57, 7, 53]. Entropy coding alone is employed in [12, 36], and fixed length coding in [70].Tzovaras and Strintzis [85] proposed a rate-distortion framework for the encoding of the disparity vector field, allowing some distortion in the transmission of the displacement vectors.

Once the DVF is transmitted, the residual and the reference images are encoded. Many proposed techniques use DCT-based block coding methods for the encoding of both images. They also require a bit allocation mechanism to determine the coding rate of each image. (This bit allocation is carried out in addition to the bit allocation between the DCT-transformed blocks of each image.) For each target bit rate, a separate optimization is used to determine the appropriate bit allocation. Woo and Ortega [90] perform a blockwise dependent optimization instead of independent optimization for the

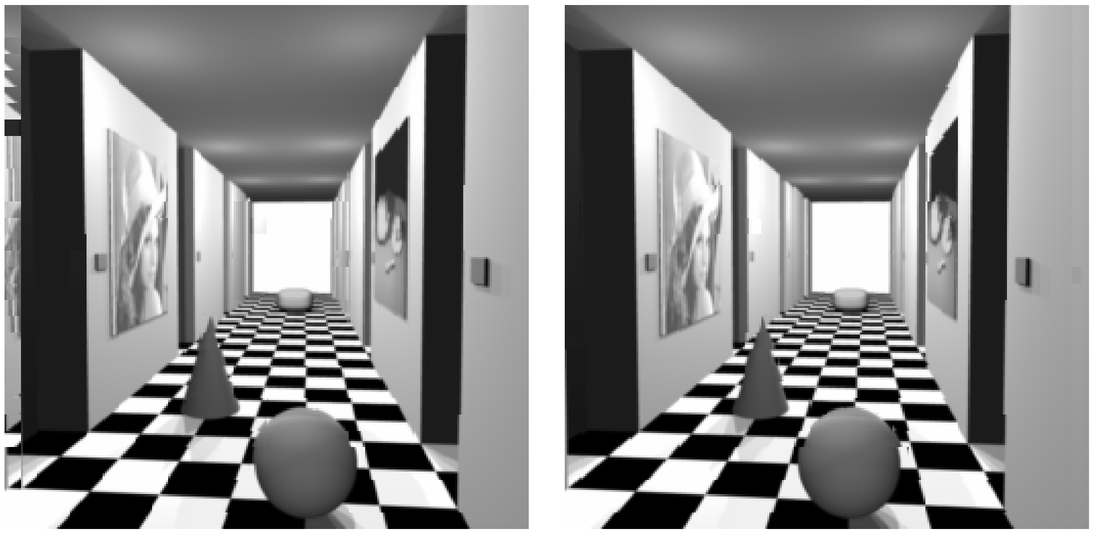


Figure 7.4: (a) Estimate of left image with right image as reference and (b) estimate of the right image with left image as reference.

reference and residual images to improve the coding performance.

Embedded image coders can be terminated at any bit rate and still yield their best reconstruction at that rate without a priori optimization. Zerotree-style techniques such as the Embedded Zerotree Wavelet (EZW, [72]) by Shapiro, or Set Partitioning in Hierarchical Trees (SPIHT, [68]) by Said and Pearlman offer excellent compression performance for still images. These zerotree techniques are extended to stereo images [12] by Boulgouris and Strintzis. The bit plane coding is performed on both the residual and reference image at the same time, guaranteeing that the most significant information for both images is sent before the less significant information.

7.3 Residual Image Coding

The goal of this chapter is to make stereo image coding more efficient by improving the coding of the residual image. The DE we chose is rather simple, but even with such a simple disparity estimator, our proposed coding technique has very good performance.

7.3.1 Image Coding Method

Embedded coding yields good performance coupled with simplicity of coding due to not having to perform any bit allocation procedure. MultiGrid Embedding (MGE) by Lan and Tewfik [43] uses a quadtree structure instead of the zerotrees of EZW or SPIHT. It uses the same bit plane coding, starting from the most significant bits of the transform domain image down to the least significant. For each bit plane, the quadtree structure is used to identify the significant coefficients, i.e., those whose most significant bit is found on that bit plane. The "sorting" pass identifies the coefficients that become significant on the current bit plane, while the "refinement" pass refines those coefficients that have previously become significant. Their results demonstrated that this technique outperforms the zerotree-based methods on images with significant high frequency content. As residual images contain edges and other high frequency information, MGE is a natural candidate for their encoding.

The way we use MGE for stereo image compression is similar to that in [12]. For each bit plane, first the sorting and refinement pass are executed for the reference image and then for the residual image. The highest magnitude coefficient is usually smaller for the residual image than for the reference image.

7.3.2 Occlusion

As noted in Section 7.2, there are two kinds of occlusion that may occur in DE. A finite viewing area can be overcome in certain cases. If a one directional search is used (as suggested by the observation on the direction of the displacement in Section 7.2) that method could run out of image pixels at the edge of the image where it would also have difficulty finding the corresponding block in the reference image. If, however, we allow the search to continue in the other direction, it may find blocks similar to the one to be estimated. This can be seen in Figure 7.4, where the estimate of the left image on the left edge clearly displays some occlusion error, while the right edge of the right image looks almost identical to the original.

The residual image of those blocks that are occluded because of depth discontinuity display different characteristics from the other parts of the image. As noted in [52], the occluded blocks are more correlated. We propose to detect such blocks, and code them differently from the rest of the residual image blocks for improved efficiency.

7.3.3 Image Transform

Moellenhoff and Maier's analysis [52] indicates that residual images show significantly different characteristics from natural images. Residual images mainly contain edges and other high frequency information. The correlation between neighboring pixels is smaller as well. This suggests that transforms that work well for natural images may not be as effective for residual images.

In wavelet transform coding, one of the most widely used filters is the 9-7 filter by Antonini et al. [3]. It is preferred for its regularity and smoothing properties. With the image pixels less correlated in residual images, shorter filters can better capture the local changes. For this reason we propose the use of Haar-filters. These 2-tap filters take the average (lowpass) and difference (highpass) of two neighboring pixels. As our experimental results show, the use of Haar-filters improves performance.

DE uses $k \times k$ size blocks to find the best estimates for the image. There is no reason to expect neighboring blocks to exhibit similar residual properties. For one block, the algorithm can find a relatively good match, while its neighbor could be harder to predict from the reference image.

Moellenhoff's results indicate that the pixels of the residual image are less correlated than those of the original image. But they do not reveal much about the local

Table 7.1: Comparison of 1-pixel horizontal correlation for pixels in a given column of an 8×8 block of the right image of the *Room* and *Aqua* stereo image pairs.

	Column number within 8×8 block							
	1	2	3	4	5	6	7	8
Room original	0.93	0.94	0.96	0.96	0.97	0.95	0.94	0.94
Room residual	0.27	0.38	0.41	0.45	0.44	0.31	0.33	0.03
Aqua original	0.90	0.89	0.89	0.89	0.88	0.88	0.87	0.89
Aqua residual	0.24	0.25	0.22	0.23	0.25	0.23	0.26	0.12

correlation of pixels, namely across the $k \times k$ block boundaries. We investigate 1-pixel correlation on a more local scale in both horizontal and vertical directions. Instead of gathering these statistics for the whole image, we only look at the correlation between all pixels in the n^{th} column/row and its immediate neighbor in the $(n+1)^{th}$ column or row of all $k \times k$ blocks for the case of horizontal or vertical correlation, respectively. Note that the correlation between the k^{th} and $(k+1)^{th}$ columns/rows gives the correlation just across the boundary between two neighboring blocks. Table 7.1 shows the 1-pixel correlation in the horizontal direction for the right image and its residual using blocks of size 8×8. (The trends are similar for vertical correlation and for the left image as well.) It can be seen in Table 7.1 that the 1-pixel correlation drops significantly at the block boundary (column 8) in the residual image, supporting our assumption that different blocks exhibit different properties in the residual image.

Based on this observation, we focus on block-based transforms that can better capture the differences between the blocks than a global transform, such as the wavelet transform, that sweeps across the block boundaries. The DCT in practice is performed on $k \times k$ blocks. Its performance is diminished by the JPEG encoding method. However, if the DCT coefficients are regrouped into a wavelet decomposition style subband structure as proposed in [92], and are encoded using an embedded coder, the performance approaches that of wavelet based methods. (This method is referred to as Embedded Zerotree DCT (EZDCT).)

None of the proposed image transforms so far take into account the effect of occlusion. For an occluded block, the best match can still be a very distorted one. In those cases, not using the estimate for the given block at all could be the best strategy. This is similar to coding an I (intra coded) block in the H.263 video compression algorithm. For each block, the estimator should decide if the best match is good enough. If not, the given block is left intact. This process creates a mixed residual image, with some parts having mostly edges and high frequency information, and other parts blocks from the original image. For residual blocks that contain significant high frequency information, a uniform band partitioning (such as with DCT) works better than octave-band signal decomposition (see [83]), while octave-band decomposition is desirable in blocks of the original image.

Note that the Haar transform only uses two neighboring pixels to compute the low and high frequency coefficients, then moves on to the next pair. If the block size k is even, then starting at the left edge of the block, the Haar transform can be performed without having to include pixels from outside the block for the computation of Haar-wavelet coefficients for all pixels in the block. Furthermore, this can be repeated up to $\lfloor \log_2 k \rfloor$ levels without affecting coefficients from outside the $k \times k$ block. Here we propose to use a mixed image transform. This transform consists of a Haar transform of three levels for occluded blocks and DCT for others with the DCT coefficients regrouped into the wavelet subbands to line up with the Haar-transformed coefficients.

7.4 Experimental Results

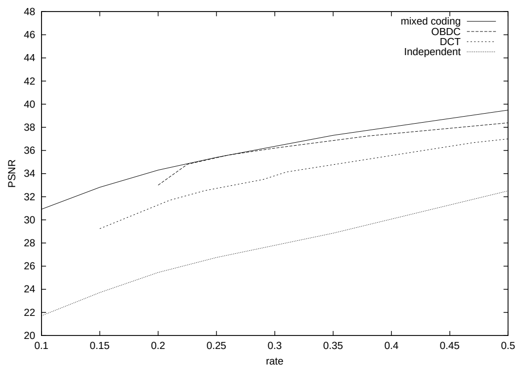
In our simulations, we used the 256×256 *Room* stereo image pair, and the Y component of the color stereo image pairs *Aqua* (360×288) and *Outdoors* (640×480) shown in Figure 7.1. The reference image was transformed using the 9-7 filters. For DE, a simple scheme was used with a 64-pixel horizontal search window. Occlusion detection consisted of looking for blocks where the estimation error was above a given threshold.

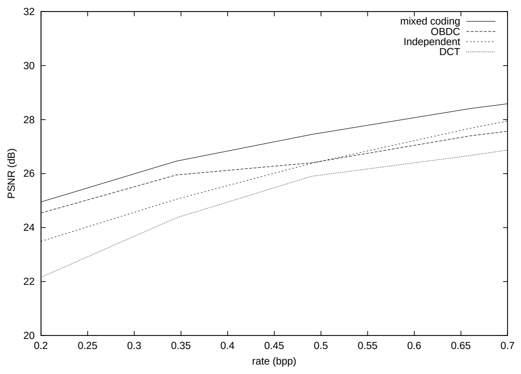
We present our results both visually and in terms of Peak Signal-to-Noise Ratio (PSNR). For stereo images, the PSNR is computed using the average of the mean squared error (MSE) of the reconstructed left and right images,

$$PSNR = 10 \log_{10} \frac{255^2}{(MSE_l + MSE_r)/2}.$$

First we compare different methods for the coding of the disparity estimated left image for the *Room* and *Aqua* pairs. The reference image is the JPEG coded (quality factor 75) right image. (This is chosen in order to be able to compare the results with previously published work [91].) The bit rate figures include the coding of the disparity vector field. In the case of the mixed transform, for each block an extra bit is encoded using context based arithmetic coding to signal if that block is considered as occluded. (In the case of independent coding there is no need to encode any disparity information.) The PSNR is computed using the MSE for the left image alone. The JPEG-style coder in our comparison uses quantization tables from the MPEG predicted frame coder.

Figure 7.5 compares independent coding, JPEG-style coding, overlapped block disparity compensation (OBDC) [91], and mixed transform coding. Mixed transform coding significantly outperforms both independent and JPEG-style coding with a gain of about 3 dB over the JPEG-style encoding. It also performs as well or better than OBDC coding which uses a computationally more complex disparity estimator.





a b

Figure 7.5: Comparison of independent coding, JPEG-style coding, OBDC, and mixed transform coding for the left image residual (with reference image JPEG-coded with quality factor 75) for (a) the *Room* and (b) the *Aqua* images.

