SPLINE WAVELET IMAGE CODING AND SYNTHESIS FOR A VLSI BASED DIFFERENCE ENGINE

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE,
FACULTY OF SCIENCE
AT THE UNIVERSITY OF CAPE TOWN
IN FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Patrick Craig Marais
June 1994

Supervised by
Professor E.H. Blake

Abstract

The efficiency of an image compression/synthesis system based on a spline *multi-resolution analysis* (MRA) is investigated. The proposed system uses a quadratic spline wavelet transform combined with minimum-mean squared error vector quantization to achieve image compression. Image synthesis is accomplished by utilizing the properties of the MRA and the architecture of a custom designed display processor, the Difference Engine. The latter is ideally suited to rendering images with polynomial intensity profiles, such as those generated by the proposed spline MRA. Based on these properties, an *adaptive* image synthesis system is developed which enables one to reduce the number of instruction cycles required to reproduce images compressed using the quadratic spline wavelet transform. This adaptive approach is computationally simple and fairly robust. In addition, there is little overhead involved in its implementation.

Contents

[1]Joint Photographic Experts Group

Chapter 1

Introduction

1.1 Image Compression and Synthesis

Image data are playing an increasingly important role in our society, primarily as a result of the emergence of multi-media applications and services. The evolution of a world-spanning network, and the ever-expanding expectations of its users with regard to multi-media integration, demand that mechanisms be found to transmit a rapidly increasing volume of image data more efficiently.

To achieve data reduction, one uses either a ***lossless*** or ***lossy*** compression scheme — the former enables one to reconstruct the input data precisely, whereas the latter approach sacrifices perfect reconstruction in favour of far higher compression ratios. The gains that such schemes can produce are dependent on the requirements imposed by the type of data: text data must be compressed losslessly, and is thus only amenable to low compression ratios (of the order 2:1). On the other hand, for video (or sound) data, one can employ lossy compression techniques since the human visual (audio) system is insensitive to the loss of certain kinds of information. At the most primitive level, such techniques exploit the correlations between neighbouring samples: much of the information used to represent an image is redundant, and can thus be neglected in the encoding. The gains can be be enormous — around 30:1, with good reproduction, using techniques such as wavelet or fractal compression. For video streams the gains are even more astounding, since one now has a large measure of temporal redundancy — images do not change much from frame to

frame, and this redundancy can be exploited. The current standards for image compression are JPEG (for still images) and MPEG (for compression of video streams) — Section 4.1.1.

Real-time applications, such as teleconferencing, require that both the encoding and synthesis of the (image/sound) data be accomplished quickly. To accommodate these specifications, such compression/synthesis systems are normally implemented in hardware. However, even hardware cannot always produce data at an adequate rate to ensure a smoothly changing video sequence. Thus, any mechanism which could accelerate this final reconstruction phase would be extremely useful. The efficiency one can achieve when reconstructing compressed (image) data is of great importance. In the context of this dissertation, ***synthesis efficiency*** is quantified by the number of display processor cycles required to render the image. It should be noted that the attainable efficiency will depend on the amount of detail one wishes to maintain in the reconstructed image. For example, if only a crude image approximation is required for, say, identification purposes, then much of the detail may be discarded and rendering time will be accordingly reduced. In addition, this 'efficiency' measure does not take account of the time needed to compute the information required by the display processor — it only considers the time required to render the image. However, since the required information (inverse transform data) can be computed quickly using optimized convolution hardware, this is not a serious objection.

'Standard' image synthesis hardware computes each pixel value explicitly before generating instructions to illuminate the corresponding screen locations. The complete system proposed in this dissertation consists of two phases

- wavelet compression (at the encoder)
- adaptive image synthesis (at the decoder)

The former ensures that the amount data required to represent an image is significantly smaller than that present in the input image, while the latter provides an efficient means to reconstruct the transmitted image. To facilitate image synthesis on the *Difference Engine* (See Section 1.3.1), the input image is taken to lie in the space of polynomial spline approximations. In fact, the framework within which the synthesis algorithms are developed — ***spline multi-resolution analysis*** — lends itself to the implementation of the compression phase as well, since the analysis provides the wavelet representation upon which the compression algorithms operate.

1.2 Wavelet Image Coding and Multi-Resolution Analysis

Multi-resolution analysis (MRA) provides a means of investigating an image with differing degrees of precision. A low resolution image is one which has a slowly varying intensity profile, while a high resolution image exhibits considerably more variation in its intensity values. The analysis decomposes an image by subjecting it to a variety of filtering operations; each of the resulting sub-bands contains information about the image at the specified resolution. There are two kinds of images in such a decomposition: approximation and difference images. The former are low-pass filtered versions of the input, and a consequently less 'detailed'. The difference images are obtained by simply differencing consecutive approximation images. The complete representation consists of the lowest resolution approximation and the sequence of difference images which may be added to this image to reconstruct the input.

Such an analysis can tell one important things about the image, such as the location and strength of dominant edges or the orientation of such edges.

The approximation and detail spaces (in which these images reside) are spanned by 'scaling functions' and 'wavelets', respectively. These kernel functions are scaled and translated to produce resolution-dependent bases. The basis coefficients describing the MRA may be used as an alternative representation — they are obtained through the *wavelet transform*. The wavelet transform produces a sparse representation, in that many of the basis coefficients are zero or close to zero. Consequently, such a transformation may be used for data compression.

1.3 Adaptive Multi-Resolution Synthesis on the Difference Engine

Given a compressed representation, one must rebuild the input image by means of an inverse transform. In terms of the representation above, this means weighting the relevant bases by the stored coefficients and summing the resulting functions.

1.3.1 The Difference Engine

The 'Difference Engine', developed at the CWI in Amsterdam, is a display processor which allows efficient rendering of images expressed in terms of polynomial primitives. Intensity data which lies on a polynomial can be rendered in a small number of processor cycles, by employing forward difference calculations; the time taken to perform the calculations is proportional to the degree of the polynomial to be interpolated but is independent of the length of the span. In other words, it costs the same to interpolate a thousand pixel span as it does to interpolate a ten pixel span. Since the processor clock runs at 90MHz and the difference calculations are cheap, the Difference Engine can produce pixel values at the line refresh rate. The Difference Engine has an accumulator to which successive (positive or negative) pixel values can be added before the final value is rendered. This feature is of vital importance in the proposed multi-resolution synthesis scheme.

It should be noted that, while wavelet decomposition is amenable to progressive transmission, the hardware is unable to buffer scan-line information from previously rendered levels and so cannot perform this function at present.

1.3.2 Image Synthesis on the Difference Engine

The spline MRA provides an image decomposition which can be easily synthesized on the Difference Engine (scan-line by scan-line). However, such a direct multi-resolution synthesis is not efficient, since each pixel will have several levels of accumulated detail, each of which requires additional processor cycles to produce. The aim of the synthesis algorithms is to reduce the number of cycles required to render the image below that required for direct image rendering i.e., setting each pixel explicitly.

To achieve this, one uses *adaptive* reconstruction, that is, rather than adding back each pixel in the detail images, only visually relevant information is used. The method presented in this dissertation uses a two pronged approach:

- The wavelet detail coefficients are used to provide an indication of visual relevance. Large coefficients are retained (typically those preserved by the compression phase) and only the wavelet basis elements which these weight are used in the reconstruction.

- A spline merging algorithm merges redundant splines (neighbouring splines with the same or similar differences) into larger non-redundant spans. Since the time required to render a polynomial span is independent of the span length, this is desirable since it is cheaper to render one span than to render several.

Using these two methods significant gains can be achieved (as measured by the reduction in rendering time). This subject is treated fully in Chapter 5.

1.4 Second Generation Image Coding

The wavelet transform algorithms used to compress the image use the fact that high frequency noise (quantization error) is less visually disturbing than low frequency noise. However, edge information, which plays a very important role in perception, is not used to any benefit. This edge data can be used to provide a compact image representation ([7], [29]), communicating much of the visually relevant content to the viewer. Unfortunately, such a representation does not represent texture information well. This problem can be remedied by subjecting the residue (the difference between the input and the edge reconstruction) to wavelet coding. Since the strong edges have been removed, the wavelet representation will be amenable to higher compression and, providing the edge representation is compact, one should experience compression gains over direct wavelet coding (See Chapter 6).

1.5 Outline of Dissertation

The dissertation is set out as follows:

Chapter 2 The notation used in this dissertation is summarized in Section 2.1. The theory of multi-resolution and wavelet analysis is then introduced. This work lays the foundation for the more specialised theory that follows.

Chapter 3 The semi-orthogonal spline MRA is introduced and the framework established for subsequent difference engine manipulations. The unexpected advantages of quadratic over cubic splines are presented and analysed. Techniques are developed to overcome the difficulties which usually hamper quadratic spline implementations. A complete

set of boundary conditions are derived which permit perfect reconstruction and do not require that the image have a particular origin or size i.e., one does not have to pad the image to a power of two or have the origin on the zero vector.

Chapter 4 Wavelet compression (using vector quantization) is investigated. An analysis of the compression results is undertaken and some suggestions are made as to how performance might be improved, through the use of adaptive vector quantization with a distortion measure better suited to the Human Visual System than the Mean-Square Error (MSE).

Chapter 5 Efficient image synthesis on the difference engine is investigated. New algorithms are developed which enable the DE to render images more efficiently than would be possible by direct synthesis alone. These algorithms are based on the 'truncated' MRA generated by wavelet compression and also use the C^1 continuity enforced by the choice of a quadratic MRA to reduce the number of instructions required to render each scan-line. The results obtained indicate that images with a high degree of smoothness are particularly suited to synthesis on the Difference Engine.

Chapter 6 Edge extraction and edge coding are introduced in this chapter. Some preliminary results on edge extraction are presented.

Chapter 7 The results of the dissertation are summarized in this chapter. Chief amongst these is the effectiveness of the Difference Engine as a means of accelerating the reconstruction of spline wavelet compressed images. Suggestions for future extensions are discussed, with the intention of sparking research into the development of a more specialized spline rendering engine and more effective compression strategies.

Each chapter is concluded with a summary of the most salient observations and results.

Chapter 2

Wavelet Theory

The *Wavelet Transform (WT)* has attracted a great deal of interest from a wide variety of disciplines, essentially because it is such a versatile tool for analysis — of both data and more abstract structures. In general, the WT produces data which is *sparse* in the transform domain in the sense that many of the transform coefficients are (relatively) small; this is desirable since the smallest of these coefficients may be approximated by zero and ignored in subsequent computations. However, this property alone could scarcely justify the interest surrounding the WT and indeed, this transform has some very definite advantages over other transform methods.

The Fourier Transform (and its derivatives) only provide information about global spectral characteristics — a consequence of the sinusoidal transform kernel upon which they are based. Since one usually desires information about local signal features, e.g., the location of a spike in a time varying signal, these transforms are of limited use in signal analysis. However, if the sinusoid is multiplied by a *windowing function*, i.e., a function which has localised extent or dies away very rapidly, then the transform provides a frequency analysis of this windowed piece of the signal, and one can then deduce the nature of local phenomena by examining the transform coefficients. Such a transform is known as a *Short-Time Fourier Transform* or STFT. The entire signal can be decomposed and analyzed in this fashion, the accuracy of our analysis depending on the size of the windowing function and the constraints imposed on the *time-frequency window* by the Uncertainty Principle: when we analyze the frequency spectrum of a signal with great precision, we loose the ability to accurately determine the location of corresponding phenomena (a spike in the signal, perhaps) in the

time domain (and vice-versa). As the range of time/space over which we wish to consider the signal grows, the corresponding increase in non-local spectral information muddies our analysis and we can say progressively less about local signal features. The problems of localised signal analysis with the STFT are further exacerbated by an additional constraints on the time-frequency window — it must be fixed prior to the analysis, which implies that one must either have knowledge of the kinds of signals to be investigated (enabling one to choose a near optimal window for signals in this class) or risk producing spurious results, since the STFT will not yield accurate information for frequencies outside the frequency-band the window was designed to analyze.

The Wavelet Transform, on the other hand, is well suited to the analysis of signals with arbitrary spectra, being constructed in such a way that its time-frequency window adapts to the local characteristics of the signal, thus permitting the study of signals without *a priori* determination of the window. Since images generally have widely varying intensity characteristics, it is clear why the WT has found acceptance amongst those engaged in the analysis of such signals.

In addition, the *multi-resolution* structure of the WT it well suited to image analysis and compression (see, for example, [29, 13, 15, 17, 37]). The concept of a *Multi-Resolution Analysis (MRA)* is central to much of wavelet theory — Section 2.5 discusses this subject at length.

The following sections provide a simple introduction to the theory underpinning this dissertation. The primary aim of this chapter is to introduce the essentials of wavelet theory to the non-specialist. With this in mind, Section 2.1 provides a brief explanation of the symbols and notation employed throughout the rest of this work. Sections 2.3 and 2.4 discuss the Wavelet Transform — the former deals with the continuous version, and the latter introduces the discrete form. Multi-resolution analysis (in both the 1-D and 2-D settings) is covered in Section 2.5.

Those who are familiar with the issues and concepts involved may omit this chapter. For the most part, I have referred only to aspects of the theory which I have actually used; those desiring a more extensive exposition are referred to [10, 13].

2.1 Mathematical Preliminaries

In the course of reading this dissertation, the reader may encounter unfamiliar mathematical symbols and concepts. This section introduces the material required to understand the theory developed later.

It is assumed that the reader is familiar with basic linear algebra, set theory and the concept of a linear or vector space.

The following (standard) symbols for number systems are used:

$\mathbb{C}$ Complex numbers,

$\mathbb{N}$ Natural numbers,

$\mathbb{R}$ Real numbers,

$\mathbb{Z}$ Integers.

The symbols $\|.\|$ and $\langle .,. \rangle$ are used to denote norms and inner products, respectively. The dots represent the positions of arguments. The former is a generalized length measurement and the latter a generalization (to arbitrary vector spaces) of the Euclidean dot product. For example, if two elements of a vector space have a zero inner product, they are considered 'orthogonal', where orthogonality is interpreted in the most intuitively appropriate way. When an inner product exists on a space U, one may define a norm on the same space as $\|u\|^2 = < u, u >,\ u \in U$.

The following are common vector spaces which might be encountered:

$L^2(\mathbb{R})$ The space of (measurable) square integrable functions. The concept of a *measure* is one that I shall not refer to further. For our purposes, any function which satisfies $\int |f(x)|^2 dx < \infty$ is contained in this space, where the integral is a Riemann integral. The inner product on this space is defined as $\langle u, v \rangle = \int u\bar{v}dx$, where the overbar denotes complex conjugation. Thus, our requirement states that $\|u\| < \infty$ for membership.

$\ell^2(\mathbb{R})$ The space of all square summable sequences, that is a sequence $\{a_k\}_{k\in\mathbb{Z}}$ is contained in this space if it satisfies the criterion $\sum_{k\in\mathbb{Z}} |a_k|^2 < \infty$. That is, it has a bounded norm. This space is the discrete analogue of L^2.

$C^m(\mathbb{R})$ The space of all m-times continuously differentiable functions. That is, a function belongs to this space if its first m derivatives are continuous. $C^0(\mathbb{R})$ is the space of continuous functions.

Any other spaces I require will be described when they are encountered.

The Fourier transform of an L^2 function, $f(x)$, is denoted by $\hat{f}(\omega)$ and is given by

$$\hat{f}(\omega) = \int e^{-i\omega x} f(x) dx.$$

The symbol $*$ is used to denote both continuous and discrete convolution (one can deduce which easily enough). The former is given by

$$(f * g)(x) = \int f(x-t)g(t)dt$$

and the latter by

$$(f * g)(i) = \sum_{k \in \mathbb{Z}} f_{i-k} g_k,$$

where the both the integral and sum are assumed to converge.

The idea of a *direct sum* of two spaces, U, V will be used presently. If the space W is obtained via a direct sum then $W = U \oplus V \equiv \{w : w = u + v, u \in U, v \in V\}$. There is a subtlety here, since $\oplus$ is sometimes used to denote an ***orthogonal*** direct sum, that is, one is which the summand spaces are orthogonal to each other. When this is not the case, the symbol $\dot{+}$ will be used.

The operation of *closure* adds in all the limit points of a space, thus 'closing it up'. This operation is usually denoted by clos_B, where B is the set w.r.t which the closure is taken, that is, we take our limits in the set B. For example, if we are given a finite subset (a, b) of $\mathbb{R}$, then $\mathrm{clos}_{\mathbb{R}}(\mathrm{a}, \mathrm{b}) = [\mathrm{a}, \mathrm{b}]$; the limiting points of the open interval have been included.

The notation $\mathrm{span}\{\mathrm{v_i} : \mathrm{i} \in \mathbb{Z}\}$ is used to denote the span of the (basis) vectors, v_i; that is, the space which consists of all possible linear combinations of these vectors. An alternative notation consists of a pair of angle brackets, thusly: $< v_i : i \in \mathbb{Z} >$.

The symbol $\otimes$ is used to denote a tensor product. The tensor product, $U \otimes V$, of two spaces, U and V, yields a new space, the elements of which are product combinations of vectors in the two component spaces.

The support of a function is the closure of the set of non-zero values it assumes; for a sequence I will take it to mean the set of sequence indices which have non-zero sequence values associated with them. If a function (sequence) possesses compact support, then (technicalities aside) this set is of finite extent (or has a finite number of members).

To simplify summation formulae, I will often ignore the range subscript, it usually being the case that our index ranges from $-\infty$ to $+\infty$. Similarly, if there are no range limits on an integral, one may assume it is taken over the entire domain.

2.2 Time-Frequency Analysis

When one takes the Fourier Transform of a signal, the spectral information produced provides a description of the way in which the various sinusoids which compose the signal contribute. While this is useful, the information which the Fourier Transform yields is global in nature i.e., only the global contribution of each frequency to the entire signal is available. Thus, any transient (and potentially important) occurrences, such as a sudden momentary drop in a fairly constant time-varying signal, will not be accurately reflected in the analysis performed by the FT — one might be able to infer that they occurred by examining the spectrum, but the location of this event will be highly indeterminate.

The purpose of the STFT is to provide a means of extracting local spectral data, thus permitting a meaningful analysis of the signal. This is achieved by *windowing* off part of the signal, and performing an analysis on this segment — hence the name. This windowing is achieved by a so-called *window function.*

Definition 2.1 *A function $w(t) \in L^2$ is called a* window function *if $t\,w(t) \in L^2$. This window function has a well defined centre, t^* and radius, Δ_w:*

$$t^* \equiv \frac{1}{\|w\|_{L^2}^2} \int t|w(t)|^2 dt$$

and

$$\Delta_w \equiv \frac{1}{\|w\|_{L^2}} \left\{ \int (t-t^*)^2 |w(t)|^2 dt \right\}^{\frac{1}{2}}$$

A well known example of a STFT (with a Gaussian window function) is the *Gabor Transform*, defined by

$$(G^{\alpha}f)(\omega, b) = \int e^{-i\omega t} f(t) g_{\alpha}(t-b)\, dt \tag{2.1}$$

where

$$g_{\alpha}(t) = \frac{1}{2\sqrt{\pi\alpha}} e^{-\frac{t^2}{4\alpha}}. \tag{2.2}$$

Associated with such transforms is a so-called ***time-frequency window***, which indicates the region in the time-frequency domain about which the transform yields information. For example, the Gabor window is

$$[b-\sqrt{\alpha}, b+\sqrt{\alpha}] \times [\omega - \frac{1}{2\sqrt{\alpha}}, \omega + \frac{1}{2\sqrt{\alpha}}]. \tag{2.3}$$

Now, to analyze low-frequency phenomena, one would like a window which is wide in the time domain, since frequency is inversely proportional to period. Likewise, to effectively analyze high-frequency phenomena we would like a narrow time window. Unfortunately, the time-frequency window is constrained by a minimum size requirement; as the time window narrows, the frequency window expands so as not to violate this requirement, and vice versa. That is, while we might be able to isolate a section of our signal with great precision, we are simultaneously faced with an increasing amount of frequency information which garbles the data we really want and mitigates the effect of our contracting time-window. Similarly, as the time-window widens, our frequency information becomes more accurate, but we are now faced with uncertainty as to which region of the time-window contains the interesting phenomena which led us to window the signal in the first place! This is the Uncertainty Principle; it is inviolable and places a lower limit on the accuracy with which we can investigate a signal.