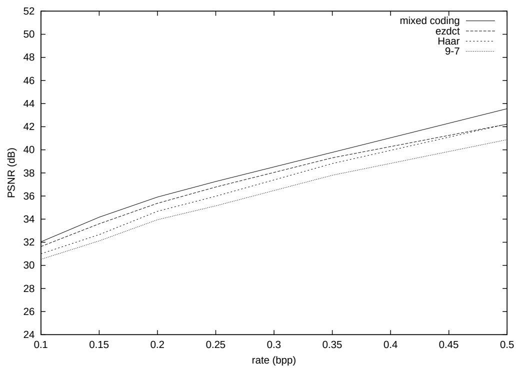


Figure 7.6: Comparison of 9-7 wavelet transform, Haar-transform, EZDCT-style, and mixed transform coding for the left image of the *Room* stereo pair.

Figure 7.6 shows the effect of different coding techniques of the residual image and compares their performance. As can be seen, the 9-7 wavelet filters perform poorest on the residual image, followed by Haar-filtering, EZDCT-style coding, and mixed transform coding. The differences range from 0.5 dB up to 1 dB for different bit rates between the pairs in the above ranking.

A similar comparison is given for the *Outdoors* image pair in Figure 7.7. These images contain fewer occluded areas, and the natural images are also harder to predict using such a simple disparity estimator. Using our mixed transform still produces up to 0.2 dB improvement over wavelet coding of the residual image. While the results for

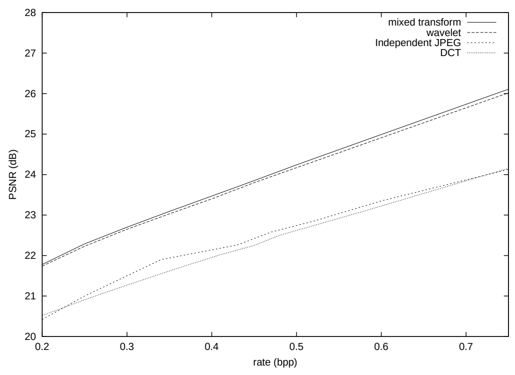


Figure 7.7: Comparison of independent coding, JPEG-style image coding, wavelet transform, and mixed transform coding for the left image of the *Outdoors* stereo pair.

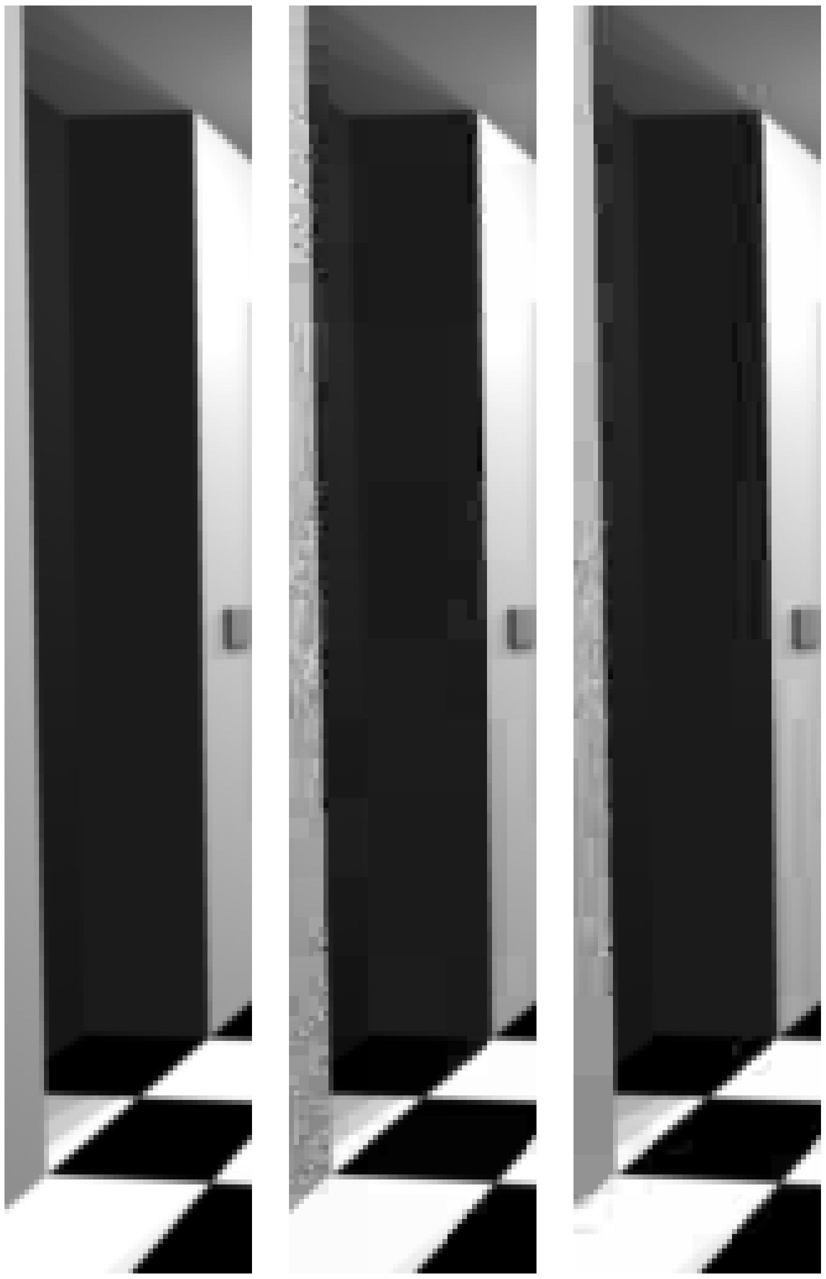
these two images improve the performance of DCT-based techniques they still fall short of the performance of individual wavelet coding of these images by about 0.2 dB.

Figure 7.8 demonstrates the effectiveness of the mixed transform. Occlusion in the left image occurs around its left edge. Figure 7.8(a) shows the original image, Fig. 7.8(b) shows the result compressed using the Haar transform, and Fig. 7.8(c) the outcome after using the mixed transform. The wall area is more uniform in Figure 7.8(c) because the mixed coder better preserved the occluded block.

For images that do not contain significant occluded information, the performance of the mixed transform coder is almost identical to that of the EZDCT-style coder.

Next we compare our proposed method and the results from [12] for the *Room* pair. Good residual image performance alone does not guarantee overall good performance when the entire stereo image is concerned in an embedded coding scenario. Recall that the decoder uses the compressed reference image to recreate the estimate for the predicted image. If the coding of the residual image takes away bits from the coding of the reference image, the overall result may not be as good as the coding of the residual image would suggest.

Figure 7.9 demonstrates this comparison. In this case, the left image is chosen as the reference image. In the comparison, “Boulgouris2” refers to new results [13] by



a b c

Figure 7.8: Occluded area of the left image of the *Room* stereo pair: (a) original, (b) compressed with the Haar transform at 0.15 bpp, and (c) compressed with the mixed transform 0.15 bpp.

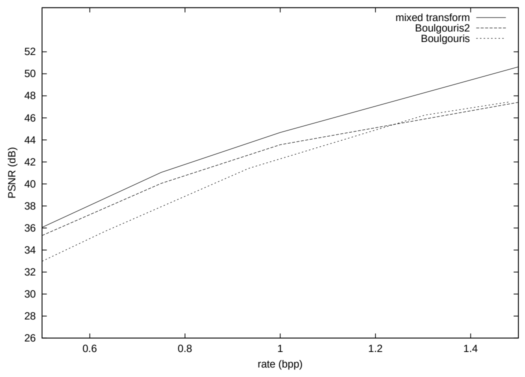


Figure 7.9: Comparison of proposed method with the Embedded Stereo Coding scheme from Boulgouris and its improved version for the full *Room* stereo pair.

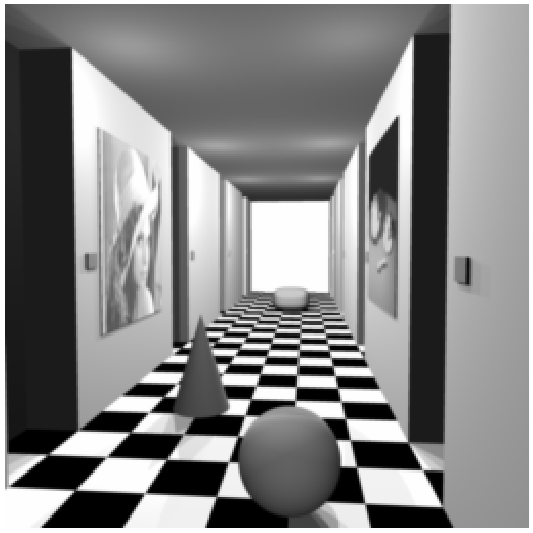
Boulgouris and Strintzis, obtained by an improved version of the original Embedded Stereo Coder. It uses half-pixel accuracy for disparity estimation and a compressed reference image to estimate the predicted image. Our proposed method outperforms this improved algorithm as well by 0.70 to 2.3 dB.

The original *Room* stereo image pair and its mixed transform compressed version at 0.5 bpp is presented in Figure 7.10. At this rate, there is little noticeable distortion between the original and the compressed images. To fully evaluate the effect of compression on stereo perception, one would need a stereo viewer to fuse these images.

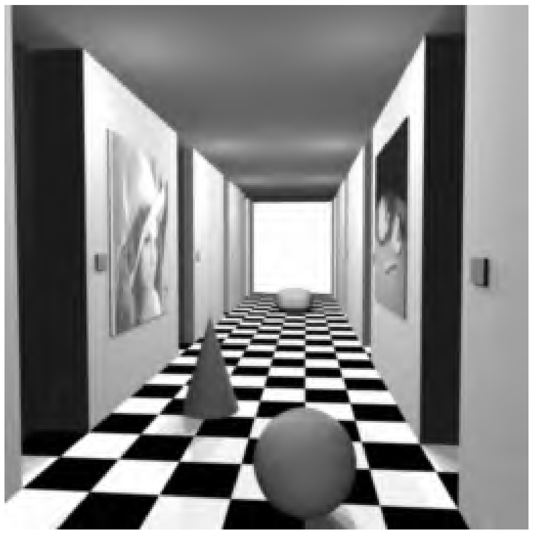
7.5 Summary

This work focused on the coding of the residual image in a stereo image compression scenario. Our method specifically addresses the issue of the image transform and the handling of the occluded blocks in the residual image. We showed that by individually coding the blocks of the residual image corresponding to the DE process, we can take advantage of the correlation properties of residual images. Occlusion is handled by foregoing estimation for those blocks whose prediction is very distorted. Using an embedded encoding scheme enables the encoding to be stopped at any given rate with-

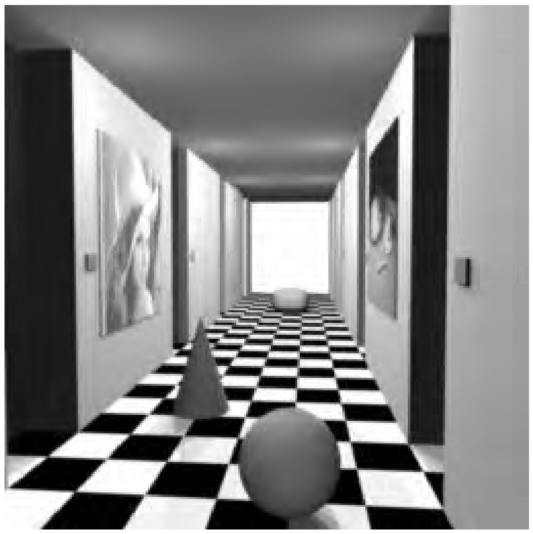
a



b



c



d

Figure 7.10: (a) and (b) The original *Room* image pair and (c) and (d) mixed transform compressed version of the *Room* image at 0.5 bpp.

out having to perform bit allocation. While the encoding is computationally simple, our simulations show improvements over previously published results.

In future research, this work can be extended to investigating the properties of residual images that result from more sophisticated DE techniques and applying some of the proposed methods to improve their coding, especially for the case of natural images.

This chapter, in part, is a reprint of the material as it appears in T. Frajka and K. Zeger. Residual Image Coding for Stereo Image Compression. *Optical Engineering*, 42(1):182-189, Jan. 2003. The dissertation author was the primary investigator of this paper.