Nonetheless, provided one has knowledge of the kinds of signals which one is going to analyze, a good windowing function may be chosen. In particular, the Gaussian window of the Gabor Transform provides the optimal time-frequency trade-off for localised signal analysis. If the signal has widely varying spectral characteristics, however, any analysis provides by a STFT is going to be less than satisfactory: the window, once chosen, is static and only suitable for analyzing a particular type of signal. What we desire is a time-frequency window which adapts to the underlying characteristics of the signal, becoming

wider in the time domain when analyzing low-frequency phenomena and contracting to study high-frequency phenomena. This is precisely the kind of behaviour that the time-frequency window of the Wavelet Transform exhibits.

2.3 The Integral Wavelet Transform (IWT)

The IWT is defined in terms of a special kernel function ψ (in L^2) known as a *wavelet.*

Definition 2.2 *If $\psi \in L^2(\mathbb{R})$ satisfies the admissibility condition:*

$$C_\psi = \int \frac{|\hat{\psi}(\omega)|^2}{|\omega|} d\omega < \infty, \tag{2.4}$$

then ψ is called a **basic wavelet.**

This requirement enables us to define the Integral Wavelet Transform:

Definition 2.3 *If we are given a basic wavelet, ψ, we may define the Integral Wavelet Transform, $(W_\psi f)(b,a)$, of an L^2 function f as*

$$(W_\psi f)(b,a) = |a|^{-\frac{1}{2}} \int f(t) \overline{\psi\left(\frac{t-b}{a}\right)} dt. \tag{2.5}$$

where a, $b \in \mathbb{R}$ and $f \in L^2$.

The overbar denotes complex conjugation. It was included for completeness — I only use real valued wavelets. The 'admissibility condition' is all that is required to define the IWT. However, one generally wishes to employ a basic wavelet which has other desirable properties; for example, one for which both ψ and $\hat{\psi}$ are *window functions.*

With this additional constraint, the admissibility condition implies that

$$\int \psi(t)\, dt = 0. \tag{2.6}$$

Consequently, ψ will be localised (since it is a window function) and exhibit a certain measure of osscilation (Equation 2.6) — hence the name 'wavelet' (See Figure 2.1).

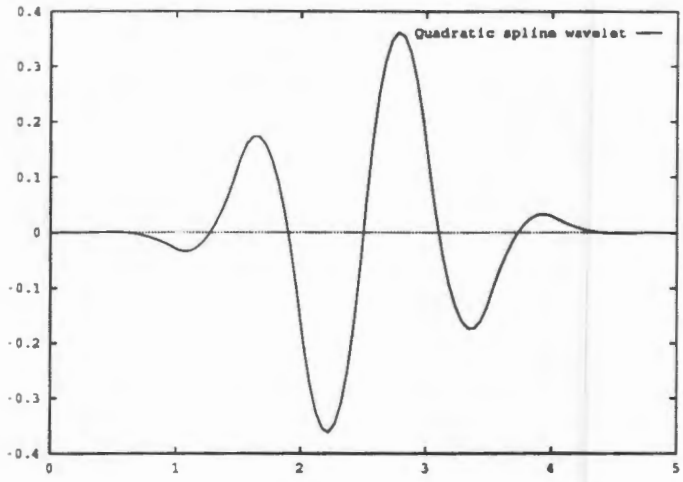


Figure 2.1: A (quadratic spline) wavelet. See Chapter 3 for details.

The variables b and a represent time (space) and frequency (scale), respectively. Because ψ is localised in both time and frequency, the IWT is also localised and gives us information in both domains, within the bounds of the Uncertainty Principle. In fact, the time-frequency window is

$$[b + at^* - a\Delta_\psi, b + at^* + a\Delta_\psi] \times \left[\frac{\omega^*}{a} - \frac{1}{a}\Delta_{\hat{\psi}}, \frac{\omega^*}{a} + \frac{1}{a}\Delta_{\hat{\psi}}\right] \tag{2.7}$$

where ω^*, $\Delta_{\hat{\psi}}$ and t^*, Δ_ψ are, respectively, the centres and widths of the frequency and time windows for the window function, $\psi(t)$. If we identify $\frac{\omega^*}{a}$ with the frequency variable ω, we see that the time/space window narrows for high frequency (small scale) phenomena ($a > 0$, a small) and widens for analysis of large-scale structures (a large). Of course, the frequency window simultaneously dilates or contracts in accordance with the Uncertainty Principle (See Figure 2.2).

Once our manipulations in the transform domain (e.g. thresholding) are complete, we wish to return to our original domain. This is accomplished via an inverse transform; however, if we permit our variables a, b to be continuous, this inverse transform involves computing a $(n+1)$ dimensional integral (if the function f has n space variables, $f(x_1, \ldots, x_n)$). The 1-D inverse formula is

$$f(x) = \frac{1}{C_\psi} \int \int (W_\psi f)(b, a) \psi_{b;a}(x) \frac{da}{a^2} db. \tag{2.8}$$

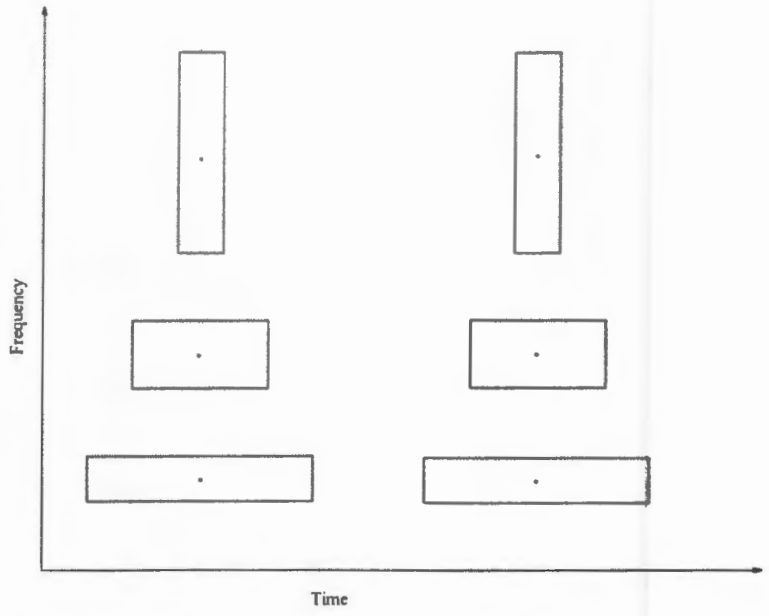


Figure 2.2: The wavelet time-frequency window. The window provides an illustration of the time-frequency localization of the IWT. When we have a low centre-frequency (the dots at the centres of the cell) the time window expands to better analyze these large-scale phenomena; for high centre-frequency, the time window narrows to provide a better analysis of small scale features. The window area is the same in all cases.

where $\psi_{b;a}(x)$ is defined as

$$\psi_{b;a}(x) \equiv |a|^{-\frac{1}{2}} \psi\left(\frac{x-b}{a}\right). \tag{2.9}$$

Since one generally only considers positive frequencies (scales), the inverse formula may be modified to reflect this. One may also discretize one or the other of a and b. For details, see [10, pg. 60–68].

2.4 The Discrete Wavelet Transform (DWT)

To ensure computational efficiency, we discretize both the scale, a and the time-localization, b, in the following manner: $a = 2^{-j}$, $b = k2^{-j}$, $k, j \in \mathbb{Z}$. If we then define

$$\psi_{j,k}(x) \equiv 2^{j/2}\psi(2^j x - k), \quad j, k \in \mathbb{Z}, \tag{2.10}$$

we obtain

$$(W_\psi f)\left(\frac{k}{2^j}, \frac{1}{2^j}\right) = \int f(x)\{\overline{2^{\frac{j}{2}}\psi(2^j x - k)}\}dx = \langle f, \psi_{j,k}\rangle, \tag{2.11}$$

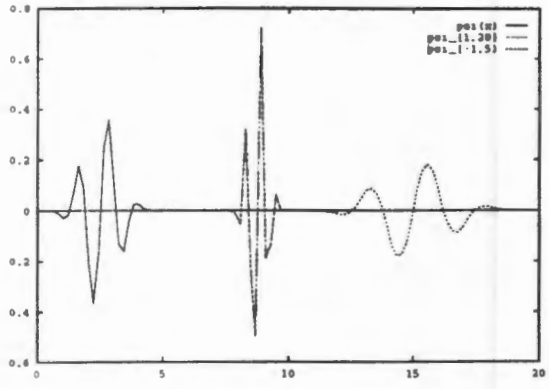


Figure 2.3: Three wavelet basis elements , corresponding to $\psi_{0,0}(x)$, $\psi_{1,20}(x)$ and $\psi_{-1,5}(x)$. The wavelet dilation and scaling are such that $\|\psi\|_{L^2} = 1$

where we have used inner product notation for compactness. From (2.7) we see that the parameter k localizes our transform about $(t^* + k)2^{-j}$ in the time/spatial domain[1]. This is the Discrete Wavelet Transform (DWT) of an $L^2(\mathbb{R})$ signal. Although our data is generally also discrete, we still employ this formalism to produce our wavelet coefficients, as is done in general [10, 13, 29]. This is possible since this discretized WT can be shown to be reversible (see below), which is all that one requires for a usable transform. Given the generality of L^2 functions, any discrete data we have may be considered as a sampled version of such function and we may proceed with the DWT outlined above. Since this transform is invertible, we can regenerate the input 'function' and hence, the original input samples.

There is, in fact, a DWT which operates on discrete input data (see, for example, [41]); however, this algorithm has filter lengths which become progressively larger as the scale increases (it is an *undecimated* wavelet transform.) This implies that the filtering operations become progressively more expensive the further down you decompose; this is not desirable in most circumstances.

In order that we may recover our original function from this sampling, we require that $\{\psi_{j,k}\}_{j,k\in\mathbf{Z}}$ form a *Riesz basis* [10]. This is a less restrictive requirement than orthogonality of the $\psi_{j,k}$ in that it permits us to construct wavelets which possess certain desirable properties which the latter do not. This will be expanded upon in Chapter 3.

[1]It should be noted that when ψ is (anti-)symmetric about the origin, t^* is zero.

Definition 2.4 *The function $\psi \in L^2(\mathbb{R})$ generates a Riesz basis if the following two properties are satisfied:*

1. *the linear span $< \psi_{j,k} : j,k \in \mathbb{Z} >$ is dense[2] in $L^2(\mathbb{R})$.*
2. *there exist positive constants A and B, $0 < A \leq B < \infty$ such that*

$$A \| \{c_{j,k}\} \|^2_{\ell^2} \leq \| \sum_j \sum_k c_{j,k} \psi_{j,k} \|^2_{L^2} \leq B \| \{c_{j,k}\} \|^2_{\ell^2} \tag{2.12}$$

for all $\{c_{j,k}\} \in \ell^2(\mathbb{Z}^2)$, where $\| \{c_{j,k}\} \|^2_{\ell^2} = \sum_j \sum_k |c_{j,k}|^2 < \infty$.

If ψ generates a Riesz basis, then there is a unique Riesz basis $\{\psi^{j,k}\}$ which is *dual* to $\{\psi_{j,k}\}$ i.e.

$$\langle \psi_{j,k}, \psi^{l,m} \rangle = \delta_{jl} \cdot \delta_{km}, \quad j,k,l,m \in \mathbb{Z}. \tag{2.13}$$

where δ_{jk} is the Kronecker Delta, being one when its indices are the same and zero otherwise. Every $f \in L^2$ then has the unique series expansion:

$$f(x) = \sum_{j,k} \langle f, \psi_{j,k} \rangle \, \psi^{j,k}(x). \tag{2.14}$$

If, in addition, there is a function $\tilde{\psi} \in L^2$ which generates the dual basis in the same fashion that ψ generates the Riesz basis $\{\psi_{j,k}\}$, then we may also expand $f(x)$ as follows:

$$f(x) = \sum_{j,k} \langle f, \psi^{j,k} \rangle \, \psi_{j,k}(x). \tag{2.15}$$

Formulae (2.14) and (2.15) are inverse transform formulae. These formulae relate the transform coefficients to the original function. Property (2.13) is called the ***bi-orthogonality*** property and is satisfied by all wavelets. If a wavelet is *orthogonal* it satisfies

$$\langle \psi_{j,k}, \psi_{l,m} \rangle = \delta_{jl} \cdot \delta_{km}, \quad j,k,l,m \in \mathbb{Z}. \tag{2.16}$$

That is, orthogonal wavelets are *self-dual*, having $\psi = \tilde{\psi}$. Thus, when one deals with orthogonal wavelets, the added complexity of having a dual present is avoided. A wavelet which is orthogonal only between scales (frequencies) is called a *semi-orthogonal* wavelet; this is formalized as follows:

$$\langle \psi_{j,k}, \psi_{l,m} \rangle = 0. \; j \neq l; \quad j,k,l,m \in \mathbb{Z}. \tag{2.17}$$

[2]That is, one can approximate any L^2 function to arbitrary precision by taking an appropriately defined linear combination of these basis functions.

Properties	Wavelet Classes		
	Orthogonal	Semi-orthog	Bi-orthog
Dual?	self-dual	yes	yes
Compact support?	wavelet	only wavelet	wavelet and dual
Symmetry?	no	yes	yes
Sequences?	finite	truncated	finite

Table 2.1: Some comparisons between the three major classes of wavelet. **Only orthogonal wavelets do not possess a separate Dual Wavelet. If the wavelet or dual has compact support, we can achieve perfect reconstruction; otherwise we must truncate when we implement. Symmetry is important to reduce distortion when we reconstruct. Orthogonal wavelets are orthogonal between scales and, within a particular scale, to their translates; semi-orthogonal wavelets are orthogonal between scales only; bi-orthogonal wavelets are not restricted by any orthogonality constraints save the bi-orthogonality relation, Equation (2.13).**

Table 2.1 provides a summary of the properties possessed by these three wavelet classes.

In some cases, all that one requires is that the reconstruction be *stable*, in the sense that small changes in the DWT coefficients do not lead to large changes in the reconstruction. If this is true, then one need only insist that the $\psi_{j,k}$ constitute a *frame* of $L^2(\mathbb{R})$. This is a weaker condition than requiring that they form a Riesz basis, and as such the resulting $\psi_{j,k}$ may not possess certain properties (such as linear independence) which some wavelet algorithms require. However, since every Riesz basis is also a frame, any nice properties which frames possess, are automatically assumed by Riesz bases.

The DWT as outlined above, requires that one generate coefficients for each scale, j. Since the number of scales is infinite, the transform has to be reworked to enable perfect reconstruction when we perform practical calculations. This remodeling is achieved by considering the multi-resolution nature of the transform; I will return to this question after clarifying the ideas behind a multi-resolution analysis.

2.5 Wavelets and Multi-Resolution Analysis

The concept of a *Multi-Resolution Analysis* (MRA) is of fundamental importance to wavelet theory. As the name implies, the notion of *resolution* occupies a central role: the analysis decomposes a signal into components of differing frequency. What does one mean by resolution? Intuitively, resolution serves to quantify the amount of permissable variation in a region. Thus, a high resolution image has a large amount of variation (detail) in a region, whereas a low resolution image is much smoother over this same region. Fourier Analysis

also provides a description of resolution: the frequency components of a signal provide a measure of the contribution of each (spatial) frequency component. Fine detail corresponds to high-frequency information: thus, a high-resolution signal will contain a large proportion high frequency sinusoids. Removing some of this detail i.e., leaving out some of the sinusoids leaves us with a 'lower' resolution, smoother, signal.

These ideas are formalized below. Since we wish to present as simple an introduction as possible, we first consider 1-D signals before generalizing to 2-D Signals. This also appropriate when one considers the manner in which our 2-D MRA is generated — see Section 2.5.3.

2.5.1 Wavelet Series and Decompositions

In the previous section we saw that every wavelet ψ generates a *wavelet decomposition* of $f(x) \in L^2(\mathbb{R})$, viz.

$$f(x) = \sum_{j,k} c_{j,k}\psi_{j,k}(x), \tag{2.18}$$

where the coefficients of the wavelet basis are are given by the IWT with respect to $\tilde{\psi}$. If we define the subspaces $W_j \equiv \mathrm{clos_{L^2}span}\{\psi_{j,k} : k \in \mathbb{Z}\}$, that is, define *scale* subspaces or *frequency bands*, W_j, then we may write

$$L^2(\mathbb{R}) = \dot{\sum_j} W_j \equiv \cdots \dot{+} W_{-1} \dot{+} W_0 \dot{+} W_1 \dot{+} \cdots \tag{2.19}$$

That is, every $f(x) \in L^2(\mathbb{R})$ has a *direct sum* decomposition:

$$f(x) = \cdots g_{-1}(x) + g_0(x) + g_1(x) + \cdots, \tag{2.20}$$

with $g_j \in W_j$, for all $j \in \mathbb{Z}$.

If ψ is an orthogonal (or semi-orthogonal) wavelet, we have $\langle g_l, g_j \rangle = 0$, $j \neq l$ where $g_k \in W_k$; that is, the subspaces W_j are mutually orthogonal, written as $W_i \perp W_j$, $i \neq j$. Then the direct sum in (2.19) becomes an *orthogonal direct sum* which is written in a similar manner,

$$L^2(\mathbb{R}) = \bigoplus_j W_j \equiv \cdots \oplus W_{-1} \oplus W_0 \oplus W_1 \oplus \cdots. \tag{2.21}$$

Both orthogonal and semi-orthogonal wavelets generate such orthogonal decompositions of $L^2(\mathbb{R})$ (in both cases the wavelet basis is orthogonal across scales and hence the scale subspaces are mutually orthogonal).

One may now define so-called ***approximation spaces***, V_j, which contain the jth resolution approximations of function in $L^2(\mathbb{R})$, in terms of our ***detail spaces***, W_j as follows:

$$V_j = \cdots \dot{+} W_{j-2} \dot{+} W_{j-1}, \quad j \in \mathbb{Z} \tag{2.22}$$

Approximation functions in a lower resolution subspace have had their higher (spatial) frequency components removed during the approximation operation and are consequently less detailed (they have been smoothed).

These definitions ensures that the following properties hold when we have a valid wavelet [10, pg. 120–121]:

1. $\cdots \subset V_{-1} \subset V_0 \subset V_1 \subset \cdots$;
2. $\mathrm{clos}_{L^2}\left(\bigcup_j V_j\right) = L^2(\mathbb{R})$;
3. $\bigcap_j V_j = \{0\}$;
4. $V_{j+1} = V_j \dot{+} W_j, \quad j \in \mathbb{Z}$;
5. $f(x) \in V_j \iff f(2x) \in V_{j+1}, \quad j \in \mathbb{Z}$.

Property 1 tells us that an approximation function at a specified resolution is also contained in all higher resolution approximation spaces. This approximation may be viewed as an input function which does not possess intrinsic detail at a higher resolution. Thus, the approximation operation projects this function into all the higher resolution spaces, 'stripping off' non-existent detail in the process.

Property 2 states that by combining a sufficient number of detail signals (elements of the detail spaces) we can approximate our input function to arbitrary precision. The set union provides us with all the detail spaces and we then sum the appropriate difference signals from each, taking as many as we need to achieve a prescribed accuracy. The closure operation ensures that we can generate all of $L^2(\mathbb{R})$, since the procedure of adding detail levels only allows one to approach the input signal, and hence we need to include the limit functions as well.