Chapter 8

Disparity Estimation Window Size

Disparity estimation plays a crucial role in many stereo image compression techniques. To reduce computational complexity most, methods limit the estimation search area to a limited window. The performance of the disparity estimation depends on the choice of the limited search window. Most techniques use a predetermined value for the window size which is not optimal over a wide range of images. We show how the choice of the window size affects the performance of the stereo image compression algorithm and propose a method to obtain a better search window size. Our simulation results indicate an improvement of up to 1.81 dB over rigid window size selection and with performance very close to the optimal selection.

8.1 Introduction

Disparity estimation aims at finding the displacement of an object between the left and right images. Unlike in motion estimation, the displacement between the two images is restricted to a well defined direction for all parts of the image. Block matching based methods were used in [2, 93] and further reduction of complexity was achieved with hierarchical matching in [71, 96, 97] and by matching using selective sample decimation in [40]. To improve the matching performance of these techniques, others proposed more complex algorithms for disparity estimation, such as the use of overlapped blocks [91], the combination of block matching with subspace projection [8, 70], rate-distortion optimization [84], dynamic programming [35], and generalized block matching [69]. An overview of disparity estimation is given in Section 7.2.

To find the best match for any given block one would need to search over all blocks of the corresponding image pair. To reduce the complexity of this operation the matching is generally limited to a smaller window. In previous works this window size was some predetermined, fixed value. Because of the nature of the true disparity in images, such a predetermined value does not tend to work well across a wide range of images.

In this chapter we propose an efficient method for determining an estimate for the window size to be used with disparity estimation. This estimate is independent of the actual disparity estimation technique used and it reflects the underlying characteristics of the stereo image pair. We also show how the choice of window size affects the coding rate of the disparity vector field for both fixed rate and variable rate encoding. Our simulation results show that a proper search window size for disparity estimation can improve coding efficiency by up to 1.81 dB over using some predetermined value.

The effect of the search window size is analyzed in Section 8.2. Our method for window size estimation based on examining the correlation between the shifted image pairs is given in Section 8.3. Simulation results follow in Section 8.4 and we conclude with Section 8.5.

8.2 Disparity Window Size

Most disparity estimation algorithms use a predetermined, fixed maximum search window size, w. For example, the following window sizes or ranges of window sizes were used: 15 in [90], 63 in [9, 7], $\{-8, \ldots, 48\}$ in [70], $\{-30, \ldots, 30\}$ in [97], $\{-63, \ldots, 63\}$ in [39, 40], and some fixed unspecified values were used in [84, 35, 12, 36]. Many of these choices reflect that disparity estimation was modeled after motion estimation. As noted above, the disparity of each object is inversely proportional to its distance from the camera. A predetermined, fixed window size may not work well for any image.

The best disparity estimate is a function of the window size as given in (7.3). It is non-decreasing in w, since searching over a larger area can only improve the displacement estimate; i.e. if $w_1 < w_2$ then $\tilde{d}_{ij}(w_1) \leq \tilde{d}_{ij}(w_2)$ for all blocks in the image. Then, the disparity estimation distortion is non-increasing in w, i.e.

$$\tilde{D}_{pred}(\{\tilde{d}_{ij}(w_2)\}, \hat{I}_{ref}) \leq \tilde{D}_{pred}(\{\tilde{d}_{ij}(w_1)\}, \hat{I}_{ref}) \qquad if\ w_1 \leq w_2,$$

since searching for the best match over a larger area can only improve the outcome.

The DVF needs to be transmitted to the decoder for the reconstruction of the predicted image. The transmission rate, R_{dvf}, is a non-decreasing function of the disparity search window size, w. It is most obvious in the case of fixed length encoding where $\log_2 w$ bits are used to transmit a disparity value.

If the window size is smaller than the true disparity of certain objects in an image, the disparity estimation process cannot provide the best prediction, and the rate-distortion performance can be improved by increasing w. On the other hand, if the window size is significantly larger than the disparity of the objects in the image, their encoding may not be the most efficient. Even with DPCM coding followed by arithmetic coding for the DVF, too large a window size and thus too large a potential maximal value would dilute the probability model and thus yield a sub-optimal coding rate.

In [31] the authors showed that using adaptive arithmetic coding, the description length, R, of a source of alphabet size n, is

$$R = \log_2 \left(\frac{\prod_{i=0}^{t-1} (n+i)}{\prod_{i=1}^{k} c_i!} \right) \tag{8.1}$$

where k is the number of alphabet symbols that occur in the stream to be compressed, c_i is the number of occurrences of symbol i, and t is the total length of the stream. In the case of DVF coding, t equals the number of disparity vectors and k is the number of distinct displacement values. If only arithmetic coding is used, then $n = w$, and if it is coupled with DPCM coding, then $n = 2w + 1$.

Let d^*_{max} denote the maximum true disparity of any object in the image and $R_{dvf}(d^*_{max})$ the description length using an ideal disparity estimator. If one chooses a disparity window size larger than d^*_{max}, then the change in rate from (8.1) is

$$\begin{aligned} \triangle R_{dvf} &= R_{dvf}(w) - R_{dvf}(d^*_{max}) \\ &= \log_2 \frac{\prod_{i=0}^{t-1} (n_w + i) / \prod_{i=1}^{k'} c'_i!}{\prod_{i=0}^{t-1} (n^* + i) / \prod_{i=1}^{k} c_i!} \end{aligned} \tag{8.2}$$

where n^* and n_w are the number of possible input symbols to the arithmetic coding using window sizes of d^*_{max} and w, respectively; $k' \geq k$, i.e. the new disparity values found by increasing the window size from d^*_{max} to w are added after the initial k found using a window size of d^*_{max}; c_i and c'_i represent the occurrence of the same symbol, obtained using a search window size of d^*_{max} and w, respectively. For $k < i \leq k'$, $c_i = 0$ and for some $i \leq k$, c_i' can become zero. For such cases we adopt the usual convention that $0! = 1$ in (8.2).

Note, if $k' = k$, the increase in disparity window size does not affect the DVF and one can just transmit d_{max} to the decoder and only incur a small overhead penalty for choosing too large a disparity window size. Unfortunately, because of photometric variations and occlusion, the disparity estimation process often finds *false matches* for a block beyond the true disparity of the object.

Since $w \geq d^*_{max}$, let $w = d^*_{max} + \triangle_w$ and $n_w = n^* + \Delta_n$. To show that the rate does not increase with increasing window size, it suffices to show that the argument of the logarithm in (8.2) is greater than 1.

We have

$$\frac{\prod_{i=0}^{t-1}(n_w+i)}{\prod_{i=0}^{t-1}(n^*+i)} = \prod_{i=0}^{t-1}\frac{n^*+\Delta_n+i}{n^*+i} > \left(1+\frac{\Delta_n}{n^*+t-1}\right)^t \tag{8.3}$$

and

$$\begin{aligned}
\frac{\prod_{i=1}^{k} c_i!}{\prod_{i=1}^{k'} c_i'!} &= \prod_{i=1}^{k'}\frac{c_i!}{c_i'!} && (8.4)\\
&= \prod_{i:c_i>c_i'}\left((c_i'+1)\ldots c_i\right)\prod_{i:c_i<c_i'}\frac{1}{(c_i+1)\ldots c_i'}\\
&> \prod_{i:c_i>c_i'}(c_i'+1)^{c_i-c_i'}\prod_{i:c_i<c_i'}(1/c_i')^{c_i'-c_i}\\
&> (c'_{min,*}+1)^{\sum_{i:c_i>c_i'}(c_i-c_i')}(1/c'_{max,w})^{\sum_{i:c_i<c_i'}(c_i'-c_i)}\\
&= \left(\frac{c'_{min,*}+1}{c'_{max,w}}\right)^{\sum_{i:c_i>c_i'}(c_i-c_i')} && (8.5)
\end{aligned}$$

where $\sum_{i:c_i>c_i'}(c_i-c_i') = \sum_{i:c_i<c_i'}(c_i'-c_i)$, $c'_{min,*} = \min_{i:c_i>c_i'} c_i'$, and

$c'_{max,w} = \max_{i:c_i<c'_i} c'_i$.

Thus (8.3) and (8.5) imply that

$$\frac{\prod_{i=0}^{t-1}(n_w+i)/\prod_{i=1}^{k'}c'_i!}{\prod_{i=0}^{t-1}(n^*+i)/\prod_{i=1}^{k}c_i!} = \frac{\prod_{i=0}^{t-1}(n_w+i)}{\prod_{i=0}^{t-1}(n^*+i)}\frac{\prod_{i=1}^{k}c_i!}{\prod_{i=1}^{k'}c'_i!} \tag{8.6}$$

$$> (1+\frac{\Delta_n}{n^*+t-1})^t\left(\frac{c'_{min,*}+1}{c'_{max,w}}\right)^{\sum_{i:c_i>c'_i}(c_i-c'_i)}. \tag{8.7}$$

In practice, the changes in the occurrence count are around one or two and the number of all changes is typically less than one or two percent of all disparity vectors.

The first term on the right hand side of (8.7) is always greater than 1. The second term is approximately one if arithmetic coding is used without DPCM, or when DPCM and arithmetic coding are used, but all the displacement vectors found using the larger window size create new difference values.

For DPCM followed by arithmetic coding, the ratio $(c'_{min,*}+1)/c'_{max,w}$ can be less than 1 if the new displacement values create differences that existed for window size d^*. In that case, the lower bound in (8.7) is not useful in bounding the ratio in (8.6).

For this case, we evaluated the actual value of this ratio for 23 test images in the following experiment. For each stereo image pair, first the optimal window size d^* was estimated using exhaustive search disparity estimation with a window size equal to half the image size. Then d^* was chosen as the largest displacement value that was used for at least 1% of all blocks. Given the values of d^*, the ratio in (8.6) was computed using $w = d^*+1$, $w = d^*+5$, $w = d^*+10$, $w = d^*+50$, and $w = d^*+100$. For all images and for all window size choices, the ratio was observed to always be larger than 1. While this observation is not always guaranteed, for practical purposes we assume

$$\frac{\prod_{i=0}^{t-1}(n_w+i)/\prod_{i=1}^{k'}c'_i!}{\prod_{i=0}^{t-1}(n^*+i)/\prod_{i=1}^{k}c_i!} \geq 1$$

for both DPCM/arithmetic coding and plain arithmetic coding of the disparity vector field.

The choice of the optimal window size is image dependent. Since d^*_{max} is generally unavailable at the time the images are compressed, it is necessary to be able to find a good maximum in real time without having to perform an exhaustive search. Using predetermined, fixed values does not work well over a wide range of images, as is shown in Tables 8.1 and 8.2. These results were obtained for the following three images: the 256×256 synthetic *Room* stereo image pair, and the 640×480 *Outdoors* and *Closeup* image pairs (Figure 8.1). The disparity estimation and the encoding are the same as in [26], where the DVF was encoded using DPCM followed by adaptive arithmetic coding. The optimal window size was determined using an exhaustive search. It is optimal given the encoding mechanism and the target bit rate. The choice of a predetermined window size of 64 displacement values was motivated by previous results in the literature using that value. The *peak signal-to-noise ratio* (PSNR) for the predicted image alone (Table 8.1) is defined as

$$PSNR = 10\log_{10}\frac{255^2}{D_{pred}} \tag{8.8}$$

and for the stereo image pair (Table 8.2) as

$$PSNR = 10\log_{10}\frac{255^2}{(D_{ref}+D_{pred})/2}. \tag{8.9}$$

The notation $PSNR(x)$ in the tables refers to the PSNR obtained using a maximum window size x in the disparity estimation process.

These results indicate that using the optimal window size is always better than using a predetermined one. In the case of the *Closeup* and *Room* images, the improvement ranges between 0.31 dB and 1.76 dB for the predicted image alone, and 0.09 dB and 0.95 dB for the stereo image pair. Note that with the *Room* stereo pair, the optimal window size is smaller than the predetermined value, while with the *Closeup* image it is larger.

8.3 Determining Disparity Window Size

Conducting a full search over all possible disparity search window size values is a time consuming process. It is also analytically difficult since the disparity and the resulting distortion are image dependent. Even quick search methods are difficult to implement because the distortion at a given target rate is not a monotone, concave, or convex function of the disparity window size, as revealed in Figure 8.2.

Here we propose a heuristic approach that yields good results for many different stereo image pairs.

In [93] the authors use correlation measurements between the shifted left and right images of the stereo pair to determine a global disparity vector. The maximum of the correlation is assumed at the "average" disparity of the image. While using just a single value for the entire image is not the optimal strategy, it still gives an indication of the typical disparity values to be found in the image pair.

We use a similar strategy based on the correlation coefficient between two images:

Table 8.1: Effect of disparity window size choice on predicted image PSNR alone. *L or R* indicates whether the left or right image was predicted using the uncompressed version of the other image. The rate is the sum of the rate of the DVF (coded using DPCM and arithmetic coding) and the difference image.

image	L or R	rate	w_{opt}	$PSNR(64)$	$PSNR(w_{opt})$
		(bpp)		(dB)	(dB)
Room	L	0.1	10	30.51	31.53
Room	R	0.1	10	32.39	32.70
Closeup	L	0.2	171	24.03	25.49
Closeup	R	0.2	262	23.95	25.71
Outdoors	L	0.2	434	21.74	21.82
Outdoors	R	0.2	386	21.77	21.85

Table 8.2: Effect of disparity window size choice on overall image quality. *L or R* indicates whether the left or right image was predicted using the compressed version of the other image. The rate is the sum of the rate of the DVF (coded using DPCM and arithmetic coding), the difference image, and the reference image.

image	L or R	rate	w_{opt}	$PSNR(64)$	$PSNR(w_{opt})$
		(bpp)		(dB)	(dB)
Room	L	0.2	10	27.45	27.93
Room	R	0.2	10	27.93	28.02
Closeup	L	0.4	177	29.48	30.33
Closeup	R	0.4	273	29.27	30.22
Outdoors	L	0.4	91	23.18	23.20
Outdoors	R	0.4	97	23.16	23.17

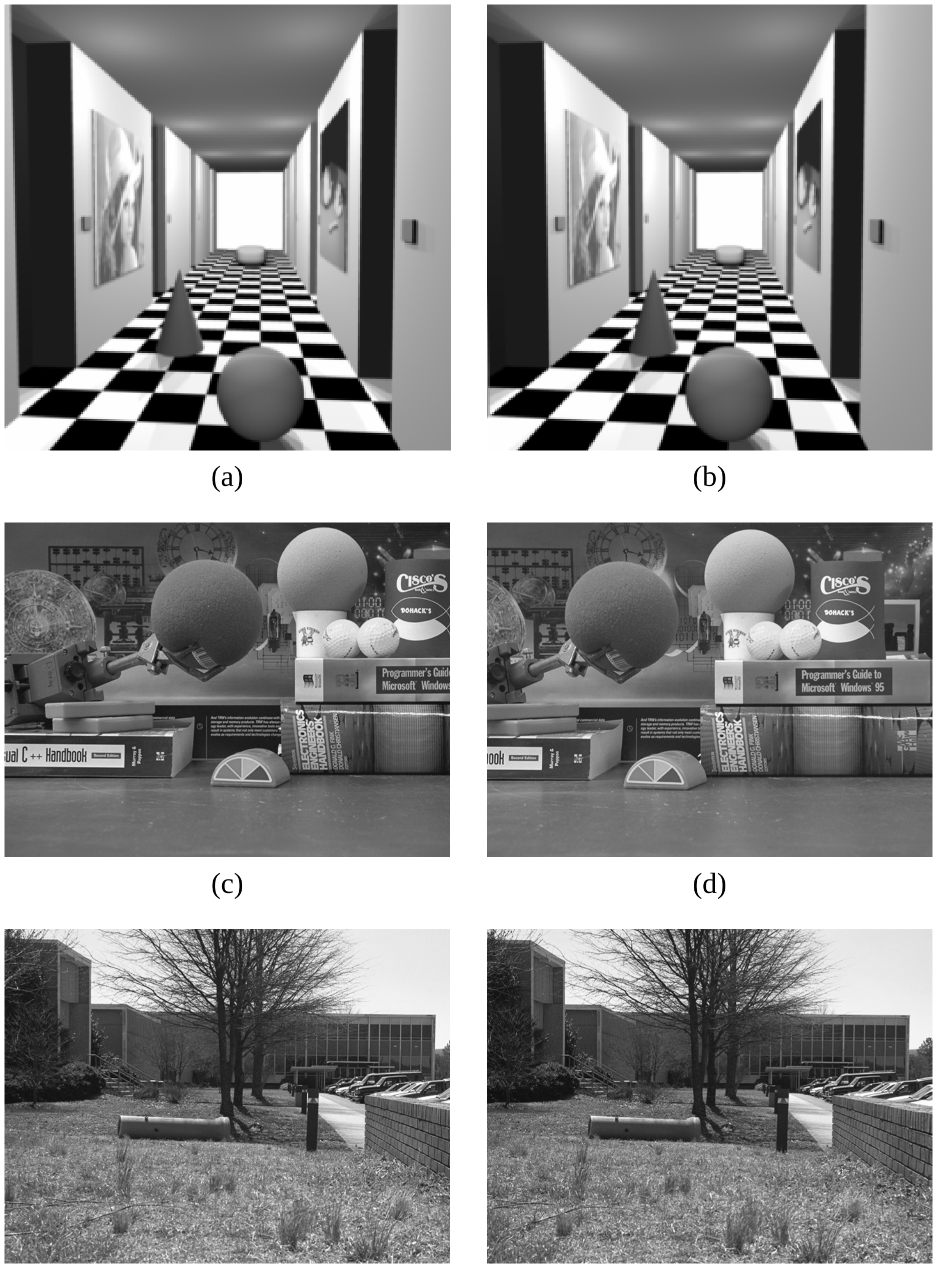


Figure 8.1: Original images of the (a)-(b) *Room*, (c)-(d) *Closeup*, and (e)-(f) *Outdoors* stereo pairs.

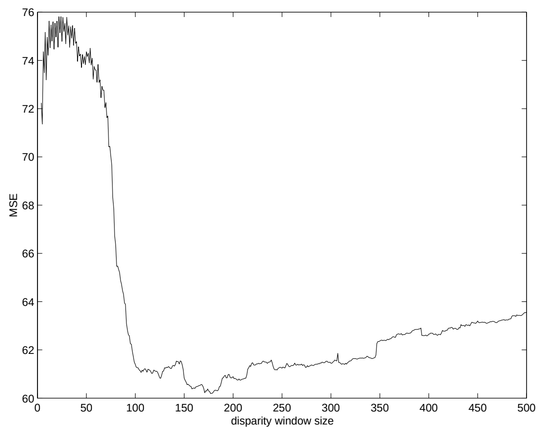


Figure 8.2: Distortion D_T as a function of maximum window size w for the *Closeup* stereo image pair at an overall bit rate of 0.4 bpp.

$$C(I_1, I_2) = \frac{Cov(I_1, I_2)}{\sigma_{I_1}\sigma_{I_2}}$$

where I_1 and I_2 are images given by their intensity values, and $Cov(.)$ is the covariance (the expectations are taken over the sample distribution of the images). As one image is shifted with respect to the other, columns at the leading end of the shifting move out of the image and columns on the trailing end need to be filled. We propose to fill those using a mirroring of the columns at the trailing end. This is a natural extension of the disparity estimation process. In disparity estimation, one tries to find matching blocks along the same horizontal direction for each block. However, at the far edge of the matching, that direction points outside the image at which point the estimation will look for matches in the reverse direction.

This is a computationally complex process since the correlation is determined for each disparity value. We perform the correlation computation on a subsampled version of the images instead. For computational simplicity, the subsampling is carried out as a succession of averaging. Using images subsampled by 8 in each direction reduces the computational cost by a factor of 64. Let $C_8(d)$ denote the correlation value for shift d when the images subsampled by 8 in each direction are used. Once these correlation values are obtained, the algorithm finds the maximum value, $C_{8,max} = \max_d C_8(d)$. Let d denote the value after which the correlation drops below $C_{8,max}/2$, that is $C_8(d) \geq C_{8,max}/2$, but $C_8(d+1) < C_{8,max}/2$. The window size is chosen as $w_C = 8d$, the displacement corresponding to d in the full resolution image. The particular choice of the factor of $1/2$ as the cutoff value was motivated by experiments that indicated good approximation of the optimal disparity window size given the DVF encoding method and the target rate.

This results in a less accurate estimate of the window size as it is quantized to the subsampling factor, but it allows a faster computation.

8.4 Simulation Results

We tested our proposed method over a large set of test images. The PSNR is computed as defined in (8.8) and (8.9). Tables 8.3 and 8.4 show the results using the correlation based estimation for maximum window size. Two different scenarios of pre-determined window sizes are shown as well. The correlation based approximation works very well, especially when compared with pre-determined window sizes. Neither of the pre-determined values show a clear advantage over the other pre-determined value, indicating that one fixed choice does not work for all images. Our proposed technique yields the best performance on most images, outperforming the fixed choice of 64 in 82% of all all cases, and the fixed choice of 200 in 78% of all cases. Our results trail the best achievable result given by exhaustive search for the given DVF coding technique by a range of 0.02 dB to 1.01 dB.

When compared with a scheme with a predetermined window size this method increases complexity due to the correlation computation. This increase is far less than performing a full search disparity compensation and encoding of the DVF and the residual image to find the true optimum value.

8.5 Summary

We presented a technique that estimates the optimal choice for the disparity estimation search window size. It can be combined with any particular disparity estimation algorithm. The performance improvement due to an adaptive choice of search window is up to 1.81 dB. While it leads to some increase in computation complexity, our proposed method is still computationally less complex than trying to find the optimal window size using an exhaustive search.

This chapter, in part, has been submitted for publication as: T. Frajka and K. Zeger. Disparity Estimation Window Size. *Optical Engineering*, Dec. 2002. The dissertation author was the primary investigator of this paper.

Table 8.3: Comparison of predetermined, optimal, and approximated disparity window size choices on the predicted right image quality at 0.2 bpp using the left image as a reference image. The rate is the sum of the rate of the DVF (coded using DPCM and arithmetic coding) and the difference image.

Image	Size	w_{opt}	w_C	$PSNR$ (w_{opt}) (dB)	$PSNR$ (w_C) (dB)	$PSNR$ (64) (dB)	$PSNR$ (200) (dB)
Toys	212×134	12	32	30.14	$\mathbf{29.36}$	28.98	28.79
Fruit	512×512	503	32	34.67	$\mathbf{34.26}$	34.04	33.66
Whgarden	250×250	29	24	25.98	$\mathbf{25.89}$	25.70	25.46
Cart	250×250	69	64	30.94	$\mathbf{30.88}$	$\mathbf{30.88}$	30.35
Parts	512×512	456	264	35.40	$\mathbf{33.91}$	33.43	33.68
Rubik	512×512	4	40	42.44	$\mathbf{41.89}$	41.86	41.77
Arch	512×512	4	24	42.55	$\mathbf{41.39}$	40.35	40.53
Room	256×256	10	16	35.73	$\mathbf{35.68}$	35.55	35.15
Closeup	640×480	262	128	25.71	$\mathbf{25.53}$	23.95	25.49
Outdoors	640×480	386	120	21.85	21.80	21.77	$\mathbf{21.81}$
Oldbridge	320×192	317	24	25.55	$\mathbf{25.48}$	25.40	25.27
Ball	512×512	510	24	43.53	$\mathbf{41.99}$	41.39	41.49
Book1r	250×250	6	40	34.46	$\mathbf{33.32}$	32.96	32.55
Plants	512×400	112	120	30.29	$\mathbf{30.27}$	25.78	30.14
Cart-alt	250×250	57	40	32.20	$\mathbf{32.10}$	32.05	31.66
Bottle	320×240	17	48	27.53	$\mathbf{27.24}$	27.09	26.65
Apple	512×512	504	192	26.09	25.39	$\mathbf{25.49}$	25.39
Manege	720×288	31	72	26.77	26.50	$\mathbf{26.54}$	26.07
Book	512×512	4	88	37.45	35.50	$\mathbf{35.67}$	35.18
Aqua	360×288	251	72	26.04	25.52	25.47	$\mathbf{25.88}$
Sphere	256×256	252	80	33.17	32.61	30.17	$\mathbf{32.97}$
Tunel	720×288	509	48	27.31	25.58	25.64	$\mathbf{26.46}$
Fjord	233×256	4	120	28.60	28.35	27.93	$\mathbf{28.54}$

Table 8.4: Comparison of predetermined, optimal, and approximated disparity window size choices on overall image quality at 0.4 bpp using the left image as a reference image. The rate is the sum of the rate of the DVF (coded using DPCM and arithmetic coding), the difference image, and the reference image.