Property 3 is essentially the inverse of Property 2, since it strips away detail rather than adding it. As we consider more of the approximation spaces, the set intersection becomes smaller and smaller, hence the amount of detail we can produce becomes progressively less (having fewer subspaces to synthesize our function from). In the limit, when we take the intersection of all the V_j (and this includes those with near infinitely low resolution), we are left with no detail at all. That is, a signal which corresponds to the zero function, possessing no energy and, consequently, no information.

Property 4 encapsulates the essence of a MRA: an approximation function is the sum of a lower-resolution (smoothed) approximation and the detail difference between them. The former resides in the approximation space, V_j, and the latter in the detail space, W_j. The formula itself follows immediately from Equation (2.22).

Property 5 tells us that a function which resides in a particular approximation space, V_j, also resides in the higher resolution approximation space, V_{j+1}, but scaled in such a manner that it now reflects the resolution of the latter space. The factor of two is a consequence of our dyadic resolution specification (i.e. resolution $= 2^j$).

2.5.2 Multi-Resolution Analysis of 1-D Signals

Just as the subspaces W_j are generated by the wavelet ψ, it may be possible to find a single function ϕ which generates a Riesz basis $\{\phi_{j,k}\}$ of V_j, where $\{\phi_{j,k}\}$ is defined in an analogous manner to $\{\psi_{j,k}\}$. If this is possible we have a *Multi-Resolution Analysis* of $L^2(\mathbb{R})$, satisfying the properties listed above.

Definition 2.5 *A function $\phi \in L^2(\mathbb{R})$, called the* scaling function, *is said to generate a MRA if it generates a nested sequence of closed subspaces V_j which satisfy Properties 1,2,3 and 5 of Section 2.5.1, where*

$$V_j = \mathrm{clos}_{L^2} < \phi_{j,k} : k \in \mathbb{Z} >, \quad j \in \mathbb{Z}$$

and $\{\phi_{0,k}\}$ forms a Riesz basis of V_0.

This definition emphasizes an important subtlety involved in the construction of a MRA: the scaling function is assumed to have an identity which is independent from the existence

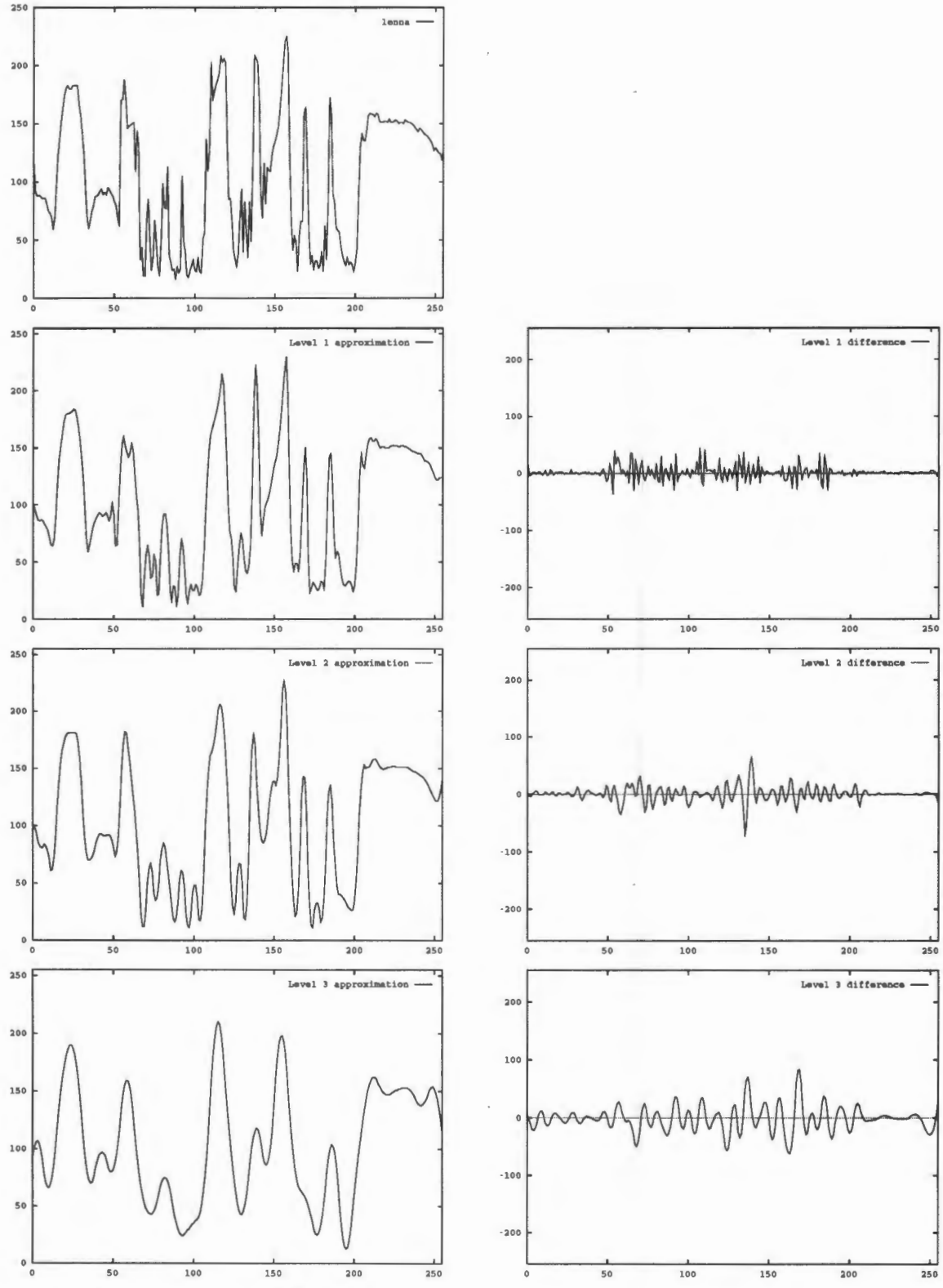


Figure 2.4: An example of a 1-D MRA. The top-most signal represents our input (taken from the space V_0). Each subsequent (lower) resolution tier consists of an approximation signal (left) and a detail signal (right). The detail signal for a particular level is obtained by differencing the current and previous approximation signals.

of a wavelet. Indeed, Equation (2.26) shows that the wavelet is dependent on our choice of scaling function. One has to decide, based on the particular needs of one's application, which sequence $\{p\}$ will describe the scaling function. For example, the choice of a binomial filtering kernel produces a spline scaling function; the sequence $\{q\}$ may then be deduced from additional constraints enforced by the user as well as those imposed by the choice of $\{p\}$.

The detail spaces, W_j, referred to above, constitute additional (useful) structure which is imposed on the MRA; the aim is then to find a wavelet which will generate the bases to construct these spaces. One can always define these complement spaces; hence, Property 4, while not explicitly required for a MRA, is always assumed to hold.

It is worthwhile mentioning here that the alternative wavelet-MRA convention has its nested subspaces defined such that those with higher indices are nested in those with lower indices, that is, $V_j \subset V_{j-1}$. Naturally, the other properties must be appropriately adjusted to reflect this; for example, the scale parameter j is replaced with $-j$ in the expansion of $\psi_{j,k}(x)$. For details on this formalism the reader is referred to [13].

Given that our subspaces $\{V_j\}$ and $\{W_j\}$ are generated by a scaling function and wavelet, respectively, we wish to utilise the structure of a MRA to find an algorithm for efficient calculation of our different resolution approximations. From Property 2 in the list above, we are able to approximate our initial function $f(x)$ as closely as desired by considering its projection, f_N, into a subspace V_N with sufficient detail for our purposes (we usually take this to be V_0). Then, by Property 4, this function is expressible as the direct sum of the next lower resolution approximation and the detail difference between them:

$$f_N = f_{N-1} + g_{N-1}, \quad f_{N-1} \in V_{N-1}, \quad g_{N-1} \in W_{N-1}. \tag{2.23}$$

After M iterations of this decomposition, we obtain:

$$f_N = g_{N-1} + g_{N-2} + \cdots + g_{N-M} + f_{N-M}. \tag{2.24}$$

We are free to proceed further, but after a point this becomes meaningless since the useful informational content of the lowest resolution signal may be nil. Equation (2.24) expresses the fact that our Nth level approximation consists of a low resolution (smoothed) approximation and a series of different resolution detail functions. Using this decomposition strategy one is able to derive a means of calculating the basis coefficients of both the smoothed

and detail spaces without explicitly referring to either the wavelet or the scaling function (details in [29] and [10, pg. 157–159]).

Since $\phi \in V_0$ and $\psi \in W_0$ are in V_1 (by Properties 1 and 4) and this space has basis

$$\left\{\phi_{1,k} \equiv 2^{1/2}\phi(2x-k): \quad k \in \mathbb{Z}\right\}$$

there are two sequences $\{p_k\}$ and $\{q_k\} \in \ell^2$ such that

$$\phi(x) = \sum_k p_k \phi(2x-k); \tag{2.25}$$

$$\psi(x) = \sum_k q_k \phi(2x-k); \quad \forall x \in \mathbb{R}. \tag{2.26}$$

These formulae are known as *two-scale* relations (of the scaling function and wavelet, respectively), since they relate the functions $\psi(x)$, $\phi(x)$ to the scaled translates of the scaling function, $\phi(x)$. Furthermore, since both $\phi(2x)$ and all its translates are in V_1 (they are the unweighted basis elements) and $V_1 = V_0 \dot{+} W_0$, one is able to derive the following relationship [10, pg. 142–143, 157]:

$$\phi(2x-l) = \sum_k [a_{l-2k}\phi(x-k) + b_{l-2k}\psi(x-k)], \quad l \in \mathbb{Z} \tag{2.27}$$

where the sequences are also in ℓ^2 [10]. This relation is called the ***decomposition relation*** of ψ and ϕ. Given these four sequences, one is able to formulate ***decomposition*** and ***reconstruction*** algorithms for the detail and approximation coefficients. These algorithms provide a means of implementing the DWT.

Since $f_j \in V_j$ and $g_j \in W_j$, we have the following series representations:

$$f_j(x) = \sum_k c_k^j \phi(2^j x - k), \quad \left\{c_k^j\right\}_{k\in\mathbb{Z}} \in \ell^2 \tag{2.28}$$

$$g_j(x) = \sum_k d_k^j \psi(2^j x - k), \quad \left\{d_k^j\right\}_{\mathbb{Z}} \in \ell^2 \tag{2.29}$$

where the normalization factor $2^{j/2}$ has been included in the sequence. The superscript in the sequence represents the resolution level — it is not an exponent. In particular, using Equation (2.24):

$$f(x) \equiv f_0(x) = g_{-1}(x) + \cdots + g_{-J}(x) + f_{-J}(x) \tag{2.30}$$

$$= \sum_k d_k^{-1}\psi_{-1,k}(x) + \cdots + \sum_k d_k^{-J}\psi_{-J,k}(x) + \sum_k c_k^{-J}\phi_{-J,k}(x). \tag{2.31}$$

Thus, given the sequences $\{d_k^j, j = 1, \ldots, J,\ k \in \mathbb{Z}\}$ and $\{c_k^J,\ k \in \mathbb{Z}\}$, we can reconstruct our input function, by expanding (2.31).

The sequence $\{d_k^j\}_{k\in\mathbb{Z}}$ corresponds directly to the ***wavelet coefficients*** generated by the DWT (taken w.r.t. the dual wavelet); the coefficients $\{c_k^j\}_{k\in\mathbb{Z}}$ encode the information contained in the wavelet coefficients for the resolution levels we do not wish to evaluate. These coefficients are formally obtained (for the wavelet series expanded in terms of $\psi_{j,k}$) as $c_k^j =< f, \tilde{\phi}_{j,k} >$, where $\tilde{\phi}_{j,k}$ is generated from a unique ***dual scaling function***, $\tilde{\phi}$, which satisfies the relationship $< \tilde{\phi}_{j,k}, \psi_{j,l} >= 0;\ j, k, l \in \mathbb{Z}$ [10, pg. 154]. However, I do not use this approach explicitly: rather than computing integrals, one employs the algorithms introduced in [29] to produce the desired transform coefficients.

The *decomposition algorithm* is given by:

$$c_k^{j-1} = \sum_l a_{l-2k} c_l^j; \tag{2.32}$$

$$d_k^{j-1} = \sum_l b_{l-2k} c_l^j. \tag{2.33}$$

That is, given the basis coefficients of a smoothed function on resolution level j, we can use this algorithm to find the *approximation* and *detail* detail coefficients, $\{c_k^j\}$ and $\{d_k^j\}$ respectively, for all successive lower resolution levels. When using this scheme to implement the DWT, one decompose from level 0 to a particular level, J, keeping all the wavelet (detail) coefficients and discarding all but the last tier of approximation coefficients. This provides all the information we require to reconstruct our input function. This reconstruction is performed by means of the following *reconstruction algorithm*:

$$c_k^j = \sum_l [p_{k-2l} c_l^{j-1} + q_{k-2l} d_l^{j-1}]. \tag{2.34}$$

Both these algorithms can be recast as convolutions. Equation (2.32) in the decomposition algorithm can be considered as the downsampled (discrete) convolution of $\{a_{-k}\}$ and $\{c_k^j\}$.

The second equation may be viewed similarly (See Figure 2.5). One can show that $\{a_k\}$ acts as the impulse response of a low-pass filter, while $\{b_k\}$ acts as the impulse response of a band-pass filter[3]. The filter $\{a_k\}$ corresponds to a discrete version of the dual scaling function, whereas the filter $\{b_k\}$ may be considered as the discrete analogue of the dual wavelet.

[3]Band-pass, because the detail coefficients represent difference information in particular frequency bands.

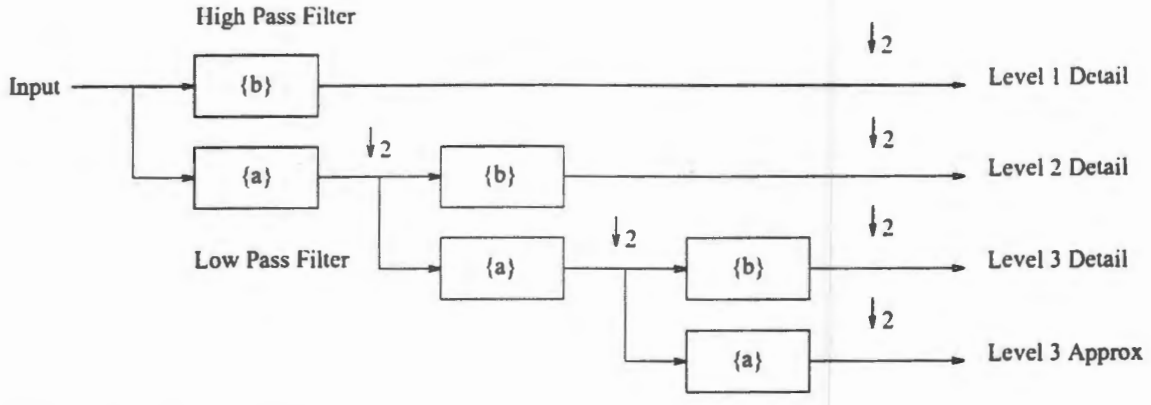


Figure 2.5: The decomposition process. The diagram represents the filtering steps that are involved in the decomposition process. Each box corresponds to a convolution with the time-reversal of indicated sequence i.e., $\{a_{-k}\}$. The arrows indicate that the output from each stage is decimated (that is, only even numbered samples are kept). The low-pass component is then fed back into the filter bank and the process is repeated.

The reconstruction algorithm performs a convolution between its sequences only after up-sampling the detail and smoothing coefficients, i.e., zeros are placed between consecutive coefficient values (Figure 2.6). As in the previous case, the filters may also be considered as discrete versions of continuous counterparts: $\{p_k\}$ corresponds to the scaling function, and $\{q_k\}$ corresponds to the wavelet. The former is a low-pass filter, whereas the latter is band-pass in nature. This convolutional interpretation means that one can implement these schemes effectively in hardware using FFT-based algorithms.

2.5.3 Multi-Resolution Analysis of 2-D Signals

Since we wish to apply these techniques to images, we have to extend the previous results to 2-D. There are different methods which one can employ in this generalization. One may, for example, attempt to find a 2-D scaling function, $\Phi(x, y)$, which generates a 2-D MRA, $\mathbf{V}_j$ which satisfies all the appropriate analogues to our 1-D scheme [13, pg. 317–318]. This approach provides us with *one* 2-D wavelet, $\Psi(x, y)$; however, the extra degree of freedom that 2-D space provides us enables us to have *several* wavelets underlying our detail spaces, as the next paragraph illustrates.

However, the most popular method for constructing a MRA of $L^2(\mathbb{R}^2)$, is to define the space $\mathbf{V}_j$ as the tensor product of the space V_j with itself [13]. Then $\mathbf{V}_j$ induces a MRA of

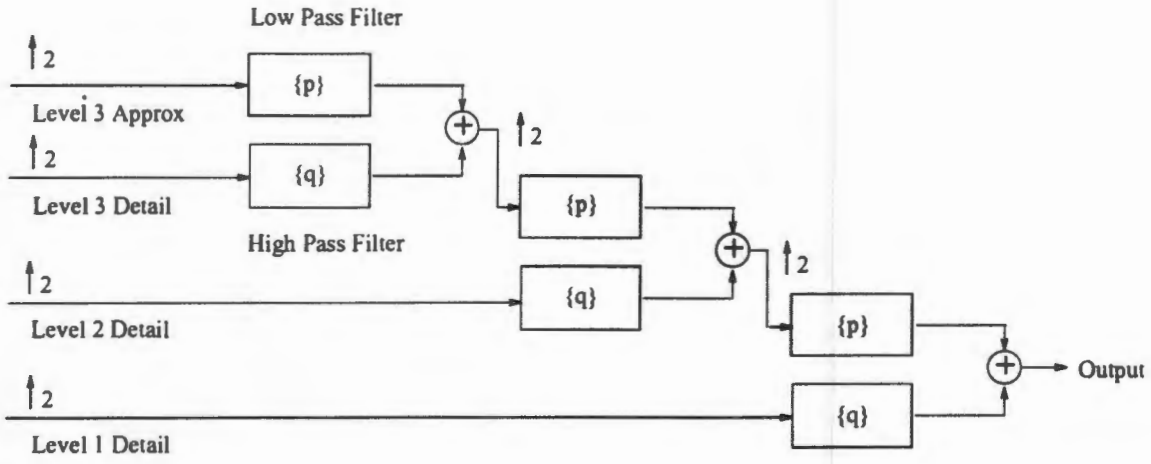


Figure 2.6: The reconstruction process. To reconstruct the approximation coefficients for a particular level from the DWT representation, one again uses a convolutional scheme. Now however, the input sequences to each stage are *upsampled* and the outputs are added together. This process is repeated until all the detail tiers have been used.