Image	Size	w_{opt}	w_C	$PSNR$ (w_{opt}) (dB)	$PSNR$ (w_C) (dB)	$PSNR$ (64) (dB)	$PSNR$ (200) (dB)
Toys	212×134	12	32	33.76	**33.14**	32.77	32.77
Fruit	512×512	5	32	38.19	**37.92**	37.77	37.59
Whgarden	250×250	33	24	28.08	**28.02**	27.96	27.90
Cart	250×250	70	64	35.11	**35.10**	**35.10**	34.76
Parts	512×512	456	264	39.62	**39.01**	38.92	38.95
Rubik	512×512	5	40	46.89	**46.66**	46.64	46.54
Arch	512×512	4	24	45.16	**44.75**	44.71	44.70
Room	256×256	11	16	33.54	**33.52**	33.46	33.28
Closeup	640×480	273	128	30.22	**30.18**	29.27	30.13
Outdoors	640×480	97	120	23.17	**23.17**	23.16	23.16
Oldbridge	320×192	11	24	26.80	**26.76**	26.70	26.64
Ball	512×512	506	24	45.34	**44.99**	44.81	44.73
Book1r	250×250	5	40	37.85	**37.24**	37.07	36.82
Plants	512×400	109	120	33.87	**33.84**	32.03	33.77
Cart-alt	250×250	25	40	36.40	**36.32**	36.21	36.05
Bottle	320×240	18	48	26.26	**26.18**	26.12	25.95
Apple	512×512	504	192	27.54	27.22	**27.30**	27.23
Manege	720×288	31	72	28.18	28.09	**28.11**	27.97
Book	512×512	4	88	42.34	41.33	**41.44**	41.08
Aqua	360×288	266	72	26.02	25.80	25.78	**25.95**
Sphere	256×256	243	80	32.99	32.81	31.98	**32.93**
Tunel	720×288	510	48	28.85	28.13	28.13	**28.48**
Fjord	233×256	4	120	30.16	29.80	29.68	**29.86**

Chapter 9

Downsampling Dependent Upsampling of Images

Downsampling an image results in the loss of image information that cannot be recovered with upsampling. We demonstrate that the particular combination of downsampling and upsampling methods used can significantly impact the reconstructed image quality, and then we propose a technique to identify patterns associated with different downsampling methods in order to select the appropriate upsampling mechanism. The technique is low complexity and achieves high accuracy over a wide range of images.

9.1 Introduction

Digital imagery can be viewed on various different display sizes, depending on the electronic device being used (e.g. computer monitor, laptop computer screen, PDA, cell phone, etc.). Similarly, digital cameras offer a wide range of image resolutions, yielding images of various different sizes.

The choice of display size is typically determined by some constraint on the device, such as its size, price, quality, etc. Often small display sizes are used on devices with limited amounts of on-board memory. If a large amount of memory is available, a full resolution image can be stored and locally subsampled for viewing on the given screen size. However, if the on-board memory is tightly limited, this may not be a feasible option. In such a case, typically the device could either store the appropriate resolution image or only some portions of the original.

A lower resolution image can be obtained by manipulating the image in the spatial domain. A comparison of some of these techniques for image resampling is given in [58]. When transmitted over limited bandwidth communication channels, images are often compressed to conserve resources.

Image compression algorithms need to allow flexibility in choosing a decompression resolution. This property is called “spatial scalability”. With spatial scalability, different devices can decode different resolution versions of the same image without having to encode the same image to several different decoding resolutions. Both the original JPEG standard and the more recent standard, JPEG 2000, have built-in modes for spatial scalability.

Once a lower resolution image is decoded, a display device typically does not have access to the full resolution image any more. If the user wants to resend the image to a different receiver whose device is capable of displaying the image at a higher resolution, the image needs to be upsampled at the receiver to make full use of the display’s capabilities.

A challenging task in image upsampling is to best preserve the sharpness of the edges in the image. Traditional spline-based methods [30, 38, 86] result in sharper edge reconstruction when compared with linear filtering approaches. Yang and Truong [94] proposed interpolated Mth band filters for image size conversion and achieve some improved image quality. Edge-directed methods [1, 34] seek to identify edges at a subpixel level in the downsampled image and avoid their smoothing in the resizing operation. Atkins et al. gave two methods [4, 5] that aimed to locally find an appropriate filter for the scaling of each pixel. Both methods require training for the classifier and the selection of each filter's parameters, but they differ in the particular classification technique used.

All of the above techniques process an image in the spatial domain. A transform domain approach was taken by Chang et al [15]. Their solution is based on the evolution of the local minima and maxima in the different frequency bands of the wavelet transformed image. These local extrema corresponding to edges in the spatial domain are used to estimate the high frequency coefficients that are lost in the downsampling process.

Dugad and Ahuja [25] introduced an image resizing method using the Discrete Cosine Transform (DCT) that showed improved image quality when compared with bilinear spatial domain interpolation. They also found that this method performs well on images that were downsampled using bilinear interpolation. Mukherjee and Mitra [55] presented a modification of this technique based on subband DCT [37] and extended Dugad and Ahuja's method to color images.

In this paper, we confirm that the performance of the upsampling process depends on the particular upsampling method as well as on the downsampling method used to obtain the lower resolution image. We propose a method that estimates the type of downsampling method used by looking for "signatures" of different techniques in the downsampled image. Using this type-estimate we choose a specific upsampling method that results in improved image quality. In many cases, the different upsampling methods result in seemingly similar images, but these images may differ by $1 - 8.5$ dB when compared with the original image. The differences usually occur in the reconstruction of the image edges which can be visually significant if the upsampled image is subject to further image processing or edge detection-based image analysis.

Section 9.2 demonstrates the importance of using the appropriate upsampling method for a given downsampled image. Section 9.3 describes our proposed method. Simulation results are shown in Section 9.4 and we conclude in Section 9.5.

9.2 Subsampling Dependent Upsampling

Subsampling followed by upsampling is an inherently lossy process, so that the resulting image will differ from the original. The information lost in downsampling cannot in general be fully recovered. With a correct choice of an upsampling method, one can avoid introducing further distortion into the reconstruction process.

In order to demonstrate that different combinations of downsampling and upsampling techniques yield different results we used the following experiment. A set of test images were downsampled to one fourth of their original size (i.e. each dimension was divided in half) and then resampled to the original resolution. The following three methods were used for subsampling:

- **bilinear**: The subsampled value is the average of the four corresponding intensity values.

- **DCT**: For each 8×8 block in the original image, a DCT is performed. Using only the top left 4×4 block of DCT coefficients in each 8×8 block, a 4×4 inverse DCT is performed (see [25] for more details).

- **wavelet**: A single level wavelet transform is performed using 9-7 filters [3]. The appropriately scaled LL band gives the subsampled image.

Similarly the upsampling methods in the experiment are the following:

- **bilinear**: The upsampling inserts zeros between every two samples of the down-sampled image and performs a bilinear interpolation.

- **DCT**: The image is processed in 4×4 blocks, performing the DCT on each block. The 4×4 DCT blocks are then placed in the upper left corner of an 8×8 block of all zeros and the inverse DCT is performed on the 8×8 blocks (see [25]).

- **wavelet**: The image to be upsampled is taken (with appropriate scaling) to be the LL band of a single level wavelet transform with all high frequency bands set to zero and the inverse wavelet transform is performed using 9-7 filters.

The tables in Appendix 9.6 indicate that the particular subsampling-upsampling combination can have a significant quantitative effect on the PSNR of the reconstructed image. Depending on the image and the combinations, the difference between the worse and best can be up to 8.5 dB! It is also clear that bilinear interpolation-based upsampling is never optimal. DCT-based upsampling is preferred for images that result from DCT-based or bilinear subsampling. Wavelet upsampling performs best for images obtained with wavelet-based downsampling. This particular combination achieved the best overall PSNR for 84% of the test images. The PSNR results are important in applications where the image is further subjected to some image processing or if it is used for differential coding.

For human observers, the visual image quality is important. Figure 9.1 shows a comparison between the reconstructed *Lena* images. The downsampling method is the same (wavelet-based), and the difference is in the upsampling: wavelet-based versus DCT-based. Even though wavelet-based upsampling results in an image that is 5.06 dB better than the DCT-based image, the differences are visually very small.

Figure 9.2 shows where the two images differ. It was obtained by rescaling the difference image between images (a) and (b) in Figure 9.1. Pixels that are medium gray

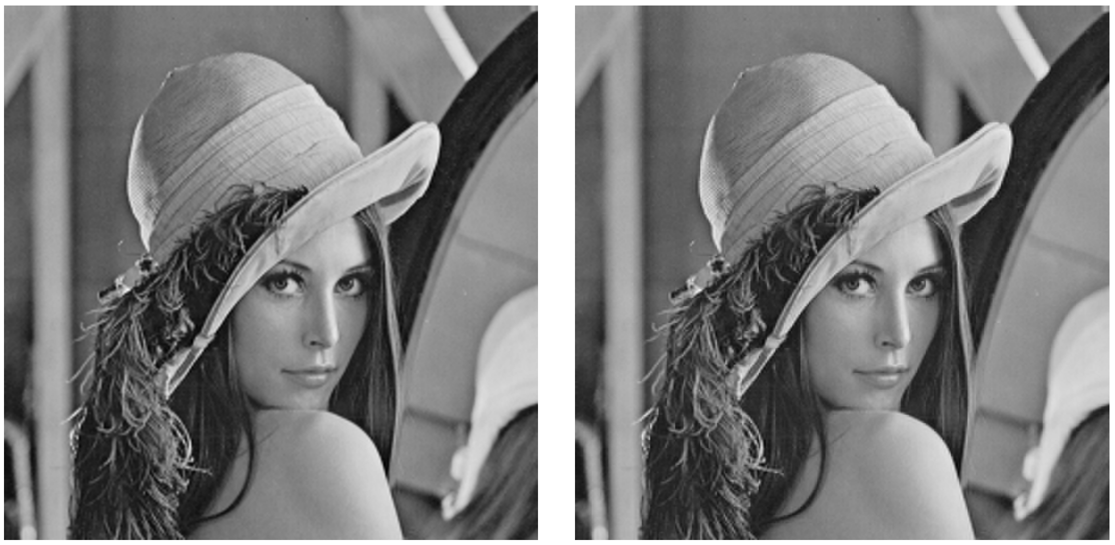


Figure 9.1: Reconstructed *Lena* image with wavelet-based downsampling and (a) wavelet-based upsampling (35.25 dB), (b) DCT-based upsampling (30.19 dB).

indicate where the two images are identical; darker and lighter pixels show where they differ. The differences occur around the edges, while the smooth areas are reconstructed the same. While the shifting of the edge transitions is visually not very noticeable, such changes represent a significant contribution to the PSNR.

9.3 Proposed Algorithm

The different subsampling techniques yield slightly different images at lower resolution. Most of the differences concentrate around the edge regions of these images. All three techniques discussed here can be thought of as a filtering followed by decimation. Figure 9.3 shows the filtered step edge (0 to 1 and 1 to 0 transitions) profiles before decimation for the bilinear, DCT, and wavelet methods. In the bilinear and wavelet cases, the filters are given. In the DCT case, the filter response was compiled from the subsampled version of two step edges that are shifted one position with respect to each other. (Note that in the DCT case the response also slightly varies depending on the location

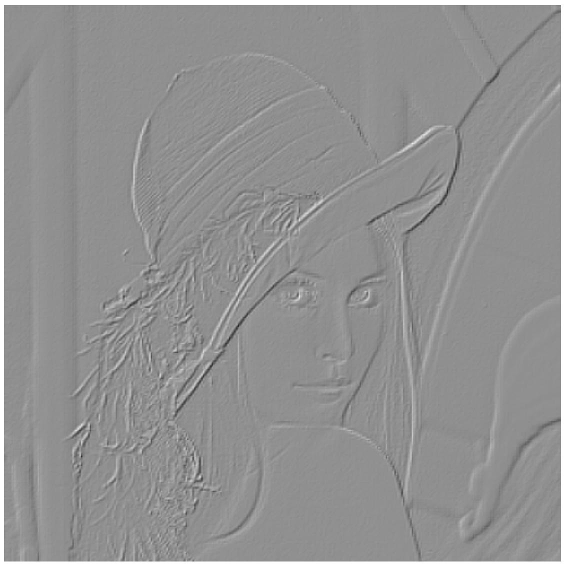


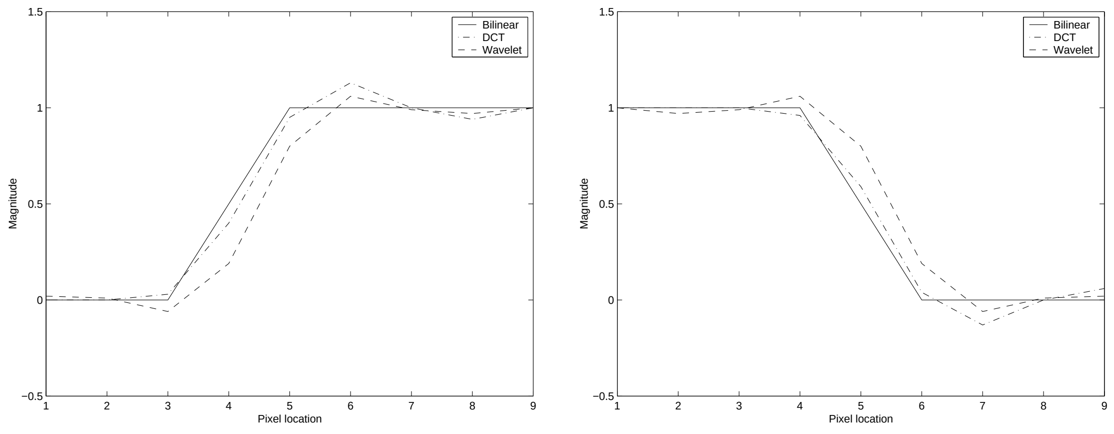
Figure 9.2: The rescaled difference of images Fig. 9.1 (a) and (b). Medium gray levels indicate where the two are identical. The differences are the dark and bright colored areas.

of the edge within the boundaries of the 8×8 blocks.)

These figures indicate that the edge response of the bilinear and DCT cases are similar in slope, being close to a smooth linear transition. The wavelet response trails below the linear slope in the case of the rising edge, while it goes above the linear slope in the case of the falling edge.

In the downsampled image only half of these edge points are present. Depending on the location of an edge in the image, it is represented by either the odd or the even samples of these transitions.

In our proposed method we use blocks formed from these odd and even samples as "signatures" of the downsampling method to be identified. In the case of a vertical edge, identical rows of the samples form the block, while for horizontal edges the samples are placed in the columns. To avoid false matches the number of these identical columns or rows should be greater than one. However, very few real life images have



(a) (b)

Figure 9.3: Edge response of the bilinear, DCT, and wavelet “filtering” for (a) a rising step edge, (b) a falling step edge. These curves indicate the edge profiles. The values are only given at the pixel locations.

long vertical or horizontal step edges. Thus in practical cases the blocks should be kept relatively small in order to find reliable matches. We found that a block size of 4×4 presents a good trade-off between these two constraints. The bilinear and DCT patterns form one collection of signatures (Table 9.1), and the wavelet patterns form another collection (Table 9.2).

For each image to be upsampled, the algorithm computes the correlation coefficient between blocks of the image and the signature patterns of the bilinear/DCT collection and those of the wavelet collection. The correlation coefficient between blocks B_1 and B_2 is

$$C(B_1, B_2) = \frac{Cov(B_1, B_2)}{\sigma_{B_1} \sigma_{B_2}}$$

where $Cov(.)$ is the covariance (the expectations are taken over the sample distribution of the blocks).

For each block, the highest correlation is chosen for the given collection. After the correlation values have been computed for an image, the technique compares the number of blocks where the correlation coefficient is above a given threshold for each

Table 9.1: 4×4 pattern collection for the bilinear/DCT-based case.

1	1	0	0	1	1	1	1	1	.59	-.13	.06
1	1	0	0	1	1	1	1	1	.59	-.13	.06
1	1	0	0	0	0	0	0	1	.59	-.13	.06
1	1	0	0	0	0	0	0	1	.59	-.13	.06
1	.5	0	0	1	1	1	1	0	0	.4	1.13
1	.5	0	0	.5	.5	.5	.5	0	0	.4	1.13
1	.5	0	0	0	0	0	0	0	0	.4	1.13
1	.5	0	0	0	0	0	0	0	0	.4	1.13
0	0	1	1	0	0	0	0	1	1	1	1
0	0	1	1	0	0	0	0	.59	.59	.59	.59
0	0	1	1	1	1	1	1	-.13	-.13	-.13	-.13
0	0	1	1	1	1	1	1	.06	.06	.06	.06
0	0	.5	1	0	0	0	0	0	0	0	0
0	0	.5	1	0	0	0	0	0	0	0	0
0	0	.5	1	.5	.5	.5	.5	.4	.4	.4	.4
0	0	.5	1	1	1	1	1	1.13	1.13	1.13	1.13

Table 9.2: 4×4 pattern collection for the wavelet-based case.

.97	1.06	.19	.01	.97	.97	.97	.97
.97	1.06	.19	.01	1.06	1.06	1.06	1.06
.97	1.06	.19	.01	.19	.19	.19	.19
.97	1.06	.19	.01	.01	.01	.01	.01
.99	.8	-.06	.02	.99	.99	.99	.99
.99	.8	-.06	.02	.8	.8	.8	.8
.99	.8	-.06	.02	-.06	-.06	-.06	-.06
.99	.8	-.06	.02	.02	.02	.02	.02
0	.01	.19	1.06	0	0	0	0
0	.01	.19	1.06	.01	.01	.01	.01
0	.01	.19	1.06	.19	.19	.19	.19
0	.01	.19	1.06	1.06	1.06	1.06	1.06
.02	-.06	.8	.99	.02	.02	.02	.02
.02	-.06	.8	.99	-.06	-.06	-.06	-.06
.02	-.06	.8	.99	.8	.8	.8	.8
.02	-.06	.8	.99	.99	.99	.99	.99

collection of patterns. The “estimate” of the downsampling technique is declared to be the one with the higher number of such blocks.

9.4 Simulation Results

Our proposed method was tested using 47 test images. The images vary in size and content. They include natural indoors and outdoors images, close-up head images, synthetic images, satellite pictures, video scenes, and texture images.

In order to evaluate the accuracy of our technique, each image was downsampled by a factor of four using all three methods and the correlation-based matching technique was applied to each outcome. To speed up the computation, the correlation was only computed for blocks where the variance was above a given threshold indicating the possible presence of an edge. In these experiments, a match is declared for a block (i.e. it is included in the count for the given pattern collection) if the correlation exceeds $.992$. This choice was motivated by our experiments and also ensures that only very strong matches are selected.

Table 9.4 in Appendix 9.7 lists the counts for each image and each outcome with the counts of the bilinear/DCT collection listed first and the wavelet collection listed second. For the bilinear and DCT cases, a good identification is given if the first number is greater. For the wavelet case, the second number should be greater. In the case of a tie, the decision is for the wavelet method. The bold image name indicates the case where all three cases have been successfully identified.

As can be seen in that table, the number of matches at this level of fidelity varies greatly. For some images it is in the thousands (mostly synthetic images that contain more clear step edge patterns) while for others no match can be found at all.

The particular pattern choices yielded an overall accuracy of 66% for correctly identifying all three downsampling methods. The wavelet method was correctly found in 70% of all cases, the bilinear in 89%, and for the case of DCT, the method had an accuracy percentage of 91%.

9.5 Summary

In this chapter we showed that the image quality after upsampling depends on both the downsampling method used to get the lower resolution image and the upsampling technique. We introduced a correlation-based technique for the identification of the subsampling method. Using a simple edge pattern, our method was able to accurately identify the downsampling method in 66% of the images used for all three downsampling techniques.

In future research more varied patterns can be used as well as investigating other differences in the edge regions of the downsampled images for improved accuracy.

This chapter, in full, has been submitted for publication as: T. Frajka and K. Zeger. Downsampling Dependent Upsampling of Images. *Signal Processing: Image Communication*, Apr. 2003. The dissertation author was the primary investigator of this paper.

Appendix

9.6 Results of different upsampling and downsampling combinations

Table 9.3: PSNR (dB) comparison of different downsampling-upsampling combinations. The rows represent the downsampling method, and the columns the upsampling technique.

Image	Method	Bilinear	DCT	Wavelet
	Bilinear	24.356	26.196	24.035
Aerial	DCT	23.999	26.809	23.887
	Wavelet	22.535	23.695	26.619
	Bilinear	28.503	29.289	27.662
Aerial2	DCT	28.123	29.794	27.635
	Wavelet	26.621	27.313	30.122
	Bilinear	33.56	34.147	32.899
Apc	DCT	33.189	34.617	32.826
	Wavelet	32.024	32.583	34.808
	Bilinear	25.489	25.931	24.596
Aqua	DCT	25.112	26.391	24.554
	Wavelet	23.817	24.295	26.557
	Bilinear	23.74	24.01	22.86
Baboon	DCT	23.33	24.49	22.84
	Wavelet	22.19	22.58	24.50
	Bilinear	29.431	30.975	28.839
Balloon	DCT	29.134	31.459	28.743
	Wavelet	27.416	28.435	31.983

Table 9.3: Continued

Image	Method	Bilinear	DCT	Wavelet
	Bilinear	25.413	25.173	24.178
Barbara	DCT	24.955	25.657	24.272
	Wavelet	23.721	23.899	25.849
	Bilinear	27.427	28.369	26.256
Beach	DCT	27.073	28.86	26.194
	Wavelet	25.22	25.907	29.206
	Bilinear	25.67	26.48	24.84
Bridge	DCT	25.31	26.97	24.81
	Wavelet	23.85	24.56	27.19
	Bilinear	18.415	18.985	17.627
Bwall	DCT	18.057	19.456	17.641
	Wavelet	16.799	17.406	19.618
	Bilinear	25.486	26.275	24.31
Camera	DCT	25.141	26.763	24.287
	Wavelet	23.444	24.094	26.954
	Bilinear	30.756	32.986	30.654
Coral	DCT	30.434	33.593	30.528
	Wavelet	28.914	30.354	33.973
	Bilinear	29.643	32.668	28.751
Crowd	DCT	29.36	33.307	28.561
	Wavelet	27.017	28.41	34.019
	Bilinear	33.683	33.685	30.312
Dna	DCT	33.329	34.058	30.375
	Wavelet	30.655	30.889	33.282
	Bilinear	31.765	32.964	31.19
Elaine	DCT	31.47	33.387	31.131
	Wavelet	30.122	31.044	33.502

Table 9.3: Continued

Image	Method	Bilinear	DCT	Wavelet
	Bilinear	29.959	30.801	29.148
Fence	DCT	29.611	31.264	29.084
	Wavelet	28.171	28.863	31.484
	Bilinear	27.551	32.469	27.133
Finger	DCT	27.281	33.435	26.841
	Wavelet	25.024	27.015	33.599
	Bilinear	26.52	28.415	25.989
Front	DCT	26.254	29.001	25.965
	Wavelet	24.376	25.567	29.615
	Bilinear	30.37	31.351	29.414
Goldhill	DCT	29.988	31.885	29.373
	Wavelet	28.384	29.172	32.096
	Bilinear	37.39	38.86	37.01
Gray21	DCT	37.53	39.08	37.19
	Wavelet	35.87	36.79	39.59
	Bilinear	23.951	24.935	23.244
House2	DCT	23.586	25.438	23.282
	Wavelet	22.148	23.054	25.675
	Bilinear	27.66	29.73	27.36
Lake	DCT	27.32	30.33	27.30
	Wavelet	25.72	27.10	30.75
	Bilinear	26.869	27.129	26.085
Landsat	DCT	26.501	27.558	26.087
	Wavelet	25.478	25.898	27.635
	Bilinear	24.98	25.23	24.09
Lax	DCT	24.60	25.68	24.08
	Wavelet	23.44	23.83	25.84