$L^2(\mathbb{R}^2)$: $\mathbf{V}_j \subset \mathbf{V}_{j+1}$ with the properties we discussed before and a scaling function

$$\Phi_{j;m,n}(x,y) \equiv 2^j \phi(2^j x - m)\phi(2^j y - n), \quad m,n \in \mathbb{Z}. \tag{2.35}$$

Defining W_j to be the orthogonal complement[4] of V_j in V_{j+1} then gives us:

$$\begin{aligned} \mathbf{V}_{j+1} &= V_{j+1} \otimes V_{j+1} \\ &= (V_j \oplus W_j) \otimes (V_j \oplus W_j) \\ &= (V_j \otimes V_j) \oplus [(W_j \otimes V_j) \oplus (V_j \otimes W_j) \oplus (W_j \otimes W_j)] \\ &= \mathbf{V}_j \oplus \mathbf{W}_j. \end{aligned} \tag{2.36}$$

So the complementary subspace $\mathbf{W}_j$ consists of three pieces, with Riesz Bases,

$$\psi_{j,m}(x)\phi_{j,n}(y), \quad \text{for} \quad (W_j \otimes V_j); \tag{2.37}$$

$$\phi_{j,m}(x)\psi_{j,n}(y), \quad \text{for} \quad (V_j \otimes W_j); \tag{2.38}$$

$$\psi_{j,m}(x)\psi_{j,n}(y), \quad \text{for} \quad (W_j \otimes W_j) \tag{2.39}$$

These three *detail* spaces contain the detail lost between two consecutive resolution approximations. In fact, each space contains the sharp variation (high frequency) information of the previous approximation in a *particular direction*[5]

[4] We will use $\oplus$ rather than $\dot{+}$ since this is appropriate for the spline-based MRA we will consider shortly.

[5] $\psi(x)$ and $\phi(x)$ may be viewed as the impulse responses of a band- and low-pass filter, respectively. The parameter j produces scaled filters, whereas k allows these filters to pick out localised detail information about $(t^* + k)2^{-j}$.

Figure 2.7: The first four approximation images in our quadratic Multi-Resolution structure. The resolution decreases clockwise from top left.

Equation (2.37) gives the basis for the detail space which detects (represents) sharp variations in our 2-D detail signal which are orientated in the x-direction, i.e. vertical edges. Similarly, the basis given by (2.38) will represent edges in the horizontal direction. Equation (2.39) is the basis for the detail space which detects diagonal edges. We now define *three* 2-D wavelets as follows,

$$\Psi^{[1]}(x,y) = \phi(x)\psi(y) \tag{2.40}$$

$$\Psi^{[2]}(x,y) = \psi(x)\phi(y) \tag{2.41}$$

$$\Psi^{[3]}(x,y) = \psi(x)\psi(y). \tag{2.42}$$

Then $\left\{\Psi^{[i]}_{j;m,n}(x,y) \equiv 2^j\Psi^{[i]}_{m,n}(2^jx-m,2^jy-n);\ i=1,2,3;\ m,n\in\mathbb{Z}\right\}$ is a Riesz basis for $\mathbf{W}_j$; when we allow the scale parameter j to vary over all integers this basis is then a basis for $L^2(\mathbb{R}^k)$. As in the 1-D case, we can also find a dual basis, $\tilde{\Psi}^{[i]}_{j;m,n}$ which satisfies the

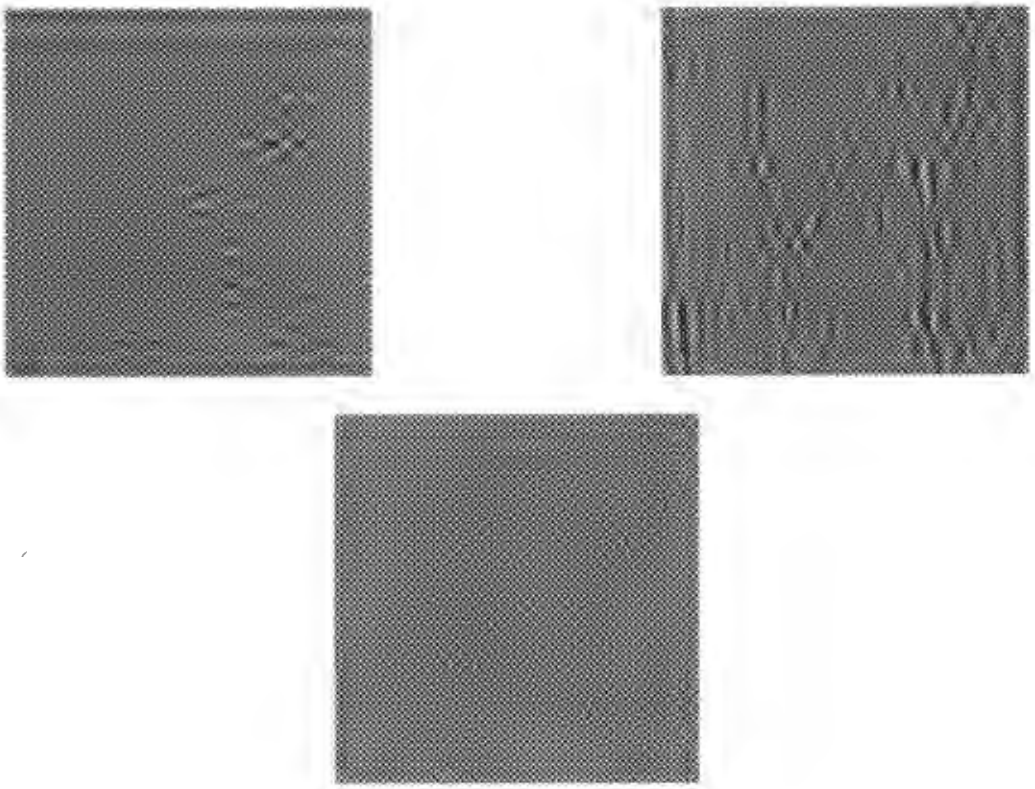


Figure 2.8: Third level detail images. These images illustrate the directional sensitivity of the Wavelet Transform. It decomposes the detail lost between consecutive levels into images which contain the detail information in the horizontal (top left), vertical (top right) and diagonal (bottom) directions. The bases underlying these images are (respectively) the wavelets $\Psi^{[2]}(x,y)$, $\Psi^{[1]}(x,y)$ and $\Psi^{[3]}(x,y)$ — cf. Equations (2.40, 2.41,2.42).

bi-orthogonality relationship

$$\langle \Psi^{[m]}_{k;i,p}, \tilde{\Psi}^{[n]}_{l;j,q} \rangle = \delta_{mn}\delta_{kl}\delta_{ij}\delta_{pq}. \tag{2.43}$$

In our analysis, we will use a 2-D L^2 function, $I(x,y)$, to represent the intensity profile of our image, on the understanding that the lattice of points $\mathbb{Z} \times \mathbb{Z}$ contains our image description. We will thus manipulate the image as if it were in $L^2(\mathbb{R}^2)$, until we wish to produce output; at such time we will sample $I(x,y)$ on this integer lattice to produce our pixel values. As explained earlier, the DWT which is generally employed is defined for $L^2(\mathbb{R})$ functions, rather than a discrete data set. Thus, the introduction of an image function, while apparently out of place in a discrete digital scenario, is in fact quite standard: no mathematical rigour has been sacrificed. Furthermore, the issue of efficient computation of

the transform coefficients has already been dealt with: the decomposition and reconstruction algorithms ensure that we do not have to compute integrals at all. I will return to these considerations in the next chapter, in which a full discussion of the spline-wavelet scheme employed will be undertaken.

The kth resolution approximation of our image, $I(x,y)$, is given by

$$I_k(x,y) = \sum_{i,j} c_{ij}^k \Phi_{k;i,j}(x,y). \tag{2.44}$$

This relationship has the same form as the 1-D case — a consequence of the tensor product technique used to generate the 2-D MRA. Similarly, just as we did in 1-D, we may write

$$I_{k+1}(x,y) = I_k(x,y) + g_k(x,y). \tag{2.45}$$

where $g_k(x,y)$ encodes the detail lost between resolutions k and $k+1$ and is given by

$$g_k(x,y) = \sum_{i,j,p=1}^{3} d_{k\,ij}^{[p]} \Psi_{k;i,j}^{[p]} \tag{2.46}$$

The 2-D decomposition algorithm is a simple extension from the 1-D algorithm; the same sequences are used, but now they occur in product combinations.

$$c_{kl}^{j-1} = \sum_m \sum_n a_{m-2k} a_{n-2l} c_{mn}^j \tag{2.47}$$

$$d1_{kl}^{j-1} = \sum_m \sum_n a_{m-2k} b_{n-2l} c_{mn}^j \tag{2.48}$$

$$d2_{kl}^{j-1} = \sum_m \sum_n b_{m-2k} a_{n-2l} c_{mn}^j \tag{2.49}$$

$$d3_{kl}^{j-1} = \sum_m \sum_n b_{m-2k} b_{n-2l} c_{mn}^j. \tag{2.50}$$

As with the decomposition algorithm, the 2-D reconstruction algorithm employs the same sequences as its 1-D counterpart:

$$\begin{aligned} c_{km}^j = \quad & \textstyle\sum_l \sum_p p_{k-2l} p_{m-2p} c_{lp}^{j-1} + \\ & \textstyle\sum_l \sum_p p_{k-2l} q_{m-2p} d1_{lp}^{j-1} + \\ & \textstyle\sum_l \sum_p q_{k-2l} p_{m-2p} d2_{lp}^{j-1} + \\ & \textstyle\sum_l \sum_p q_{k-2l} q_{m-2p} d3_{lp}^{j-1}. \end{aligned} \tag{2.51}$$

Once again, a convolutional interpretation can be attached to these formulae; however, these 2-D convolutions are *separable* and can thus be implemented by successive horizontal

and vertical filtering operations with two 1-D filters — a far more economical approach. Our decomposition algorithm proceeds by filtering the 2-D coefficient matrix, $\{c_{ij}\}$, along each of its rows, downsampling each in turn, and then repeating this procedure (with the second filter) on the resulting columns. The reconstruction algorithm differs in that the input 2-D sequence (the detail and approximation coefficients) are first *upsampled* before the convolutions are applied.

Given the input approximation coefficients, one can decompose and reconstruct as desired; the only time we must explicitly concern ourselves with function evaluations is when we wish to compute our difference or approximation images. The former are computed via Equation (2.46), and the latter with Equation (2.44). The means of determining the input coefficients, $\{c^0_{ij}\}$, will be dealt with in the next chapter.

2.6 Concluding Remarks

Having read this chapter, the reader should now have a working knowledge of the Wavelet Transform and a good understanding of the relationship between this transform and multi-resolution analysis. The reason for choosing the WT over a STFT should be apparent: it provides a better analysis of general signals. In addition, the transform can be described by a series of convolutions, making it suitable for hardware implementation.

The 2-D wavelet scheme is a simple generalization from 1-D and is constructed by means of tensor products. This construction adds a new element, which was absent in one dimension: directional sensitivity. That is, the (separable) filters which characterize the 2-D Wavelet Transform, are *oriented*: they only pass information which is oriented in the same manner that they are. This aspect is useful in image analysis, where one might very well wish to extract features which have a particular orientation. The wavelet MRA is a sub-band filtering scheme, hence the directional sub-band interpretation of human visual processing which is associated with these schemes [28, 15] is applicable. These issues will be discussed in great depth when we deal with the WT in the context of an image compression scheme. However, before doing this, we must negotiate a little more theory, viz. that of the semi-orthogonal spline MRA.

Chapter 3

A Spline-Based Multi-Resolution Analysis

The previous chapter introduced the general theory of multi-resolution analyses — the function of the following sections is to fix the particular MRA upon which the image compression/synthesis algorithms which follow are based. I shall be using a spline-based semi-orthogonal MRA, independently developed by Chui [10] and Unser [48]. This MRA has a number of desirable features:

- It provides a wavelet and scaling function with compact support;
- Closed form or non-iterative expressions are available for all the formulae employed (compare this with the orthogonal scheme of Daubechies, [13] — where the scaling function is obtained as the limit of an iterative process);
- Both the scaling function and the wavelet, as well as their duals, possess either linear of generalized linear phase. This property, a consequence of the symmetry possessed by these functions, is also reflected in the decomposition and reconstruction filters. If a filter has (generalized) linear phase then distortions in the input sequence (perhaps from thresholding or rounding errors) are not unduly magnified by the filtering operation. If a wavelet is orthogonal and has compact support it cannot possess linear phase [10] (the exception being the Haar wavelet, which does not provide a window function and is consequently not well suited to signal analysis).

- Fast and efficient algorithms exist to compute the values of the sequences required. For example, one can use so-called Linear Pascal Triangular Algorithms (LPTA's) to compute the values of the reconstruction sequences rapidly. Furthermore, the symmetry of the sequences implies that one need only compute a subset of all the coefficients.
- The symmetry of the sequences means that one can use simple symmetric boundary conditions.

The are, of course, some drawbacks associated with this approach:

- Separate, non-orthogonal dual functions exist;
- These dual functions have infinite support and, consequently, so do the decomposition sequences $\{a_k\}$ and $\{b_k\}$. They are infinite impulse response (IIR) filters.

The latter point implies that these infinite sequences have to be truncated before implementation — this is, in fact, what I have done. An alternative approach, which does not involve explicit truncation, was proposed by Unser in [48]: rather than truncating the sequences directly, one decomposes the inverse filter components of these sequences (which are responsible for their having an IIR) into a pair of recursive filters which can be implemented with very few arithmetic operations. The stopping condition on the recursion implicitly encodes the sequence truncation. The details of this approach are given in [47].

The final decision as to which MRA scheme one should employ will ultimately depend on the nature of the application involved. An important consideration was the fact that the above scheme was based on polynomials. The hardware implementation underlying the synthesis algorithms discussed later, achieves optimal performance when reconstructing polynomials; hence, any scheme based on polynomials is highly desirable. While there are orthogonal schemes (such as that of Le Marie, used in [29]) which are also spline-based, the wavelet no longer possesses compact support. This, in turn, implies that the underlying decomposition and synthesis filters are IIR filters and must consequently be truncated. In contrast, the semi-orthogonal spline approach outlined below, only requires that the decomposition filters be truncated. The decomposition sequences for both methods decay exponentially and consequently induce similar errors upon truncation. The reconstruction

filters are very short FIR filters. This implies that reconstruction is very fast with the latter method, which is desirable when retrieving and reconstructing an encoded image.

Furthermore, since wavelet compression involves extensive quantization of wavelet coefficients, the linear phase characteristics of the spline filters is very useful.

The following sections are intended to provide fairly extensive coverage of the issues surrounding this spline MRA, particularly those concerning implementation. Section 3.1 and Section 3.2 provide the theory one needs to utilize the spline MRA as well as sketching the derivation processes employed to obtain the decomposition and reconstruction sequences. The issue of boundary conditions is extensively addressed in Section 3.3. A brief comparison between cubic and quadratic spline schemes is then given in Section 3.4. Finally, the means of producing the initial input approximation coefficients is discussed (Section 3.5).

3.1 Cardinal Splines

A (polynomial) spline curve consists of polynomial segments which are pieced together at *knot-points* with a pre-defined level of continuity.

One normally requires that they have at least C^1 continuity, that is, that the tangent does not change abruptly when one moves across segment boundaries. Of course, the *degree* of the polynomial determines the maximum permissible smoothness across a knot-point: a linear spline curve cannot possess more than simple continuity across a join. Spline curves are normally encountered in the context of data set interpolation; in this case, the knot-points constitute a set of pairs, $\{(x_i, f(x_i)) : i \in \mathbb{Z}, \}$ through which the interpolant is constrained to pass. The intervals between the knot-points are smoothly filled by the spline curve. The term *knot-sequence* is used to mean the set of abscissas corresponding to the knot-points.

If the knot-points are spaced at regular intervals one speaks of a *cardinal spline*. The functions which underlie the spline MRA are of this kind. The *space of cardinal splines*, S_m, contains all cardinal spline functions, of order m:

Definition 3.1 *For a positive integer m, the space S_m, of cardinal splines of order m and with knot-sequence $\mathbb{Z}$, is the set of all functions $f \in C^{m-2}$ such that the restrictions of f to any interval $[k, k+1)$, $k \in \mathbb{Z}$, are in the space of polynomials of degree at most $m-1$, π_{m-1}; that is*

$$f|_{[k,k+1)} \in \pi_{m-1}, \quad k \in \mathbb{Z}. \tag{3.1}$$

The elements of S_m are expressible as a weighted sum of mth order *cardinal B-splines*:

$$f(x) = \sum c_k N_m(x-k), \quad f \in S_m. \tag{3.2}$$

The mth order cardinal B-spline is defined as

$$N_m(x) \equiv (N_{m-1} * N_1)(x) = \int_0^1 N_{m-1}(x-t)dt, \quad m \geq 2, \tag{3.3}$$

where

$$N_1(x) = \chi_{[0,1)}(x) = \begin{cases} 1 \text{ if } x \in [0,1); \\ 0 \text{ otherwise.} \end{cases} \tag{3.4}$$

The cardinal B-splines are thus generated by repeatedly convolving the unit box with itself. Figure 3.1 shows some of these functions. The 'central limit theorem' implies that this iterative process converges towards a Gaussian function.

Cardinal B-splines satisfy the following identity, which enables one to compute their values without resorting to integral formulations:

$$N_m(x) = \frac{x}{m-1} N_{m-1}(x) + \frac{m-x}{m-1} N_{m-1}(x-1). \tag{3.5}$$

The following list summarizes some of the properties which B-splines possess.

Compact Support: supp $N_m = [0, m]$, where *supp* means support,

Positivity: $N_m(x) > 0$, for $0 < x < m$,

Partition of Unity: $\sum_{k=-\infty}^{\infty} N_m(x-k) = 1$, for all x,

Symmetry: $N_m(\frac{m}{2} + x) = N_m(\frac{m}{2} - x), \quad x \in \mathbb{R}$.

So, B-splines are symmetric, have compact support and are never negative in value. They are also easy to compute — Equation (3.5). I will return to the computational advantages of cardinal spline functions when I discuss implementation of the spline multi-resolution analysis.