Table 9.3: Continued

Image	Method	Bilinear	DCT	Wavelet
Lena	Bilinear	31.54	34.08	30.51
	DCT	31.24	34.70	30.31
	Wavelet	28.97	30.19	35.25
Linespr	Bilinear	32.709	35.143	32.53
	DCT	32.401	35.753	32.403
	Wavelet	30.692	32.159	36.316
Man	Bilinear	29.57	31.02	28.71
	DCT	29.23	31.54	28.60
	Wavelet	27.45	28.38	31.89
Meter	Bilinear	36.23	39.81	35.94
	DCT	35.96	40.57	35.71
	Wavelet	33.87	35.58	41.06
Moon	Bilinear	30.897	31.49	30.539
	DCT	30.533	31.95	30.51
	Wavelet	29.529	30.159	32.265
Pentagon	Bilinear	30.01	31.797	29.035
	DCT	29.682	32.344	28.889
	Wavelet	27.644	28.652	32.871
Peppers	Bilinear	31.242	33.213	31.008
	DCT	30.895	33.827	31.013
	Wavelet	29.425	30.847	34.139
Plane	Bilinear	29.67	32.21	29.33
	DCT	29.31	32.94	29.08
	Wavelet	27.71	29.05	32.16
Plants	Bilinear	29.85	30.61	27.94
	DCT	29.46	30.97	27.88
	Wavelet	27.33	27.82	31.15

Table 9.3: Continued

Image	Method	Bilinear	DCT	Wavelet
Res_Chart	Bilinear	19.56	21.11	18.78
	DCT	19.34	21.37	18.74
	Wavelet	17.54	18.58	21.60
Ruler	Bilinear	12.866	14.78	14.093
	DCT	12.669	15.087	14.167
	Wavelet	12.608	14.922	14.308
Splash	Bilinear	33.18	35.72	33.37
	DCT	32.80	36.50	33.50
	Wavelet	31.47	33.38	36.41
Sthelens	Bilinear	25.501	27.156	24.672
	DCT	25.165	27.702	24.547
	Wavelet	23.312	24.33	28.127
Straw	Bilinear	17.151	18.554	15.198
	DCT	16.831	19.081	15.058
	Wavelet	14.193	14.884	19.528
Swimmer	Bilinear	34.852	35.274	33.406
	DCT	34.535	35.461	33.478
	Wavelet	32.683	33.233	35.963
Tajmahal	Bilinear	24.793	24.922	23.924
	DCT	24.411	25.353	23.96
	Wavelet	23.369	23.733	25.541
Tank	Bilinear	32.124	32.814	31.171
	DCT	31.757	33.29	31.122
	Wavelet	30.279	30.892	33.5
Test8g	Bilinear	15.984	15.921	15.172
	DCT	15.655	16.246	15.183
	Wavelet	14.869	15.037	16.218

Table 9.3: Continued

Image	Method	Bilinear	DCT	Wavelet
Test8r	Bilinear	15.935	15.901	15.182
	DCT	15.614	16.216	15.186
	Wavelet	14.837	15.019	16.213
Tiffany	Bilinear	31.678	33.258	31.128
	DCT	31.296	33.933	30.89
	Wavelet	30.14	31.002	32.328
Warplane	Bilinear	34.335	35.926	33.069
	DCT	34.015	36.438	32.9
	Wavelet	31.889	32.759	36.814
Wgrain	Bilinear	18.029	18.82	16.952
	DCT	17.637	19.358	16.919
	Wavelet	15.977	16.641	19.6
Woman	Bilinear	37.368	41.262	37.217
	DCT	37.11	41.976	37.029
	Wavelet	35.065	36.93	42.465

9.7 Table of simulation results

Table 9.4: Match count comparison for the test images. The columns represent the downsampling method. The first number in each column is the count of matches for the bilinear/DCT pattern collection. The second number is the count for the wavelet collection. Correct "estimation" is made for DCT and bilinear cases if the first number is larger, and vice versa for the wavelet case.

	DCT	Wavelet	Bilinear
Aerial	7/2	6/7	9/6
Aerial2	148/71	93/116	134/89
Apc	28/11	23/29	43/14
Aqua	8/0	2/18	20/0
Baboon	0/4	4/5	4/5
Balloon	150/10	14/153	162/13
Barbara	221/153	232/135	206/166
Beach	45/31	18/32	53/34
Bridge	171/105	147/161	187/115
Bwall	390/207	281/427	459/287
Camera	18/20	9/19	24/16
Coral	3/2	2/4	3/2
Crowd	38/14	45/16	48/17
Dna	1169/493	518/1228	1215/501
Elaine	37/20	28/70	94/18
Fence	257/228	191/246	237/169
Finger	55/0	13/42	61/2
Front	272/157	242/324	291/173
Goldhill	102/71	119/77	108/81
Gray21	1695/162	0/2636	1963/0

Table 9.4: Continued

	DCT	Wavelet	Bilinear
House2	139/83	110/137	138/45
Lake	116/107	137/121	121/125
Landsat	6/5	2/6	8/1
Lax	15/12	16/17	18/6
Lena	99/13	72/15	86/28
Linespr	113/87	115/146	140/97
Man	28/22	29/37	41/26
Meter	504/237	329/297	430/325
Moon	0/0	0/0	0/0
Pentagon	6/3	3/3	8/2
Peppers	67/41	47/51	76/44
Plane	97/43	81/105	105/47
Plants	469/163	167/390	462/115
Res_chart	321/278	379/404	399/265
Ruler	0/1	0/0	8/0
Splash	325/251	394/218	314/268
Sthelens	7/2	4/7	6/1
Straw	1/0	0/0	1/1
Swimmer	36/8	30/4	39/7
Tajmahal	19/8	2/9	34/1
Tank	9/17	27/3	13/5
Test8g	12/8	6/35	48/1
Test8r	12/0	3/24	31/1
Tiffany	4/2	4/6	6/4
Warplane	5/3	3/7	8/4
Wgrain	82/81	65/45	79/67
Woman2	22/2	12/11	13/13

Chapter 10

Conclusion

In this dissertation, several different image coding scenarios were presented. These covered a variety of image sources and applications. The simulation results indicated that our proposed methods can achieve better performance than just “off-the-shelf” traditional image compression techniques.

A state of the art image coder must be flexible enough to be able to handle many different types of images and uses. As shown in this work, the added flexibility that allows improvement for a particular constraint often leads to some performance degradation for traditional compression. Balancing these two criteria will remain a challenge in still image coding. Unlike the original JPEG standard, JPEG 2000 is designed to offer a variety of options for flexible coding [80, 64]. Part 1 of the standard aimed at providing a minimal coder of good performance on most common images with reasonable complexity. Many of the topics covered in this thesis were added to the standard as later extensions for increased functionality, among which were expended RoI coding in Part 2 and compound image coding in Part 6. A possible future extension will deal with the coding of 3D images in Part 10. Not all implementations of the standard will contain these extensions, but they provide a common interface instead of having many competing formats, to avoid the chaotic situation that exists with the many video file formats on the Internet today.

Many of the problems in still image coding and video compression are well studied. One area for significant future research possibilities is stereoscopic video coding. Unlike traditional video coding, stereoscopic video compression lacks many significant current applications. Potential applications could include remote sensing and surveillance, remotely operated vehicles, stereo movies, and games. Some of the proposed techniques in the literature try to fit stereo video coding into existing standards [76, 98, 62, 63], namely the temporal scalability profile of MPEG-2 or MPEG-4, while others introduce a non-standard compatible approach [75, 59]. MPEG-4 contains provisions for object-based coding which can be extended to stereoscopic and 3D sequences as well [77].

In stereo sequences, redundancy exists between temporally and spatially adjacent frames. Efficient algorithms are needed to perform (possibly joint) motion and disparity compensation in real time compression scenarios.

While the coding of the residual images of block-based disparity estimation is well researched (we showed a possible technique in Chapter 7), that of the residual images resulting from object-based disparity estimation is more or less an open problem.

An extension of the compression of stereoscopic sequences is multi-view image sequence coding [28, 75]. In this case, the sequences are taken from multiple camera locations. Their compression is very similar to that of stereo sequences. These sequences can be used for the generation of intermediate views by interpolating the views captured by the cameras. For real time applications fast algorithms need to be developed with accurate approximation qualities. One application of multi-view image coding is for sensor networks.

In Chapter 9 we showed a technique that did not rely on a reference image to determine certain properties of a given image. Having a reference image is helpful when comparing the performance of an image manipulation technique. Unfortunately that is not always possible to obtain. No-reference or blind quality assessment methods have been proposed in [88, 49] for JPEG compressed images. A more comprehensive quality assessment approach is taken in [44]. The extension of these works to different coding mechanisms and video coding is an interesting area for future work.

Bibliography

[1] J. P. Allebach and P. W. Wong. Edge-directed interpolation. In *IEEE International Conference on Image Processing*, volume 3, pages 707–710, September 1996.

[2] P. An, Z. Zhang, and L. Shi. Theory and experiment analysis of disparity for stereoscopic image pair. In *International Symposium on Intelligent Multimedia, Video and Speech Processing*, volume 1, pages 68–71, May 2001.

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

[4] C. B. Atkins, C. A. Bouman, and J. P. Allebach. Tree-based resolution synthesis. In *Conference on Image Processing, Image Quality, Image Capture Systems*, pages 405–410, April 1999.

[5] C. B. Atkins, C. A. Bouman, and J. P. Allebach. Optimal image scaling using pixel classification. In *IEEE International Conference on Image Processing*, volume 3, pages 864–867, September 2001.

[6] E. Atsumi and N. Farvardin. Lossy/lossless region-of-interest image coding based on set partitioning in hierarchical trees. In *IEEE International Conference on Image Processing*, volume 1, pages 87–91, October 1998.

[7] H. Aydinoğlu and M.H. Hayes III. Stereo image coding. In *International Symposium on Circuits and Systems*, volume 1, pages 247–250, April 1995.

[8] H. Aydinoğlu and M.H. Hayes III. Stereo image coding: A projection approach. *IEEE Transactions on Image Processing*, 7(4):506–516, April 1998.

[9] H. Aydinoğlu, F. Kossentini, Q. Jiang, and M.H. Hayes III. Region-based stereo image coding. In *IEEE International Conference on Image Processing*, volume 2, pages 57–60, October 1995.

[10] L. Bottou, P. G. Howard, and Y. Bengio. The Z-coder adaptive binary coder. In *Proceedings of the Data Compression Conference*, pages 13–22, March 1998.

[11] L. Bottou and S. Pigeon. Lossy compression of partially masked still images. In *Proceedings of the Data Compression Conference*, page 528, March 1998.

[12] N. V. Boulgouris and M. G. Strintzis. Embedded coding of stereo images. In *IEEE International Conference on Image Processing*, volume 3, pages 640–643, September 2000.

[13] N. V. Boulgouris and M. G. Strintzis. A family of wavelet-based stereo image coders. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(10):898–903, October 2002.

[14] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, CA, 1984.

[15] S. G. Chang, Z. Cvetković, and M. Vetterli. Resolution enhancement of images using wavelet transform extrema extrapolation. In *International Conference on Acoustics, Speech and Signal Processing*, pages 2379–2383, May 1995.

[16] H. Cheng and C. A. Bouman. Multilayer document compression algorithm. In *IEEE International Conference on Image Processing*, volume 1, pages 244–248, October 1999.

[17] C. Chrysafis and A. Ortega. Efficient context-based entropy coding for lossy wavelet image compression. In *Proceedings of the Data Compression Conference*, pages 241–250, March 1997.

[18] R. R. Coifman and M. V. Wicherhauser. Entropy-based algorithms for best basis selection. *IEEE Transactions on Information Theory*, 38(2):713–718, March 1992.

[19] P. Cosman and K. Zeger. Memory constrained wavelet-based image coding. In *Proceedings of the First Annual UCSD Conference on Wireless Communications*, pages 54–60, March 1998.

[20] P. C. Cosman, R. M. Gray, and M. Vetterli. Vector quantization of image subbands: A survey. *IEEE Transactions on Image Processing*, 5(2):202–225, February 1996.

[21] P. C. Cosman and K. Zeger. Memory constrained wavelet-based image coding. *IEEE Signal Processing Letters*, 5(9):221–223, September 1998.

[22] R. de Queiroz. Compression of compound documents. In *IEEE International Conference on Image Processing*, volume 1, pages 209–213, October 1999.

[23] R. de Queiroz. On data filling algorithms for MRC layers. In *IEEE International Conference on Image Processing*, volume 2, pages 586–589, September 2000.

[24] I. Dinstein, G. Guy, J. Rabany, J. Tzelgov, and A. Henik. On stereo image coding. In *International Conference on Pattern Recognition*, volume 1, pages 357–359, November 1988.

[25] R. Dugad and N. Ahuja. A fast scheme for image size change in the compressed domain. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(4):461–474, April 2001.

[26] T. Frajka and K. Zeger. Residual image coding for stereo image compression. *Optical Engineering*, 42(1):182–188, January 2003.