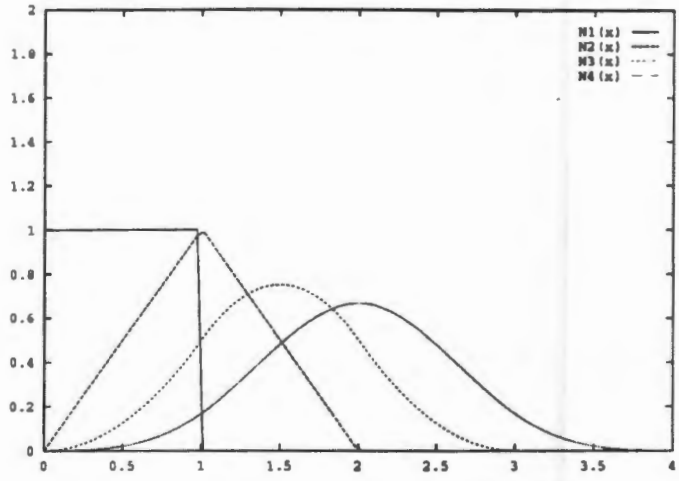


Figure 3.1: Spline scaling functions. **The cardinal spline scaling functions are generated by repeatedly convolving $N1(x)$ with itself.**

3.2 Cardinal Spline MRA

The cardinal spline spaces S^m have equally spaced integer knot-points. However, if one defines the spaces S_j^m, which contain cardinal splines with knot-points $2^{-j}\mathbb{Z}$, $\mathbb{J} \in \mathbb{Z}$, then we have the following relationship

$$\cdots \subset S_{-1}^m \subset S_0^m \subset S_1^m \cdots \tag{3.6}$$

since any cardinal spline curve defined on the knot-sequence $2^l\mathbb{Z}$ is also a cardinal spline on the finer knot-sequence $2^m\mathbb{Z}$, when $l > m$; the converse is, however, not true. This type of behaviour is precisely what one requires of a MRA. However, before one can say that the spaces $V_j^m \equiv S_j^m$ constitute a MRA, one must establish whether the 'scaling function' N_m generates a Riesz basis for V_0^m. If this is the case, then the set

$$\{2^{j/2}N_m(2^j x - k) : k \in \mathbb{Z}\}$$

is a Riesz basis for V_j^m and we have a valid MRA, satisfying the properties given in Definition 2.5. It turns out that is is indeed so [10, pg. 87–90]. Our scaling function, $N_m(x)$, must satisfy the two scale relation

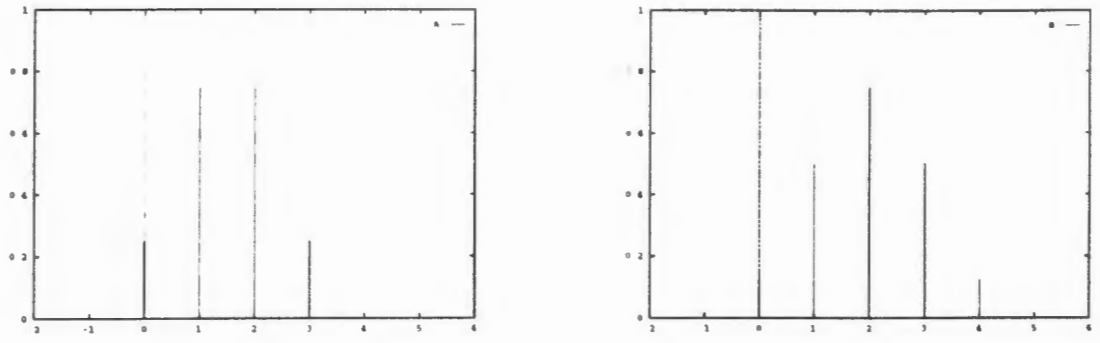


Figure 3.2: The reconstruction sequences $\{p_k\}$. 'A' shows this sequence for the quadratic case, while 'B' provides for the cubic case. The height of the impulse provides the magnitude of the sequence value at that index. Observe that these sequences have compact support and are symmetric.

$$N_m(x) = \sum_{k=0}^{m} p_k^m N_m(2x - k), \tag{3.7}$$

as explained in Chapter 2. By taking the Fourier transform of both sides of the equation and matching coefficients, this sequence is shown to be the binomial filter (Figure 3.2).

$$p_k^m = \begin{cases} 2^{-m+1} \begin{pmatrix} m \\ k \end{pmatrix} & \text{for } 0 \leq \text{k} \leq \text{m}; \\ 0 & \text{otherwise.} \end{cases} \tag{3.8}$$

A full characterization of the smoothing spaces, V_j^m, generated by these scaling function is given by

$$V_j^m = \{f \in C^{m-2} \cap L^2(\mathbb{R}) : \upsilon|_{[\frac{\gamma}{2^j}, \frac{\gamma+\kappa}{2^j})} \in \pi_{\flat-\kappa}, \ \gamma \in \mathbb{Z}\}. \tag{3.9}$$

This states that functions which are both in L^2 and satisfy the indicated continuity condition are elements of the jth resolution approximation space, provided that their restriction to the indicated interval shows that they are polynomials of degree at most $m - 1$. We see here, that as j becomes smaller, the intervals over which the function is required to have a uniform polynomial character become progressively larger. This explains the smoothed nature of low resolution approximations to the original function.

Having established the structure of the approximation spaces, one must now decide on the nature of the detail spaces and their associated wavelets. The wavelet must satisfy two constraints, in addition to those imposed by the choice of $N_m(x)$ as our scaling function:

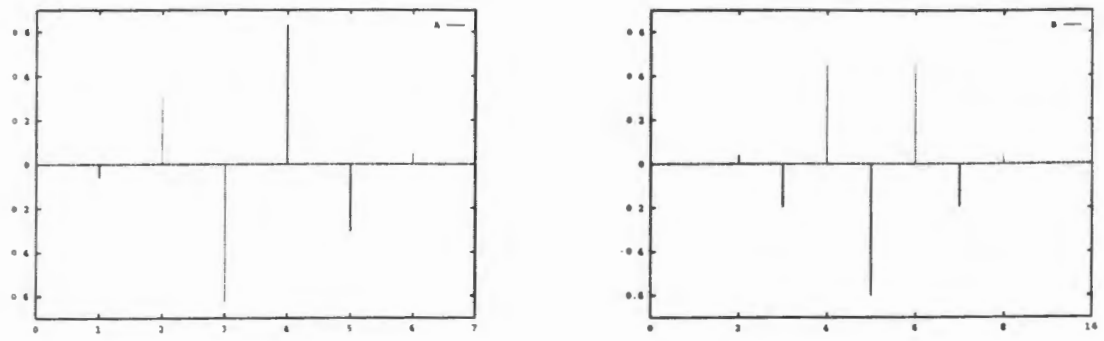


Figure 3.3: The reconstruction sequences $\{q_k\}$ for the quadratic ('A') and cubic ('B') cases. **The height of the impulse provides the magnitude of the sequence value at that index. Observe that these sequences have compact support and are symmetric: half-sample anti-symmetric for 'A' and whole-sample symmetric for 'B'.**

1. it must be orthogonal across scales and
2. it must have compact support.

The *B-wavelet*, $\psi_m(x)$, has support on $[0, 2m-1]$ and is given by [10, pg. 178–183]

$$\psi_m(x) = \sum_{k=0}^{3m-2} q_k N_m(2x-k) \tag{3.10}$$

where

$$q_n = \frac{(-1)^n}{2^{m-1}} \sum_{l=0}^{m} \binom{m}{l} N_{2m}(n+1-l),\ n = 0, \ldots, 3m-2. \tag{3.11}$$

This wavelet generates Riesz bases for the detail spaces W_j^m, in the manner discussed in Chapter 2. Figure 3.2 shows the quadratic and cubic wavelets. Note that the cubic wavelet is symmetric, whilst the quadratic wavelet is anti-symmetric. These symmetry conditions ensure that the former has linear phase and the latter, generalized linear phase. For this reason the cubic spline scheme is normally used; however, in this case a quadratic scheme was preferred. The motivations for this choice are discussed in Section 3.4.

The derivation of the decomposition sequences $\{a_k\}_{k\in\mathbf{Z}}$ and $\{b_k\}_{k\in\mathbf{Z}}$ is somewhat more difficult. I will provide an outline of the derivation; those desiring more information should consult [10].

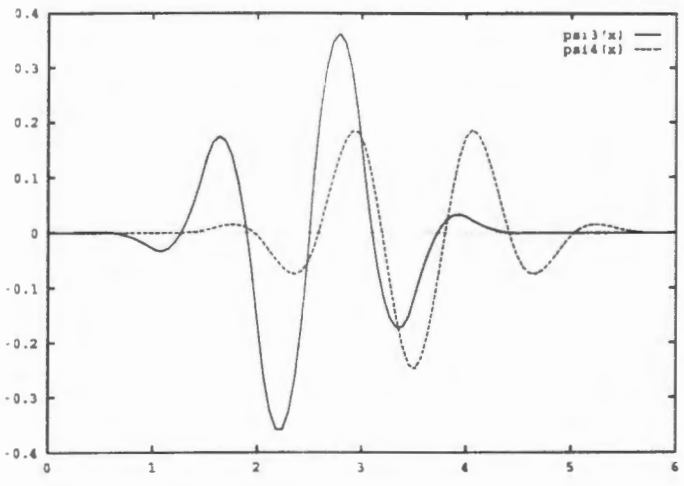


Figure 3.4: Cardinal spline wavelets. The cubic wavelet, $\psi_4(x)$ is symmetric, while the quadratic wavelet, $\psi_3(x)$ is anti-symmetric.

As a first step, one defines the following 'symbols' (which are modified Z-transforms)

$$\begin{aligned} P(z) &\equiv \frac{1}{2}\sum_k p_k z^k \\ Q(z) &\equiv \frac{1}{2}\sum_k q_k z^k \end{aligned} \tag{3.12}$$

where the sequences $\{p_k\}$ and $\{q_k\}$ are the two-scale sequences referred to previously.

The matrix

$$M_{P,Q}(z) \equiv \begin{pmatrix} P(z) & Q(z) \\ P(-z) & Q(-z) \end{pmatrix} \tag{3.13}$$

is used as a basis for the definition of two more complex functions, $G(z)$ and $H(z)$:

$$\begin{aligned} G(z) &= \frac{Q(-z)}{(\det \mathrm{M_{P,Q}})(z)} \\ H(z) &= \frac{P(-z)}{(\det \mathrm{M_{P,Q}})(z)} \end{aligned} \tag{3.14}$$

where 'det M' refers to the determinant of the matrix M. These functions are defined in this manner so that one may produce a simple inverse for the matrix $M_{P,Q}$. In fact,

$$\begin{aligned} M_{P,Q}(z)M^T_{G,H}(z) &= I, \\ M^T_{G,H}(z)M_{P,Q}(z) &= I, \quad |z| = 1. \end{aligned} \tag{3.15}$$

where I is the identity matrix and T represents a matrix transpose. The first of these equations (the only one referred to subsequently) is equivalent to the following pair:

$$\begin{aligned} P(z)G(z) + Q(z)H(z) &= 1, \\ P(z)G(-z) + Q(z)H(-z) &= 0, \quad |z| = 1. \end{aligned} \tag{3.16}$$

One may also write the following, when $\det \mathbf{M}_{\mathbf{P},\mathbf{Q}} \neq 0$ on $|z| = 1$

$$\begin{aligned} G(z) &= \frac{1}{2}\sum_k g_k z^k, \\ H(z) &= \frac{1}{2}\sum_k h_k z^k. \end{aligned} \tag{3.17}$$

where the sequences are in ℓ^1 (i.e., they are absolutely summable).

Equations (3.16) may be restated as follows:

$$\begin{aligned} P(z)[G(z) + G(-z)] + Q(z)[H(z) + H(-z)] &= 1; \\ P(z)[G(z) - G(-z)] + Q(z)[H(z) - H(-z)] &= 1, \quad |z| = 1. \end{aligned} \tag{3.18}$$

When this reformulation is coupled with Equations (3.17), one has

$$\begin{aligned} P(z)\sum_k g_{2k} z^{2k} + Q(z)\sum_k h_{2k} z^{2k} &= 1; \\ P(z)\sum_k g_{2k-1} z^{2k-1} + Q(z)\sum_k h_{2k-1} z^{2k-1} &= 1, \quad |z| = 1. \end{aligned} \tag{3.19}$$

Taking $z = e^{-i\omega/2}$ and multiplying the first equation by $\hat{\phi}(\frac{\omega}{2})$ and the second by $z\hat{\phi}(\frac{\omega}{2})$, one obtains the following relationships,

$$\begin{aligned} \hat{\phi}(\frac{\omega}{2}) &= \sum_k \left(g_{2k} z^{2k} P(z)\hat{\phi}(\frac{\omega}{2}) + h_{2k} z^{2k} Q(z)\hat{\phi}(\frac{\omega}{2}) \right); \\ \hat{\phi}(\frac{\omega}{2}) e^{-i\omega/2} &= \sum_k \left(g_{2k-1} z^{2k} P(z)\hat{\phi}(\frac{\omega}{2}) + h_{2k-1} z^{2k} Q(z)\hat{\phi}(\frac{\omega}{2}) \right). \end{aligned} \tag{3.20}$$

Now, the two-scale relations for the scaling and wavelet functions are given by

$$\begin{aligned} \phi(x) &= \sum_k p_k \phi(2x - k), \\ \psi(x) &= \sum_k q_k \phi(2x - k). \end{aligned} \tag{3.21}$$

Taking their Fourier transforms, yields $\hat{\phi}(\omega) = P(z)\hat{\phi}(\frac{\omega}{2})$ and $\hat{\psi}(\omega) = Q(z)\hat{\phi}(\frac{\omega}{2})$. Substituting these expressions into the above formulae and taking the inverse Fourier Transform of

both equations then gives

$$\begin{aligned} 2\phi(2x) &= \sum_k (g_{2k}\phi(x-k) + h_{2k}\psi(x-k)); \\ 2\phi(2x-1) &= \sum_k (g_{2k-1}\phi(x-k) + h_{2k-1}\psi(x-k) \end{aligned} \tag{3.22}$$

which is equivalent to

$$\phi(2x-l) = \frac{1}{2}\sum_k (g_{2k-l}\phi(x-k) + h_{2k-l}\psi(x-k)), \quad l \in \mathbb{Z}. \tag{3.23}$$

since the first equation yields the result for even integers and the second for odd integers, l (as a little algebra will show). Making the identification $a_n = \frac{1}{2}g_{-n}$ and $b_n = \frac{1}{2}h_{-n}$ provides us with the decomposition relation referred to in Chapter 2. The decomposition and reconstruction algorithms follow fairly simply (See Chui [10, pg. 158–159]). Details are given in [10, pg. 178–183] on the derivation of a wavelet with compact support. This process yields the following pair of equations

$$\begin{aligned} A_m(z) &= 2^{-m}(1+z)^m \frac{E_{2m-1}(z)}{zE_{2m-1}(z^2)}; \\ B_m(z) &= \frac{-(2m-1)!}{2^{-m}}(1-z)^m \frac{1}{zE_{2m-1}(z^2)}, \end{aligned} \tag{3.24}$$

where $A_m(z) \equiv \sum_k a_k^m z^{-k}$ and $B_m(z) \equiv \sum_k a_k^m z^{-k}$. The complex function $E_{2m-1}(z)$ is the *Euler-Frobenius polynomial* of order $2m-1$:

$$E_{2m-1}(z) = (2m-1)! \sum_{k=-m+1}^{m-1} N_{2m}(m+k) z^{k+m-1}. \tag{3.25}$$

The roots of the Euler-Frobenius polynomials are used to derive values for the sequences $\{a_k\}$ and $\{b_k\}$. Table 3.1 provides the first few polynomials. The roots, λ_k, are simple, real and negative and satisfy $\lambda_j \lambda_{2m-1-j} = 1$. The quotient $\frac{z^{m-1}}{E_{2m-1}(z)}$ occurs (in one guise or another) in the both the symbols $A_m(z)$ and $B_m(z)$. Hence, if one can deduce the coefficients of the complex polynomial this product induces, the values of $\{a_j^m\}_{j\in\mathbb{Z}}$ and $\{b_j^m\}_{j\in\mathbb{Z}}$ can be deduced.

That is, given the Laurent[1] series

[1] A Laurent series is a complex power series. In this context, the coefficients of this series are required to be absolutely summable.

n	$E_n(z)$
1	1
2	$1+z$
3	$1+4z+z^2$
4	$1+11z+11z^2+z^3$
5	$1+26z+66z^2+26z^3+z^4$
6	$1+57z+302z^2+302z^3+57z^4+z^5$
7	$1+120z+1191z^2+2416z^3+1191z^4+120z^5+z^6$

Table 3.1: Euler-Frobenius polynomials. **The first seven E-F polynomials. The variable $z \in \mathbb{C}$.**

$$\frac{z^{m-1}}{E_{2m-1}(z)} = \sum_{j\in\mathbb{Z}} \alpha_j^{(m)} z^j, \ |z| = 1, \tag{3.26}$$

one must find the sequence $\{\alpha_k\}_{k\in\mathbb{Z}}$. Since the quotient referred to is not a (complex) polynomial when $m \geq 2$, this sequence has infinite support for both the cubic and quadratic schemes.

In [11], the following result is derived:

Lemma 3.1 *Let m be a positive integer and $\lambda_j \equiv \lambda_j^{(m)}$, $j = 1, \ldots, 2m-2$, be the zeroes of $E_{2m-1}(z)$ arranged such that*

$$\lambda_{2m-2} < \lambda_{2m-3} < \cdots < \lambda_1 < 0.$$

If we write

$$\frac{z^{m-1}}{E_{2m-1}(z)} = \sum_{j\in\mathbb{Z}} \alpha_j^{(m)} z^j, \ |z| = 1, \tag{3.27}$$

then

$$\alpha_j \equiv \alpha_j^{(m)} = \sum_{k=1}^{m-1} \left(\frac{\lambda_k^{m-2}}{E'_{2m-1}(\lambda_k)} \right) \lambda_k^{|j|}, \ j \in \mathbb{Z}. \tag{3.28}$$

In addition, $\alpha_j = \alpha_{-j}$, for all m.

With this result, one can deduce the values of the decomposition sequences:

$\{a_k^m\}$: $a_j^m = \frac{(2m-1)!}{2} \sum_{l \in \mathbb{Z}} |q_{2l+2m-j-1}^m| \alpha_l^{(m)}$: $a_{m-j}^m = a_j^m$.

$\{b_k^m\}$: $b_j^m = \frac{(-1)^j (2m-1)!}{2^m} \sum_{l \in \mathbb{Z}} \begin{pmatrix} m \\ 2l+2m-j-1 \end{pmatrix} \alpha_l^{(m)}$, $b_{3m-2-j}^m = (-1)^m b_j^m$.

To show how these are produced, I will derive the first result; the second result is derived similarly.

$$A_m(z) = 2^{-m}(1+z)^m \frac{E_{2m-1}(z)}{zE_{2m-1}(z^2)} \tag{3.29}$$

$$= \frac{(2m-1)!}{2} \frac{Q(-z)}{z^{2m-1}} \frac{(z^2)^{m-1}}{E_{2m-1}(z^2)} \tag{3.30}$$

$$\tag{3.31}$$

Now, from Equations (3.12) (the factor of 1/2 has been included in the sequences for convenience), and the quotient discussed above, one has

$$\frac{Q(-z)}{z^{2m-1}} = \sum_j q_j^m z^{j+1-2m} (-1)^j \text{ and} \tag{3.32}$$

$$\frac{(z^2)^{m-1}}{E_{2m-1}(z^2)} = \sum_l \alpha_l z^{-2l}. \tag{3.33}$$

Taking the products indicated above, then yields

$$A_m(z) = \sum_n \sum_l \left\{ \frac{(2m-1)!}{2} |q_{2l+2m-n-1}^m| \alpha_l \right\} z^{-n}, \tag{3.34}$$

which then provides us with the sequence $\{a_k\}_{k \in \mathbb{Z}}$. The absolute value sign occurs because $(-1)^j q_j = |q_j|$. The symmetry relations are easily deduced by substitution. The decompositions sequences for the quadratic case are tabulated in Figure 3.2.

The issue of truncation error will be taken up in Section 3.4.

The sequence $\{\alpha_k\}_{k \in \mathbb{Z}}$ has further significance: it is the coefficient sequence (barring a multiplication) for the dual wavelet, hence the infinite support of this function. The dual wavelet is given by

$$\tilde{\psi}_m(x) = \frac{(-1)^{m+1}(2m-1)!}{2^{m-1}} \sum_k \alpha_k^{(m)} L_{2m}^{(m)}(2(x+m+1-k)-1), \tag{3.35}$$

where $L_{2m}^{(m)}(x)$ is the mth order derivative of the $2m$th order *interpolatory spline.* which This function is defined as

	$\{a_k\}$	$\{b_k\}$		$\{a_k\}$	$\{b_k\}$
0	0.033978977	0.049781017	12	-0.008232310	0.034166241
1	0.655340376	0.423982818	13	-0.000934671	0.003879280
2	0.655340376	-0.140377187	14	0.003544624	-0.014711266
3	0.033978977	-0.900597911	15	0.000402447	-0.001670285
4	-0.243780520	0.900597911	16	-0.001526227	0.006334313
5	-0.025936016	0.140377187	17	-0.000173284	0.000719182
6	0.103311291	-0.423982818	18	0.000657155	-0.002727399
7	0.011654634	-0.049781017	19	0.000074611	-0.000309662
8	-0.044411988	0.184116960	20	-0.000282955	0.001174351
9	-0.005039196	0.020974988	21	-0.000032126	0.000133332
10	0.019119634	-0.079343472	22	0.000121833	-0.000505646
11	0.002170658	-0.009011510			

Table 3.2: The decomposition sequences for the case $m = 3$.

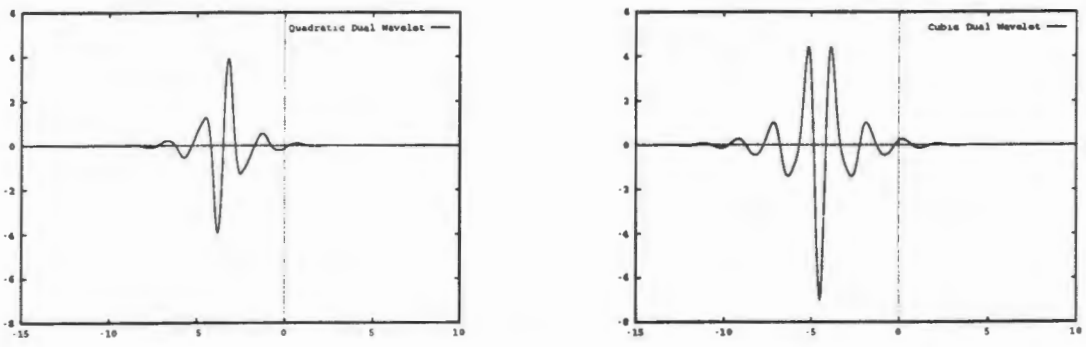


Figure 3.5: The cubic and quadratic dual wavelets. **These functions have infinite support but decay exponentially. They also possess the same symmetry as their corresponding wavelets.**

$$L_{2m}(x) = (2m-1)! \sum_k \alpha_k^{(m)} N_{2m}(x+m-k). \tag{3.36}$$

and satisfies $L_{2m}(k) = \delta_{k,0},\ k \in \mathbb{Z}$. Figure 3.5 shows the quadratic and cubic dual wavelets.

3.3 Boundary Conditions

For the wavelet transform to be usable, one must be able to ensure reversibility. This is achieved by paying careful attention to the ***boundary conditions (BCs)*** one imposes when applying the various wavelet filters — these cannot be arbitrary, but must interact in just

the right way if our transform is to be reversible. Fortunately, since the filters all possess either symmetry of anti-symmetry, one can use simple symmetric boundary extensions.

Since wavelet boundary conditions pose many problems for implementors, I have attempted to make their derivation as clear as possible. Two papers which provided important insights were those of Brislawn [4] and Unser [48]; the former went so far as to enumerate various categories of symmetric boundary conditions — but only for quadrature mirror filters, and the spline-wavelet transform does not constitute a quadrature mirror filter-bank.

3.3.1 Symmetric signal extensions

There are two major classes of signal extensions [4]:

- those with whole-sample symmetry and
- half-sample symmetry.

Whole-sample sequence extensions have their *centre of symmetry* on a integral index, while those with half-sample symmetry are symmetric about a half-integer 'index' (See Figure 3.7). It is assumed that the sequences are extended periodically using the given boundary conditions. Conceptually, one extends the sequence (using the indicated BC's) by folding it around the end-point and shifting the resulting sequence along the index axis, for as many periods as required. In practice, one uses modulo operations to map overflowing indices back into the range covered by the input signal. The modulo operations must be chosen carefully to take account of the different extension types at the end-points. Provided the BC's have been chosen correctly, this periodic extension ensures that we can deduce the correct sequence value of any index.

Definition 3.2 *The four basic sequence extensions are*

Whole-Sample Symmetric *A sequence is* whole-sample symmetric *(WSS) about an index k if $s_{k-n} = s_{k+n}$, for all n.*

Whole-Sample Anti-symmetry *A sequence is* whole-sample anti-symmetric *(WSA) about an index k if $s_{k-n} = -s_{k+n}$, for all n.*

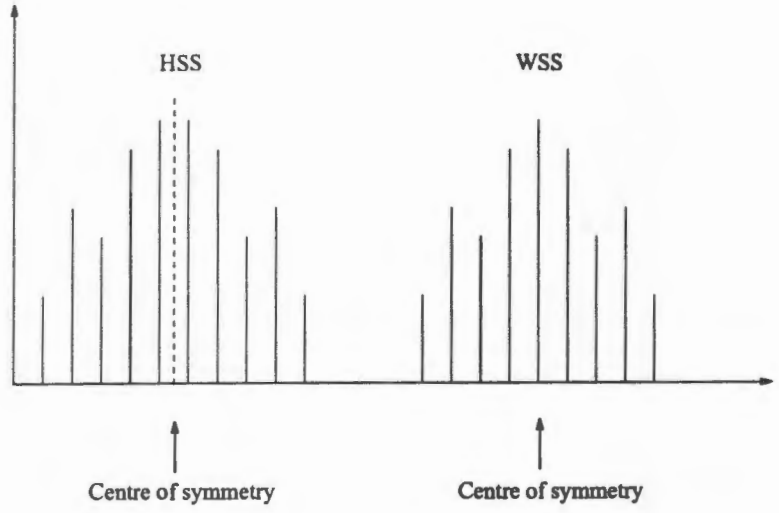


Figure 3.6: Symmetric extensions. The dashed line indicates the centre of symmetry.

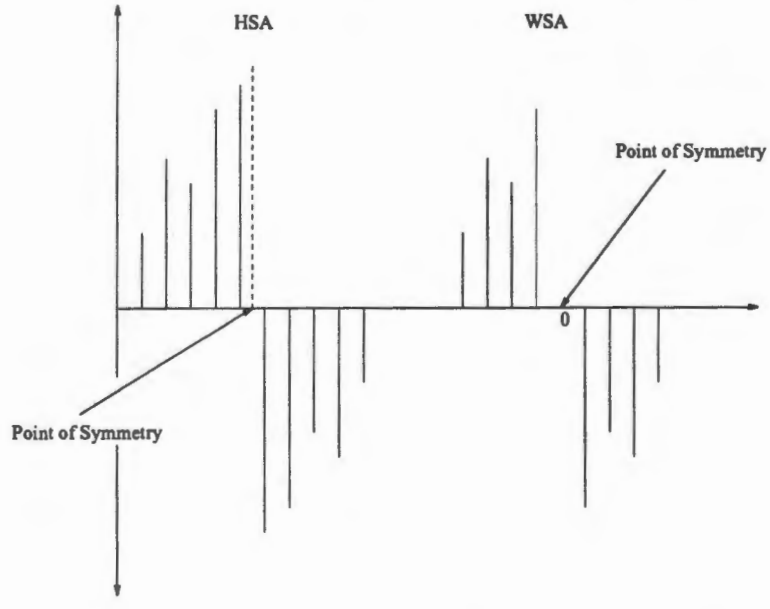


Figure 3.7: Anti-symmetric extensions. The centre of symmetry is indicated by a dashed line. Observe that WSA forces the sequence to assume the value zero at the point of symmetry. HSA enforces this zero condition at a half-sample point of symmetry i.e., an 'index' of the form $k + \frac{1}{2}$, $k \in \mathbb{Z}$.

Half-Sample Symmetry *A sequence is* half-sample symmetric *(HSS) about the 'index'* $k + \frac{1}{2}$ *if* $s_{k-n} = s_{k+1+n}$, *for all* n.

Half-Sample Anti-symmetric *A sequence is* half-sample symmetric *(HSS) about the 'index'* $k + \frac{1}{2}$ *if* $s_{k-n} = -s_{k+1+n}$, *for all* n.

Observe that WSA requires that the sequence value at the index about which the extension takes place be zero (this will have important consequences for the quadratic spline-wavelet transform).

A signal may have several points of symmetry, all with different characteristics. I only consider signals which have symmetry conditions imposed on their end-points — the existence (or not) of other symmetry points within the signal is of no consequence. Hence, the analysis will only consider the four basic types of *end-point* extension enumerated above.

Suppose that one imposes a specific symmetry on a finite signal. How does the filter (which, we will assume, possesses symmetry of its own) interact with this extension? That is, what symmetry (if any) will the resulting signal possess? To answer this question, one must know how convolution responds to the presence of symmetry.

Given a signal $\{s_k\}$ with symmetry described by $s_k = \pm s_{l-k}$ (one can describe *any* symmetry by this sort of constraint) and a filter $\{f_k\}$ which satisfies $f_k = \pm f_{t-k}$, we have

$$\begin{aligned} a_k &= (s * f)_k \\ &= \sum_n \pm s_{k-n} f_n \\ &= \sum_n \pm s_{l-k+n} f_{t-n} \\ &= \sum_{n'} \pm s_{l+t-k-n'} f_{n'} \\ &= \pm a_{l+t-k}. \end{aligned} \tag{3.37}$$

(3.38)

Thus, the centre of symmetry will be at $\frac{l \pm t}{2}$ and will be either whole-sample or half-sample in nature depending on whether $l + t$ is divisible by two or not. Using this relationship, one can enumerate the various permutations of signal/filter symmetry. The complication that arises when implementing the wavelet filter-bank scheme arise from the down- and up-sampling which occurs. This will, in general, alter the symmetry relationship, and this

must be taken into account when determining the BCs one is going to apply. The only cases I considered were those based on cubic and quadratic wavelet filters. The signal sequence may be of *arbitrary* size — it is not constrained to have a size which is a power of two, etc.

To ensure a more convenient analysis, the sequences $\{\overline{a}_k^m\}$ and $\{\overline{b}_k^m\}$ are used, where $\overline{a}_k^m = a_{-k}^m$ and so on. With this change i.e., using these sequences in place of the unbarred sequences, the decomposition algorithm corresponds precisely to filtering by $\{\overline{a}_k^m\}$ or $\{\overline{b}_k^m\}$ followed by down-sampling. No similar adjustment is required for the $\{p_k^m\}$ and $\{q_k^m\}$ sequences. The following symmetries hold, for an mth order cardinal spline scheme:

$$\begin{aligned} p_k^m &= p_{m-k}^m \\ q_k^m &= q_{3m-2-k}^m \\ \overline{a}_k^m &= \overline{a}_{-m-k}^m \\ \overline{b}_k^m &= (-1)^m \overline{b}_{-(3m-2)-k}^m \end{aligned} \tag{3.39}$$

To determine the BCs, we need only consider a one level decomposition — the procedure developed can be used for all subsequent levels. The goal is to ensure that one (or more) of the symmetry extensions holds after filtering and down-sampling. This means that one has to select an appropriate input BC. It is important to realise that some input extensions lead to *expansive* transforms, that is, the number of output coefficients is greater than the number of input samples. This is undesirable from a compression point of view. Hence, we desire an input extension which will result in a non-expansive transform.

The boundary conditions depend on the nature of the start and end indices of the input sequence i.e., whether they are even or odd. This also means that the input sequence need not start on any particular index — most schemes require that it start on index 0. The input BCs I chose are given in Table 3.3 and Table 3.4 and are presented in terms of the start and end-points of the input sequence and the type of symmetric extension required. The output values stated represent the new points of symmetry etc. after filtering and decimation. The same boundary conditions are used when decomposing for each input sequence. Because the same extension type is always used at both end-points for each input level, we are guaranteed periodic output: after down-sampling the extensions at each end-point are likely to be different, but because of the periodic input BC's (and the particular choice of input extension) the output BC's continue to be periodic.

The boundary conditions for reconstruction are simply those resulting from the previous level's filtering and decimation. They are determined by examining Table 3.3 or Table 3.4 after processing each level's input sequences. These BCs must either be stored on re-created; I decided on the latter option, since I wish to perform compression and storing unnecessary information would be counter-productive. To re-create the BCs, one just simulates the filter bank operation, without actually performing the the filtering operations — this requires very few calculations. This approach is in stark contrast to the method employed when signals are constrained to have, say, lengths that are a power of two. In this case, one can deduce the necessary BCs *beforehand* based on the length of the input signal. Nonetheless, the additional freedom one obtains when using unrestricted signal sizes more than compensates for this minor inconvenience.

The reconstruction process itself is as follows:

1. retrieve BCs for signal,
2. extend the signal using these BCs (for as many periods as desired)
3. up-sample the signal,
4. apply the appropriate reconstruction filter.

Figure 3.8 illustrates these ideas with a simple partial decomposition/reconstruction example.

3.4 Comparison of Cubic and Quadratic Cardinal Spline Schemes

The spline MRA conceived by Chui and Unser may be generated by any cardinal spline scaling function, $N_m(x)$. However, for practical and theoretical purposes, some choices of m are undesirable. For example, when we use the unit pulse, $N_1(x)$, as our scaling function, the resulting wavelet, $\psi_1 \equiv \psi_{\text{Haar}}$, does not constitute a window function, and is thus unlikely to be of great use in signal analysis. The approximation signals generated by this MRA are piecewise constant approximations to the input function (See Figure 3.4) and contain very sharp transitions. When the wavelet representation is intensively quantized, the

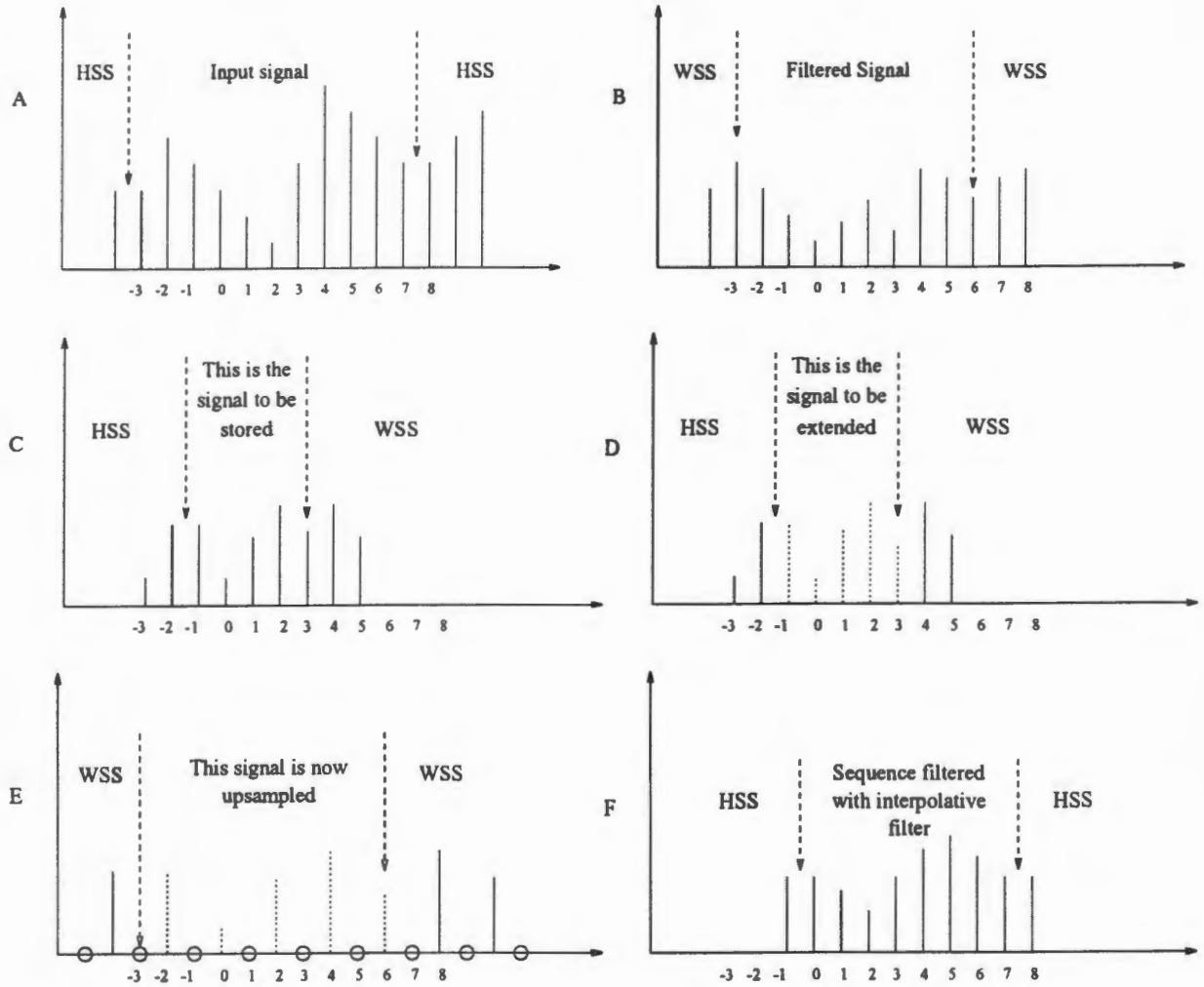


Figure 3.8: Partial decomposition and reconstruction. This example illustrates the ideas referred to above, by working through the filtering steps for the (quadratic) decomposition filter $\{a_k\}$ and the reconstruction filter $\{p_k\}$. 'A' shows the input signal and its extension (HSS, as required by Table 3.3). This signal is then filtered with $\{a_k\}$ ('B') and down-sampled ('C'). This process would yield the approximation coefficients for the next level. The next three diagrams illustrate a partial reconstruction. 'D' shows how the stored signal is extended, with the extension it had after the decomposition phase. This extended signal is then up-sampled ('E') and finally filtered with the 'interpolative' filter, $\{p_k\}$ ('F'). To complete the process, one would extract the stored detail coefficients, extend, up-sample and filter them, in a similar manner, before adding the two resulting signals together.

	Case 1	Case 2	Case 3	case 4
Start (s)	Even	Even	Odd	Odd
End (e)	Even	Odd	Even	Odd
Ext Start (in)	HSS	HSS	HSS	HSS
C.O.S Start (in)	$s-1/2$	$s-1/2$	$s-1/2$	$s-1/2$
Ext End (in)	HSS	HSS	HSS	HSS
C.O.S End (in)	$e+1/2$	$e+1/2$	$e+1/2$	$e+1/2$
Ext Start (out)	WSS	WSS	HSS	HSS
C.O.S Start (out)	$\frac{s-2}{2}$	$\frac{s-2}{2}$	$\frac{s-1}{2}-1/2$	$\frac{s-1}{2}-1/2$
Ext End (out)	WSS	HSS	WSS	HSS
C.O.S End (out)	$\frac{e-2}{2}$	$\frac{e-3}{2}+1/2$	$\frac{e-2}{2}$	$\frac{e-3}{2}+1/2$
Ext Start (in)	HSS	HSS	HSS	HSS
C.O.S Start (in)	$s-1/2$	$s-1/2$	$s-1/2$	$s-1/2$
Ext End (in)	HSS	HSS	HSS	HSS
C.O.S. End (in)	$e+1/2$	$e+1/2$	$e+1/2$	$e+1/2$
Ext start (out)	HSA	HSA	WSA	WSA
C.O.S. Start (out)	$\frac{s-4}{2}-1/2$	$\frac{s-4}{2}-1/2$	$\frac{s-5}{2}$	$\frac{s-5}{2}$
Ext End (out)	HSA	WSA	HSA	WSA
C.O.S End (out)	$\frac{e-6}{2}+1/2$	$\frac{e-5}{2}$	$\frac{e-6}{2}+1/2$	$\frac{e-5}{2}$

Table 3.3: Quadratic Boundary Conditions. The First block provides the boundary conditions for input and output sequences when filtering with $\{\overline{a}_k^3\}$; the second block provides the same information for $\{\overline{b}_k^3\}$. The abbreviation C.O.S stands for Centre Of Symmetry, while Ext denotes extension. Those conditions applying to input are qualified with (in) and those associated with output with (out).