[27] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley, 1992.

[28] N. Grammalidis and M. G. Strintzis. Disparity and occlusion estimation in multiocular systems and their coding for the communication of multiview image sequences. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(3):328–344, June 1998.

[29] P. Haffner, L. Bottou, P. G. Howard, P. Simard, Y. Bengio, and Y. LeCun. High quality document image compression with ”DjVu”. *Journal of Electronic Imaging*, 7(3):410–425, July 1998.

[30] H. S. Hou and H. C. Andrews. Cubic splines for image interpolation and digital filtering. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(6):508–517, December 1978.

[31] P. G. Howard and J. S. Vitter. Analysis of arithmetic coding for data compression. In *Proceedings of the Data Compression Conference*, volume 1, pages 3–12, April 1991.

[32] D. Huttelocher, P. Felzeszwalb, and W. Rucklidge. Digipaper: A versatile color document image representation. In *IEEE International Conference on Image Processing*, volume 1, pages 219–223, October 1999.

[33] ITU-T Recommendations T.44, Study Group-8 Contribution. *Mixed Raster Content (MRC)*, 1998.

[34] K. Jensen and D. Anastassiou. Subpixel edge localization and the interpolation of still images. *IEEE Transactions on Image Processing*, 4(3):285–295, March 1995.

[35] H.-H. Jeon, J.-H. Kim, and D.-G. Oh. Stereo image coding with disparity compensation using dynamic programming. In *International Symposium on Consumer Electronics*, pages 214–217, December 1997.

[36] Q. Jiang, J. J. Lee, and M. H. Hayes III. A wavelet based stereo image coding algorithm. In *International Conference on Acoustics, Speech and Signal Processing*, volume 6, pages 3157–3160, March 1999.

[37] S.-H. Jung, S. K. Mitra, and D. Mukherjee. Subband DCT: Definition, analysis, and applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3):273–286, June 1996.

[38] R. G. Keys. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(6):1153–1160, December 1981.

[39] W.-H. Kim, J.-Y. Ahn, and S.-W. Ra. An efficient disparity estimation algorithm for stereoscopic image compression. *IEEE Transactions on Consumer Electronics*, 43(2):165–172, May 1997.

[40] W.-H. Kim and S.-W. Ra. Fast disparity estimation using geometric properties and selective sample decimation for stereoscopic image coding. *IEEE Transactions on Consumer Electronics*, 45(1):203–209, February 1999.

[41] C. W. Kok and T. Q. Nguyen. Document image compression by subband system. In *International Symposium on Circuits and Systems*, volume 2, pages 688–691, May 1996.

[42] K. Konstantinides and D. Tretter. A method for variable quantization in JPEG for imroved text quality in compound documents. In *IEEE International Conference on Image Processing*, volume 2, pages 565–568, October 1998.

[43] T. Lan and A. H. Tewfik. Multigrid embedding (MGE) image coding. In *IEEE International Conference on Image Processing*, volume 3, pages 369–373, October 1999.

[44] X. Li. Blind image quality assessment. In *IEEE International Conference on Image Processing*, volume 1, pages 449–452, September 2002.

[45] C.-W Lin, E.-Y. Fei, and Y.-C. Chen. Hierarchical disparity estimation using spatial correlation. *IEEE Transactions on Consumer Electronics*, 44(3):630–637, August 1998.

[46] J. Liu and P. Moulin. Information-theoretic analysis of interscale and intrascale dependencies between image wavelet coefficients. *IEEE Transactions on Image Processing*, 10(11):1647–1658, November 2001.

[47] S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):127–135, March 1982.

[48] J. Luo, C. W. Chen, and K. J. Parker. Face location in wavelet-based video compression for high perceptual quality videoconferencing. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(4):411–414, August 1996.

[49] J. Luo, Q. Yu, and M. E. Miller. A triage method for determining the extent of JPEG compression artifacts. In *IEEE International Conference on Image Processing*, volume 1, pages 473–476, September 2002.

[50] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek. An overview of jpeg-2000. In *Proceedings of the Data Compression Conference*, pages 523–544, March 2000.

[51] A. C. Miguel and E. A. Riskin. Protection of region of interest against data loss in a generalized multiple description framework. In *Proceedings of the Data Compression Conference*, page 562, March 2000.

[52] M. S. Moellenhoff and M. W. Maier. Characteristics of disparity-compensated stereo image pair residuals. *Signal Processing: Image Communication*, 14(1–2):55–69, November 1998.

[53] M. S. Moellenhoff and M. W. Maier. Transform coding of stereo image residuals. *IEEE Transactions on Image Processing*, 7(6):804–812, June 1998.

[54] A. E. Mohr, E. A. Riskin, and R. E. Ladner. Graceful degradation over packet erasure channels through forward error correction. In *Proceedings of the Data Compression Conference*, pages 92–101, March 1999.

[55] J. Mukherjee and S. K. Mitra. Image resizing in the compressed domain using subband DCT. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(7):620–627, July 2002.

[56] D. Nister and C. Christopoulos. Lossless region of interest coding. *Signal Processing*, 78(1):1–17, October 1999.

[57] T. Palfner, A. Mali, and E. Müller. Progressive coding of stereo images using wavelets and overlapping blocks. In *IEEE International Conference on Image Processing*, volume 2, pages 213–216, September 2002.

[58] J. A. Parker, R. V. Kenyon, and D. E. Troxel. Comparison of interpolating methods for image resampling. *IEEE Transactions on Medical Imaging*, 2(1):31–39, March 1983.

[59] S.-C. Pei and C.-L. Lai. Very low bit-rate coding algorithm for stereo video with spatiotemporal HVS model and binary correlation disparity estimator. *IEEE Journal on Selected Areas in Communications*, 16(1):98–107, January 1998.

[60] M. G. Perkins. Data compression of stereopairs. *IEEE Transactions on Communications*, 40(4):684–696, April 1992.

[61] M. Pettersson and S.Tjärnlund. Color image compression for color printers using wavelet transform. Master's thesis, Linköping University, Dept. of Electrical Engineering, S-581 83 Linköping, Sweden, November 1997.

[62] A. Puri, R. V. Kollarits, and B. G. Haskell. Stereoscopic video compression using temporal scalability. In *Proceedings of the SPIE*, volume 2501, pages 745–756, May 1995.

[63] A. Puri, R. V. Kollarits, and B. G. Haskell. Basics of stereoscopic video, new compression result with MPEG-2 and a proposal for MPEG-4. *Signal Processing: Image Communication*, 10(1–3):201–234, July 1997.

[64] M. Rabbani and R. Joshi. An overview of the JPEG 2000 still image compression standard. *Signal Processing: Image Communication*, 17(1):3–48, January 2002.

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

[66] W. D. Reynolds and R. V. Kenyon. The wavelet transform and the suppression theory of binocular vision for stereo image compression. In *IEEE International Conference on Image Processing*, volume 2, pages 557–560, September 1996.

[67] J. K. Rogers and P. C. Cosman. Robust wavelet zerotree image compression with fixed-length packetization. In *Proceedings of the Data Compression Conference*, pages 418–427, March 1998.

[68] A. Said and W. A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3):243–250, June 1996.

[69] V. E. Seferidis and D. V. Papadimitriou. Improved disparity estimation in stereoscopic television. *Electronics Letters*, 29(9):782–783, April 1993.

[70] S.-H. Seo and M. R. Azimi-Sadjadi. A 2-D filtering scheme for stereo image compression using sequential orthogonal subspace updating. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(1):52–66, January 2001.

[71] S. Sethuraman, M. W. Siegel, and A. G. Jordan. A multiresolution region based segmentation scheme for stereoscopic image compression. In *Proceedings of the SPIE*, volume 2419, pages 265–273, February 1995.

[72] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3462, December 1993.

[73] J. M. Shapiro. Smart compression using the embedded zerotree wavelet (EZW) algorithm. In *Proc. of the 27th Asilomar Conf. on Signals, Systems and Computers*, volume 1, pages 486–490, November 1993.

[74] P. G. Sherwood and K. Zeger. Macroscopic multistage image compression for robust transmission over noisy channels. In *Proceedings of the SPIE*, volume 3653, pages 73–83, January 1999.

[75] M. W. Siegel, S. Sethuraman, J. S. McVeigh, and A. G. Jordan. Compression and interpolation of 3D-stereoscopic and multi-view video. In *Proceedings of the SPIE*, volume 3012, pages 227–238, February 1997.

[76] Y.-J. Song. Improved disparity estimation algorithm with MPEG-2's scalability for stereoscopic sequences. *IEEE Transactions on Consumer Electronics*, 42(3):306–311, August 1996.

[77] M.G. Strintzis and S. Malassiotis. Object-based coding of stereoscopic and 3D image sequences. *IEEE Signal Processing Magazine*, 16(3):14–28, May 1999.

[78] J. Ström and P. C. Cosman. Medical image compression with lossless regions of interest. *Signal Processing*, 59(2):155–171, June 1997.

[79] D. Taubman. EBCOT: Embedded block coding with optimized truncation. *ISO/IEC JTC 1/SC 29/WG 1, Document N1020R*, October 1998.

[80] D. S. Taubman and M. W. Marcellin. JPEG2000: Standard for interactive imaging. *Proceedings of the IEEE*, 90(8):1336–1357, August 2002.

[81] C. Teng and D. L. Neuhoff. New quadtree predictive image coder. In *IEEE International Conference on Image Processing*, volume 2, pages 73–76, October 1995.

[82] C. Teng and D. L. Neuhoff. Quadtree-guided wavelet image coding. In *Proceedings of the Data Compression Conference*, pages 406–415, March 1996.

[83] T. D. Tran and T. Q. Nguyen. A progressive transmission image coder using linear phase uniform filterbanks as block transforms. *IEEE Transactions on Image Processing*, 8(11):1493–1507, November 1999.

[84] D. Tzovaras and M. G. Strintzis. Disparity estimation using rate-distortion theory for stereo image sequence coding. In *International Conference on Digital Signal Processing*, volume 1, pages 413–416, July 1997.

[85] D. Tzovaras and M. G. Strintzis. Motion and disparity field estimation using rate-distortion optimization. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(2):171–180, April 1998.

[86] M. Unser, A. Aldroubi, and M. Eden. Enlargement or reduction of digital images with minimum loss of information. *IEEE Transactions on Image Processing*, 4(3):247–258, March 1995.

[87] M. Vetterli and C. Herley. Wavelets and filter banks: Theory and design. *IEEE Transactions on Signal Processing*, 40(9):2207–2232, September 1992.

[88] Z. Wang, H. R. Sheikh, and A. C. Bovik. No-reference perceptual quality assessment of JPEG compressed images. In *IEEE International Conference on Image Processing*, pages 477–480, September 2002.

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

[90] W. Woo and A. Ortega. Optimal blockwise dependent quantization for stereo image coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(6):861–867, September 1999.

[91] W. Woo and A. Ortega. Overlapped block disparity compensation with adaptive windows for stereo image coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(2):194–200, March 2000.

[92] Z. Xiong, O. G. Guleryuz, and M. T. Orchard. A DCT-based embedded image coder. *IEEE Signal Processing Letters*, 3(11):289–290, November 1996.

[93] H. Yamaguchi, Y. Tatehira, K. Akiyama, and Y. Kobayashi. Stereoscopic images disparity for predictive coding. In *International Conference on Acoustics, Speech and Signal Processing*, volume 3, pages 1976–1979, May 1989.

[94] S. Yang and T. Q. Nguyen. Interpolated Mth band filters for image size conversion. In *IEEE International Conference on Image Processing*, volume 3, pages 891–894, October 2001.

[95] W. Zeng, J. Li, and S. Lei. Adaptive wavelet transforms with spatially varying filters for scalable image coding. In *IEEE International Conference on Image Processing*, volume 1, pages 112–116, October 1998.

[96] Y. Zhang and G. Li. An efficient hierarchical disparity estimation algorithm for stereoscopic video coding. In *IEEE Asia-Pacific Conference on Circuits and Systems*, volume 1, pages 744–747, December 2000.

[97] W. Zheng, Y. Shishikui, Y. Tanaka, and I. Yuyama. Disparity estimation with hierarchical block correlation for stereoscopic image coding. *Systems and Computers in Japan*, 28(6):1–9, June 1997.

[98] M. Ziegler, L. Falkenhagen, R. ter Horst, and D. Kalivas. Evolution of stereoscopic and three-dimensional video. *Signal Processing: Image Communication*, 14(1–2):173–194, November 1998.