	Case 1	Case 2	Case 3	case 4
Start (s)	Even	Even	Odd	Odd
End (e)	Even	Odd	Even	Odd
Ext Start (in)	WSS	WSS	WSS	WSS
C.O.S Start (in)	s	s	s	s
Ext End (in)	WSS	WSS	WSS	WSS
C.O.S End (in)	e	e	e	e
Ext Start (out)	WSS	WSS	HSS	HSS
C.O.S Start (out)	$\frac{s-2}{2}$	$\frac{s-2}{2}$	$\frac{s-1}{2} - 1/2$	$\frac{s-1}{2} - 1/2$
Ext End (out)	WSS	HSS	WSS	HSS
C.O.S End (out)	$\frac{e-2}{2}$	$\frac{e-3}{2} + 1/2$	$\frac{e-2}{2}$	$\frac{e-3}{2} + 1/2$
Ext Start (in)	WSS	WSS	WSS	WSS
C.O.S Start (in)	s	s	s	s
Ext End (in)	WSS	WSS	WSS	WSS
C.O.S. End (in)	e	e	e	e
Ext Start (out)	HSS	HSS	WSS	WSS
C.O.S Start (out)	$\frac{s-4}{2} - 1/2$	$\frac{s-4}{2} - 1/2$	$\frac{s-5}{2}$	$\frac{s-5}{2}$
Ext End (out)	HSS	WSS	HSS	WSS
C.O.S. End (out)	$\frac{e-6}{2} + 1/2$	$\frac{e-5}{2}$	$\frac{e-6}{2} + 1/2$	$\frac{e-5}{2}$

Table 3.4: Cubic Boundary Conditions. **The First block provides the boundary conditions on input and output sequences when filtering with $\{\overline{a}_k^4\}$, while the second provides this information for the filter $\{\overline{b}_k^4\}$.**

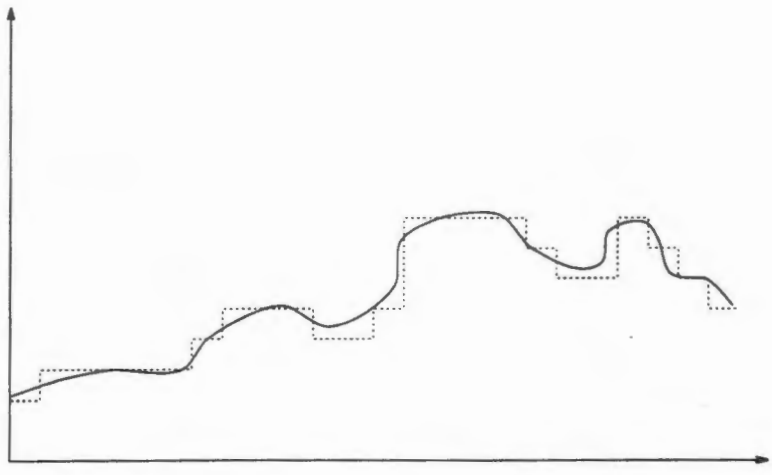


Figure 3.9: Approximation in the Haar MRA. The MRA generated by $N_1(x)$ uses piece-wise constant (0-degree) polynomials to approximate functions. Unless the resolution is high, these approximations are unacceptably blocky.

distortions introduced on reconstruction are likely to possess sharp boundaries, and as such this scheme would be unsatisfactory in the context of image compression. However, even MRA's which possess a higher degree of inherent smoothness may be unacceptable. While a high order polynomial has more degrees of freedom and is thus able to produce 'better' approximation signals than a lower order scheme, this comes at greater computational expense: the wavelet and scaling function dilate and this results in longer reconstruction sequences and, consequently, slower signal reconstruction. Thus, one would like to strike a balance between smoothness and reconstruction time. The scaling function $N_2(x)$ is certainly preferable to the unit pulse; however, it has only zeroth order continuity — one would prefer at least C^1. The quadratic scaling function, $N_3(x)$ is next in line; while the MRA generated by this function does indeed have the requisite smoothness, there are certain theoretical problems which cause implementors to pass this scheme over in favour one based on cubic splines.

In the spline formalism, schemes with odd order only possess generalized linear phase rather than linear phase. Since linear phase filters are less likely to magnify quantization errors, one would certainly prefer such a MRA. Nonetheless, this does not constitute a strong reason for dismissing the quadratic MRA — others approaches, such as Daubechies compact orthogonal scheme, do not possess linear or even generalized linear phase and yet they function more than adequately. A more serious problem is posed by the inability to generate a 'nice' interpolation using a quadratic spline basis. By this I mean that the quadratic-spline

interpolation

$$I(x,y) = \sum_k \sum_l c^0_{kl} N_3(x-k) N_3(y-k), \tag{3.40}$$

is 'ill-posed' on $\mathbb{Z} \times \mathbb{Z}$, in the sense that some of the data which one would require to proceed is not available. This is most unsatisfactory, since this is precisely the form we need to start the decomposition algorithm. Initially, it seemed that this problem would render the whole quadratic approach infeasible. Fortunately, I discovered a way to circumvent this problem, involving the use of a *shifted* multi-resolution analysis — see Section 3.5. Assuming, then, that one can produce such a suitable initial interpolation, why should we choose quadratics over cubics?

Beyond the obvious speed gain (resulting from a decrease in reconstruction filter length), there is another reason. If one truncates the decomposition sequences ($\{a_k\}$ and $\{b_k\}$), the approach adopted in this work[2], then it turns out, that the quadratic filters can bear fiercer truncation than their cubic counterparts. Only after careful examination of the filters did the reason become apparent. The decomposition filters must, not unexpectedly, obey certain pass-band conditions if they are to filter out the right information. It turns out [10] that the following conditions must hold for these filters

$$\begin{aligned} \sum_k a^m_k &= 1; \\ \sum_k b^m_k &= 0. \end{aligned} \tag{3.41}$$

If these conditions are not (approximately) satisfied, the filters will not function correctly and they will pass frequencies outside of their intended pass-band. Examination of the decomposition filter symmetries shows that the cubic detail filter, $\{b^4_k\}$, possesses a whole-sample centre of symmetry, whilst the quadratic detail filter possesses a half-sample centre of symmetry and is anti-symmetric. While this seems unremarkable, if one progressively takes fewer terms from each of these sequences, the cubic filter fails its pass-band condition much more rapidly than the quadratic filter, for comparative levels of truncation (See Table 3.5). The reason for this failure lies with the nature of the symmetries these two filters possess. The quadratic filter has the structure $(\ldots, -a, -b, -c, c, b, a, \ldots)$ while the cubic filter has

[2]The alternative is to truncate the $\{\alpha_k\}$ sequence which generates the decomposition sequences, but the benefit of doing it this way is not immediately obvious.

	$\#a$	$\#b$	$O(\sum a)$	$O(\sum b)$
Quad	26	26	10^{-3}	10^{-4}
Cubic	27	27	10^{-2}	10^{-3}
Quad	20	20	10^{-2}	10^{-4}
Cubic	21	21	10^{-2}	10^{-2}
Quad	18	18	10^{-3}	10^{-4}
Cubic	19	19	10^{-2}	10^{-2}
Quad	16	16	10^{-2}	10^{-4}
Cubic	17	17	10^{-2}	10^{-1}
Quad	14	14	10^{-2}	10^{-4}
Cubic	15	15	10^{-2}	10^{-2}
Quad	12	12	10^{-2}	10^{-4}
Cubic	13	13	10^{-2}	10^{-1}

Table 3.5: Failure of band-pass conditions. **The left-most two columns indicate the number of a, b coefficients maintained after truncation. The final two columns indicate the order of magnitude of the error to within which the sequences approach their band-pass conditions, Equations (3.41).**

the has the structure $(\ldots, a, b, c, b, a, \ldots)$. In the former case, provided one truncates about the centre of symmetry, one is guaranteed that the sum will be close to zero — because of the negative signs. However, in the cubic case this is not so, and at low truncations, there are no longer sufficient terms to (nearly) satisfy this condition. Figure 3.10 show the effect of this failure: the incorrectly registered frequencies add ripples to the signal. I experimented with the cubic filter, tweaking the trailing sequence values in the truncated filter until the condition was approximately satisfied and, as expected, the ripples disappeared. One cannot, however, apply such arbitrary sequence manipulations and still expect to get the correct results — modifying the filters will introduce distortions of some kind, the magnitude and type of this noise depending on the number of coefficients which are modified. This lack of robustness on the part of the cubic spline wavelet scheme has evinced surprisingly little comment in the literature.

Another reason for not choosing the cubic scheme arises from the form of interpolation I employ — *quasi-interpolation*. Section 3.5 reveals a somewhat startling result: quadratic quasi-interpolation produces a better fit than cubic quasi-interpolation! The details are given in that section.

There does not seem to be an overwhelming reason to use cubics. Indeed, the quadratic

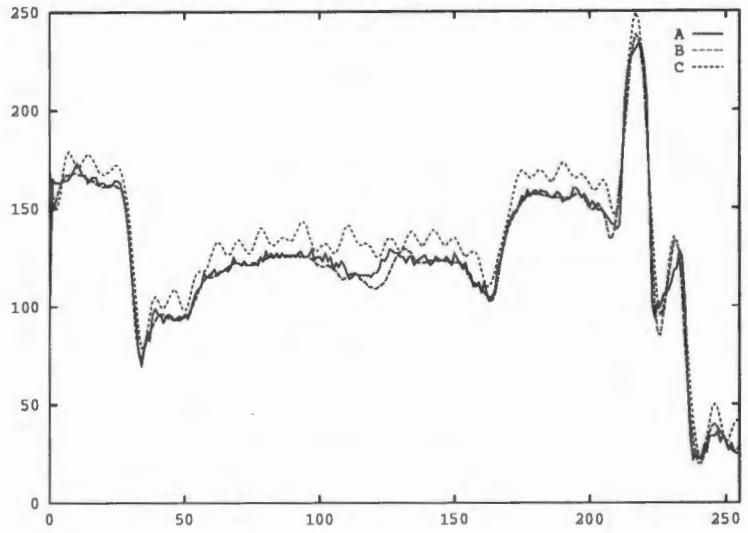


Figure 3.10: Failure of the cubic filters at low truncations. 'A' gives the input data, 'B' the 2nd resolution level cubic decomposition approximating the image and 'C' the reconstruction to level 2 after decomposing with the over truncated cubic decomposition sequences. When the cubic filters are not over truncated, they result in a reconstruction which has these sinusoidal ripples smoothed out. Note: the reconstruction sequences are *never* truncated.

scheme seems to offer some advantages. The use of quadratic curves has additional computational advantages beyond the obvious, directly related to our hardware platform, and will only become apparent once the synthesis algorithms have been discussed. This is deferred until Chapter 5

3.5 Calculation of Initial Approximation Coefficients

Before any of the algorithms referred to above can be implemented, one must produce the initial set of input coefficients, $\{c_{ij}^0\}$. Some authors use the input pixel values for the 2-D sequence. This is perfectly acceptable if one only wishes to use the MRA coefficient sequences for something like image compression. However, if one wishes to compute the various approximation and detail images, as I do, then one cannot assign the input coefficients in this manner. Indeed, these coefficients must be chosen such that they provide the correct weights for the basis elements $\phi_{j,k}$, which we then sum to return the values of our input pixel values (which are taken to lie on $\mathbb{Z} \times \mathbb{Z}$, as discussed previously). The question now

arises as to how one obtains these coefficients, given that we do not wish to use any sort of integral formulation.

We would like to

1. construct a surface which interpolates all the pixel values and
2. determine the expansion coefficients which describe this surface.

So, we need to find the 2-D spline interpolant which passes through our (discrete) pixel data:

$$\sum_{i,k\in\mathbb{Z}} c_{ik}^0 \Phi(l-i, m-k) = I(l,m)|_{X\times Y}, \quad l,m \in \mathbb{Z}, \tag{3.42}$$

where $X \times Y$ is the domain of our image.

This is equivalent to inverting a large (sparse) matrix and as such can be fairly expensive. An cheaper alternative exists, however, if one is prepared to weaken the interpolatory requirement. That is, rather than requiring true interpolation, one settles for *quasi-interpolation*, in which the input data may be arbitrarily closely approximated, the accuracy of the fit determining the computational load.

In order that we may use a quasi-interpolant we require that the function we are interpolating by bounded and continuous. This does not present a problem: the image data is certainly bounded and our pixel values may be considered a sampling, on $\mathbb{Z} \times \mathbb{Z}$, of a continuous image function, $I(x,y)$.

Definition 3.3 *If a 2-D, symmetric, origin-centred piece-wise polynomial function* $\Phi(x,y)$ *satisfies the* Fix-Strang *conditions [9]*

1. $\hat{\Phi}(0,0) = 1$;
2. $D^\alpha \hat{\Phi}(2\pi i, 2\pi j) = 0, \quad 0 \neq \quad i,j \in \mathbb{Z}, |\alpha| \leq \rho \ \ (\rho \geq \kappa)$,

then one may define the kth order quasi-interpolant, $(Q_k I)(x,y), \quad k \in \mathbb{Z}^+$, *of* $I \in C^s(\mathbb{R}^\kappa)$ *as*

$$(Q_k I)(x,y) \equiv \sum_l \sum_m (\lambda_k I)(l,m) \Phi(x-l, y-m). \tag{3.43}$$

The convolutional operator λ_k operates on the input image-sequence, $I^0(l,m)$, $l,m \in \mathbb{Z}$, and is defined by

$$\{(\lambda_k I)(i)\} = (\delta - m + \cdots + (-1)^k \underbrace{m * \cdots * m}_{k \text{ times}}) * I^0(i), i \in \mathbb{Z}^\kappa. \tag{3.44}$$

where $\delta \equiv \delta_{i,j;0} = 1$ if i, j = 0. and 0 otherwise and

$$m_{i,j} = \begin{cases} \Phi(0.0) - 1 & \text{for i, j} = 0; \\ \Phi(i,j) & \text{for i, j} \neq 0. \end{cases} \tag{3.45}$$

This quasi-interpolant has the following properties:

- only local data (the extent of which is determined by the parameter k) is used to determine the values of the sequence $\{(\lambda_k I)(i.j)\}$
- any polynomial of degree $\rho \leq 2k+1$ will be reproduced by this scheme
- the sequence of operators Q_k converges to a true interpolation operator, Q_∞, as $k \to \infty$; i.e., $(Q_\infty I - I)(l,m) = 0$, $i,m \in \mathbb{Z}$.

The Fix-Strang conditions are satisfied by the cardinal splines $N_m(x)$ and consequently also by their tensor products. To use the above scheme, we must recast it in our framework. In this case

$$\Phi(x,y) \equiv N_m(x + \frac{m}{2}) N_m(y + \frac{m}{2}) \tag{3.46}$$

where the shift is required to centre the cardinal B-spline functions. Equation (3.43) then looks like

$$(Q_k I)(x,y) = \sum_l \sum_n (\lambda_k I)(l,n) N_m(x + \frac{m}{2} - l) N_m(y + \frac{m}{2} - n). \tag{3.47}$$

What we desire is that $(Q_k I)(x,y) = I^0(x,y) \equiv \sum_l \sum_m c^0_{ln} N_m(x-l) N_m(y-m)$. From this we can see that, barring the shift in the arguments, the input approximation coefficients correspond to the lambda sequences. Of course, one cannot simply disregard the shift — it forms an integral part of the equation. If we are dealing with a scheme for which $\frac{m}{2}$ is an integer, such as the cubic ($m = 4$) scheme, one can apply a simple change of variable and include the shift in the lambda sequences indices:

Quadratic Case	Mean	Standard Deviation	|Max Error|
$k=1$	0.00	1.69	21
$k=2$	0.01	0.90	10
Cubic Case			
$k=1$	0.01	2.74	34
$k=2$	0.00	1.79	21

Table 3.6: The error induced by quasi-interpolation of our test image. The quadratic scheme ensures both a lower projection error and a lower maximum error. The benefit of using a higher order quasi-interpolation is clear: even $k=2$ provides a considerable gain over $k=1$.

$$(Q_k I)(x,y) = \sum_l \sum_m (\lambda_k I)(l+\frac{m}{2}, m+\frac{m}{2}) N_m(x-l) N_m(y-l). \tag{3.48}$$

We may then make the identification: $c_{ij}^0 = (\lambda_k I)(i+\frac{m}{2}, j+\frac{m}{2})$.

However, if the shift is *not* integral, as is the case for the quadratic scheme ($m=3$), then one cannot do this. The question is, given our desire to use the quadratic scheme, how do we get around this? I used the following approach. Since $(Q_k I)(x,y) = I^0(x+\frac{m}{2}, y+\frac{m}{2})$, we may still match the coefficients, *provided* that we remember that what we are now dealing with is a *shifted* version of the input image. This means that all our subsequent approximation images will also be shifted; in fact, our filtering scheme will now generate the coefficients for our shifted detail and approximation images. Nonetheless, by evaluating the functions with a negative shift added to the arguments, we can compute values as we normally would.

In [10], Chui derives an interesting relationship which quantifies the interpolation error which arises when using a quasi-interpolant rather than a true interpolant. After a few algebraic manipulations, one can deduce the following [10, pg. 113–114]:

$$\max_{l\in\mathbb{Z}} |(Q_k f - J_m f)(l)| \leq (\max_{l\in\mathbb{Z}} f(l) + \min_{l\in\mathbb{Z}} f(l)) \frac{1}{2} \beta_m^{k+1}, \quad \beta_3 = \frac{1}{2} \text{ and } \beta_4 = \frac{2}{3}, \tag{3.49}$$

where the sequence $\{f_k\}_{k\in\mathbb{Z}}$ is bounded.

The function $(J_m f)(x)$ is an mth order true interpolant i.e., it satisfies the conditions $(J_m f)(l) = f(l)$, $l \in \mathbb{Z}$. This shows that the maximum quasi-interpolation error is lower when we use quadratics than when we employ cubics as our underlying spline. One should bear in mind, however, that this estimate is only valid at the integer knot-points, and consequently cannot be used to infer that quadratic quasi-interpolation in globally 'better'

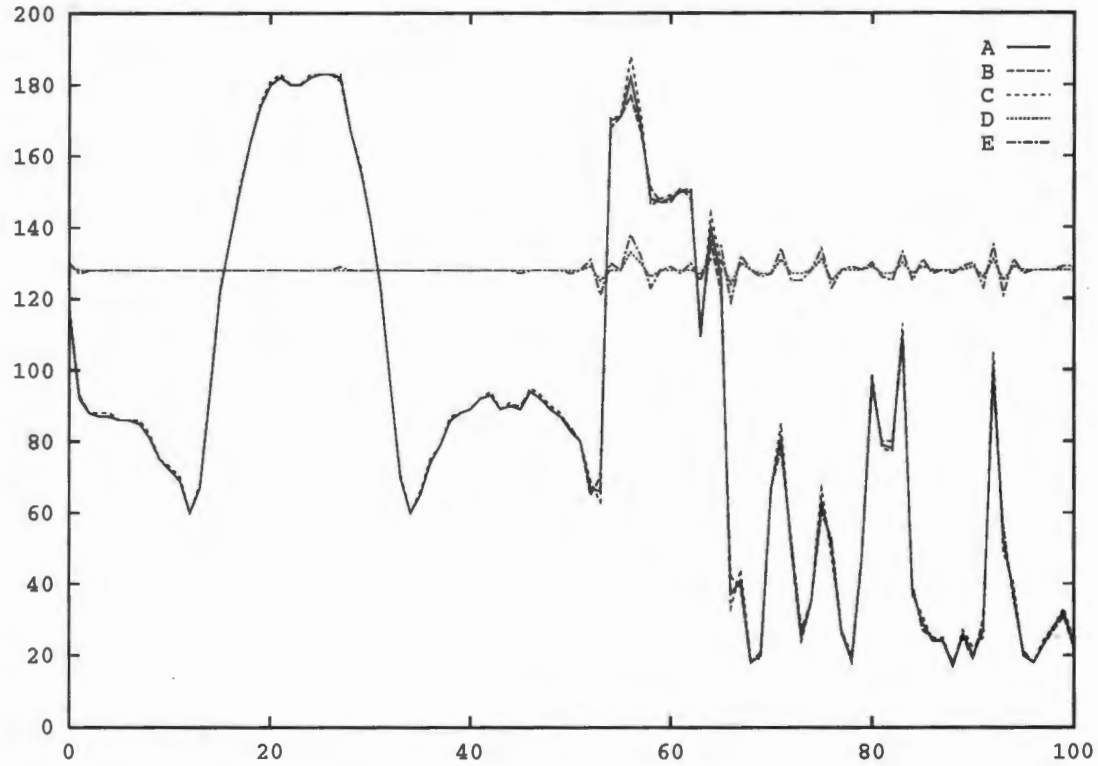


Figure 3.11: Quasi-interpolation Error Effects. The interpolation error is biased by 128. 'A' gives the quadratic quasi-interpolation ($k = 2$) of the scan-line, 'B' the cubic interpolation ($k = 2$). The graph 'C' gives the input data. Graph's 'D' and 'E' give the interpolation error for the quadratic and cubic cases, respectively. Observe that the interpolation error for the cubic scheme is greater than that of the quadratic scheme for the same k.

than cubic quasi-interpolation. Table 3.6 provides some results which quantify this claim while Figure 3.11 provides a graph which illustrates both these interpolants. Even for low order quasi-interpolation, the interpolation error is small. In fact, for quadratic quasi-interpolation, this error was less than one grey-scale value (for our test image) when $k = 2$. Since this image is typical of the sort one would normally encounter in image processing, there seems to be no need to employ higher order quasi-interpolation.

3.6 Concluding Remarks

Although the cardinal spline MRA has dual functions with infinite support, it possesses a host of properties which counterbalance this undesirable aspect. For example, the wavelet, scaling function and their duals all possess either linear or generalized linear phase — a property which is absent from non-trivial orthogonal implementations. There are also simple (analytical) calculational procedures to derive the decomposition/reconstruction filters as well as the wavelet and scaling function.

For computational and implementational reasons one desires that the scaling function be smooth and of low order. Ideally one would like to employ the quadratic B-spline — however, theoretical problems usually ensure that implementors chooses the cubic scheme. Nonetheless, Section 3.5 showed that these theoretical objections can be circumvented by employing a shifted multi-resolution analysis and adjusting all MRA computations accordingly i.e., adding the correct offsets to image function evaluations.

Having selected a scaling function, one must produce the initial set of approximation coefficients, which amounts to solving a large system of linear equations. The computational overhead may be reduced by using *quasi-interpolation*, rather than true interpolation. Although this scheme results in only approximate interpolation, it only uses local data to compute each coefficient. The quasi-interpolant is parameterized to allow control over the locality of the data used; one may obtain arbitrarily close approximation by increasing this parameter. Low order quasi-interpolation is shown to be adequate for our purposes. Furthermore, quadratic quasi-interpolation is shown to produce lower interpolation error than cubic quasi-interpolation at the integer knot-points (pixels). This additional benefit, when coupled with the others already discussed or alluded to, implies that a quadratic based spline MRA is the ideal choice.

The preceding chapters have established the basis for our image compression and synthesis algorithms. With the theoretical framework established, we may now address these issues.

Chapter 4

Spline-Wavelet Image Compression

The study of digital image compression techniques, whether for still or video images, is an area of research which enjoys a particularly high profile. The advent of the information society and the emergence of multi-media applications, which often include high band-width video services, have resulted in a dramatic increase in the amount of data which computer systems have to process and store. Image data (in its raw form) requires a large amount of storage, in the region of 3Mb for a full colour (24-bit) 1024x1024 image. When one wishes to store a large number of such images, the restrictions imposed by limited resources become acute. Image compression techniques seek to alleviate this problem, by changing the image representation so that the new form occupies less space.

This chapter begins by considering the most common methods of achieving image compression, before moving on to consider the efficacy of the wavelet transform as an image coding scheme. Section 4.3 introduces the concept of quantization and describes the type (vector quantization) which was used in this dissertation. The remaining sections provide an analysis of the compression results (4.5) and briefly discuss an alternative that one might employ to better exploit the properties of the wavelet transform (4.6).

4.1 Compression Strategies and Standards

The nature of the data to be compressed plays a fundamental role in determining the compression ratios[1] one may achieve. If the data must be precisely reproduced, that is, the compression must be *lossless*, then the compression ratio will seldom exceed 3:1. Text compression is an example of a source which must be losslessly compressed.

If, however, one need not reproduce the input precisely, but only a suitably 'close' approximation, then one may achieve significantly higher compression ratios (> 10:1). Image and audio streams provide examples of sources which can benefit from such *lossy* compression techniques. Such methods exploit the inherent redundancy present in the source; for example, the correlation between neighbouring pixels in an image. There is much debate concerning the means of determining the optimal trade-off between compression and fidelity, that is, how far one can compress and still have sufficient data to reconstruct an acceptable approximation. In particular, the choice of a fidelity metric is highly contentious; for ease of computation and mathematical simplicity one usually seeks to minimize the mean-squared error, implicitly assuming that if the difference between the original and the reconstruction is small under this metric, that the human visual system (HVS) will likewise find the artifacts induced by the compression to be acceptable. This is not, unfortunately, the case since HVS is an extremely complex system and cannot be so easily quantified. An interesting case is made by De Vore et al [15] for the use of the L^1 norm as the distortion measure which should be minimized. These issues will be touched upon in this chapter as well as those subsequent to it.

How does one remove the redundant information contained in an image? There are several accepted techniques, some simple (and less effective) and others fairly complex. On the simple side of the spectrum, one has Delta Modulation (DM) [26]. Here one encodes the difference between neighbouring pixels. Since the pixels intensities are (usually) highly correlated, one can make do with a much smaller dynamic range for the differences; consequently only a small number of bits are needed to encode this data. Unfortunately, if the image data varies too dramatically, one is faced with *slope overload*: the intensity differences exceed the allocated range and the coding scheme no longer accurately represents the input. There are more sophisticated coding techniques, such as Differential Pulse Coded Modulation (DPCM), which are based on similar principles, but the compression ratios one

[1] That is, the ratio $\frac{\text{input size}}{\text{output size}}$.

can achieve using these methods are fundamentally limited, and are scarcely better than those available when using lossless[2] techniques.

Transform Coding [26] constitutes a far more robust and powerful method of achieving lossy compression. The image is acted upon by an operator (the transform) which serves to decorrelate the input pixel values. The new image, which only encodes non-redundant information and thus constitutes a more compact representation, is examined and only relevant pixel values (the transform coefficients) are retained. Reconstruction is achieved via an inverse transform. The success of such a method relies upon the efficacy of the transform, i.e.,

- how well does the transform de-correlate the input and
- are the important (large magnitude) transformed pixel values clustered in the new image?

The de-correlation manifests itself as a decrease in the magnitude of pixel values, in all but a small region of the image. That is, most of the transformed pixel values are near zero, and those which are not, encode the non-redundant information content of the image. These near-zero regions may be discarded with minimal distortion in the reconstruction.

The clustering of the pixel values one decides to encode, determines the simplicity with which such an encoding may be achieved. Clearly, if these values always lie within a particular region, one can develop simple coding strategies to encode their position. If, on the other hand, the values to be stored are scattered randomly over the entire image, then one must encode positional information explicitly and hence suffer a loss in compressibility.

4.1.1 The JPEG[3] Compression Standard

Having said all this, which transformations are used to effect de-correlation? The current still image compression standard (JPEG) utilizes the Discrete Cosine Transform (DCT) — a derivative of the Discrete Fourier Transform. The new pixel values specify the relative strengths of the two-dimensional sinusoids which compose the input image. For most images, these coefficients are clustered in a narrow region around the x and y axes. This

[2]DM could be considered a lossless coding method, provided one guarantees that slope overload will not occur. DPCM uses predictive encoding and this element means that it constitutes a true lossless scheme.

[3]Joint Photographic Experts Group

enables a simple *zoning* strategy to be used for pixel selection: one always assumes the the high-magnitude coefficients lie within a prescribed region. Zonal coding can be less than satisfactory, since it will not code large coefficients which fall beyond its boundaries. A further problem with the DCT used in JPEG, arises from its block implementation. In order to accelerate the transform computations, the image is decomposed into small 8x8 blocks and each block is then coded i.e., treated like a separate image. Unfortunately, at high compression ratios the assumption of independence amongst the blocks fails and so-called blocking effects occur i.e., noticeable intensity discontinuities between blocks. These can be treated, to a limited extent, by post-processing, but this adds to the effective decompression time and counteracts the computational savings obtained by blocking. Nonetheless, JPEG remains the method of choice for most implementors and is endorsed by ISO. This method produces good results up to about 16:1 [49].

4.1.2 Fractal Image Compression

A new arrival on the compression scene is ***fractal compression.*** This technique uses the mathematics of iterated function systems (IFS) [2, 16] to compute a compact representation of the image. The IFS approach describes the image as a sequence of 2-D affine (linear) maps. When these maps are iteratively applied to an arbitrary input, the fixed point to which they converge (the 'attractor') provides an approximation to the input image. Since only the parameters which characterize this set of maps need be stored, this representation can achieve fairly high compression ratios — around 30:1 — without serious degradation.

The word 'fractal' arise because the attractor (the decoded image) is self-similar at every scale. Of course, digital images (and consequently, the input image) are *not* self-similar at every scale, but this aspect is only revealed when one expands the image: the IFS produces additional self-similar 'detail'. This detail is essentially fractal noise; one cannot create detail the image did not originally possess. The usefulness of this zoom facility of the IFS encoding is thus questionable, since one usually desires to investigate a region more thoroughly when performing this action, not to see a fuzzy (albeit cunningly interpolated) blow-up.

The major problem confronting this approach is the excessive time required to produce the IFS maps — several minutes in the case of large images. Decompression time, however, is very rapid — around a few seconds for the same class of images. The issue of compression

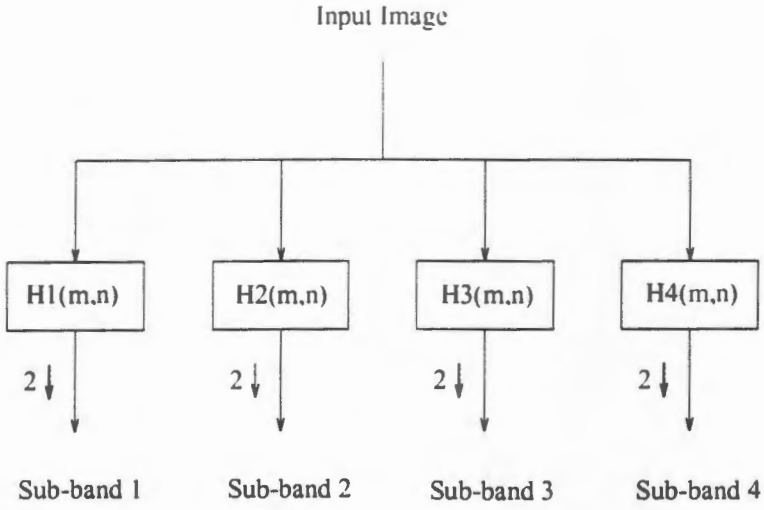


Figure 4.1: An example of 4 sub-band decomposition. **Each box represents a 2-D filtering of the input (from the previous level) with the indicated filter. The symbol 2 ↓ is used to denote 2x2 down-sampling of the filter output.**

artifacts is also somewhat contentious. Adherents of this approach claim that the blotchy patches introduced at high compression ratios are preferable to blocking artifacts. This is a highly subjective issue and currently one cannot render a meaningful objective decision as to which is better. Of course, constant advances are occurring and there are attempts underway to eliminate (or at least ameliorate) these problems.

4.1.3 Sub-Band Coding schemes

Another popular class of methods for achieving compression are *sub-band filtering* schemes. Such systems apply a series of different filters to the input image and down-sample the output, i.e., keep every alternate sample. See Figure 4.1. This decomposes the image into a number of sub-bands which contain the frequency information filtered out by the corresponding filter. The method may be applied iteratively (i.e., cascaded) to produce additional decimated sub-bands, provided that some suitable terminating criterion is provided. The size of the input signal is preserved by such a decomposition. Of course, these filters cannot be chosen arbitrarily and there should be a corresponding set of filters which enable one to perform the reconstruction. This reconstruction starts with the last level of sub-band coefficients and proceeds upwards, up-sampling and adding each level to reproduce the input.

If one wishes to obtain compression, the filters must chosen to ensure that the input image is de-correlated by the filtering operations. This de-correlation is reflected in sparsely populated sub-bands which may then be compactly encoded. Such an image coding scheme was proposed by Woods and O'Neil in [52]. They achieved compression ratios of around 15:1 with good reproduction. These results do not imply that one cannot do better than this using such a scheme; indeed, the wavelet transform may be recast as a sub-band decomposition scheme and the results obtained in this case may be substantially higher [1].

4.1.4 The Laplacian Pyramid

As with all compression schemes, the Laplacian pyramid [5] seeks to eliminate the redundancy inherent in the input (image). This is achieved by a series of filtering and decimation operations, in much the same manner as sub-band coding. However, unlike the latter, the Laplacian pyramid is an implicitly redundant representation since it contains more pixels than the original image by a factor of $\frac{4}{3}$. The decomposition proceeds as follows:

1. The image is filtered with a low-pass filter, producing a less detailed approximation.
2. this approximation is the subtracted from the original and the difference is retained.
3. the approximation is then down-sampled and the low-pass filter applied once more.
4. the new difference image (produced as before) is stored and the process repeated, until the detail left in the down-sampled approximation is sufficiently uniform.
5. this low-resolution, down-sampled image is then stored, along with the difference images.

The information contained in each difference signal is highly localised and concentrated, since only the application of a low-pass filter differentiated the images we subtracted. Thus, one need only encode a small subset of the information in each image as well as the small low-pass image, when contains contrast and intensity information. So, even though it would seem that one is increasing the redundancy, the de-correlating effect of the construction ensures that one experiences a net gain. As with basic sub-band coding, good results are obtained for compression ratios up to about 15:1.

Figure 4.2: Quadratic spline wavelet transform. **The first image provides the input image, while the second provides a 3 level quadratic spline wavelet transform. Observe that the coefficient images have different sizes; in the cubic scheme the sub-bands on a particular level have the same dimensions.**

4.2 Wavelet Image compression

The wavelet transform may be implemented as a sub-band decomposition [29, 13]; in this approach one need only compute simple convolutions rather than evaluating integrals. This formulation was presented in the preceding chapter, although little was said there concerning the application of the wavelet transform to image coding. Figure 4.2 provides an image as well as its representation in the transform domain. This representation comprises a sequence of detail coefficient images (three for each level of the decomposition) and a low-pass down-sampled approximation coefficient image:

$$\{\{d_{p_{ij}}^{l}\}, \{c_{ij}^{-L}\}, \quad l = -1, -2, ... - L; \quad p = 1, 2, 3\} \tag{4.1}$$

The supports of these 2-D sequences shrink with lower resolution; consequently, one requires fewer coefficients to represent lower resolution approximation and detail images. The total number of coefficients is the same as the number of input pixels.

The wavelet transform detects detail i.e., sharp intensity variations, such as those associated with the transition across an edge. Since texture may be viewed as a collection of edges on an appropriately small scale [34], the WT is also adept at describing texture. The (detail)

coefficients provide a measure of the strength of the variation. Smooth image regions are reflected in the approximation coefficients. The multi-scale aspect of the decomposition allows texture/edge information at different resolutions to be discriminated. One generally discards very high frequency texture information in favour of higher compression ratios. The approximation coefficient sequence $\{c_{ij}^J\}$ is normally stored using the full range of bits that the original image possessed — 8 bits, in most case, which corresponds to 256 grey levels, while the detail coefficients are quantized much more coarsely.

The highest-resolution level (which has not been smoothed) will have a large proportion of non-zero detail coefficients, but many of these may be ignored since they correspond to wavelets with small support and the error induced by their loss will be insignificant. In fact, one may go so far as to ignore the whole first level detail tier (thus immediately eliminating 75% of the information in the representation) — the induced error manifests itself as a slight blurring of high frequency texture.

Most images produce detail coefficients which are clustered around prominent edges, the remaining coefficients are considerably smaller and, for the most part, may be discarded. It is this detail coefficient sparseness which endows the WT with a high compression potential.

The sub-band structure of the WT means that the transform has a complexity of $O(n)$ The (unblocked) DCT has $O(n \log n)$ complexity — blocking decreases the overall complexity to $O(n)$. There are also means of making the inherently fast wavelet calculations faster, although these methods are dependent on the particular characteristics of the filters [41]. Because of the efficiency with which the WT can be performed, one need not resort to blocking the transform, and consequently blocking artifacts do not occur. Comparative studies (for example, [14]) have shown that the DWT is also able to retain structure and reconstructive fidelity at much higher compression ratios than the DCT.

The positions and magnitudes of the coefficients may be encoded in various ways. An obvious candidate is *run-length coding*, in which the runs of a particular coefficient are recorded along with that coefficients value; so, $(30, 0.0)$ would represent the fact that, moving sequentially along the image rows, 30 coefficients from the previous non-zero value were zero. There are many variations on this theme, some fairly sophisticated and capable of more compact representation.

Another alternative, if one is assured of a very small set of non-zero coefficients, would be to code the position and magnitude of each non-zero coefficient in full. Such a scheme would

probably be the best one to use when encoding images with few edges or textures.

One can also code position values implicitly i.e., infer the position from the way in which the magnitude information is stored. This is the method that was used in this dissertation. The most obvious implicit scheme involves coding each coefficient sequentially, row after row. Of course, coding each coefficient in full (8 bits) would not result in image compression. However, it turns out that one can get by without using the full dynamic range for each coefficient. Indeed, for those coefficients which are close to zero, we would like to use zero bits i.e., not encode them at all. The decision as to the number of bits to use in coding coefficients is called *bit-allocation* and is usually based on the statistical properties of the coefficient sub-band under consideration. Based on such allocation strategies, and given that one wishes to achieve a certain pre-specified compression ratio (i.e., a pre-specified number of bits to code the entire wavelet representation) with the minimum possible distortion, these calculations yield an estimate of the number of bits which should be used to code each sub-band.

Suppose that one has a certain number of bits with which to encode a floating point value, what should the new value be i.e., how does one *quantize* this number?

4.3 Quantization

Quantization is the process whereby a value which has a large (possibly infinite) dynamic range, is curtailed so as to fit within the dynamic range of an alternate representational system. For example, a real number (which has infinite dynamic range) must be quantized so that it can fit within the 32 or 64 bit format which a computer allows for such a representation. If the number exceeds the capacity of the format, it might be truncated to the maximum value or generate an overflow trap. However, even if the value lies within the permissible range, the accuracy with which it can be represented is limited. For example, one cannot specify a fraction such as 1/3 beyond a prescribed accuracy. The quantization process determines which alternative value amongst the permissible permutations best fits the input value.

The values which are associated with the wavelet transform are real values and if represented in 'full' would require several bytes. At best, one would need the same number of bits as the input pixels possessed to represent each coefficient — 8 bits in this case. However, the

accuracy with which one needs to represent a coefficient varies from level to level and quite often one can make do with considerably less accuracy and still maintain a high degree of fidelity. For example, coefficients which are near zero might well be left out, implicitly coding them with zero bits. In addition, a coarse representation (few bits per coefficient) of the detail coefficients produces errors which are less irritating to the eye, since these inaccuracies induce small high contrast errors in the reconstruction, which the HVS is less likely to emphasize.

There are two different approaches to quantizing an input value: uniform and non-uniform quantization. *Uniform quantization* divides the permissible range into a number of uniformly spaced *bins* or *partitions*. Any value which falls within the bounds of a particular bin is (usually) represented by the mid-point value of the bin (mid-step quantization). *Non-uniform* quantization uses bins which have varying size; this is useful if one, a priori, knows that a particular range of values has to be accurately represented while another can be coarsely quantized. One ensures that the bins are more closely spaced within the former region.

Rather than representing the quantized value directly, it may be more convenient to insert it into a *code-book*, and to refer to it by an *index*, which generally has fewer bits and is easier to manipulate (if one wishes to, say, perform Huffman coding on the quantizer output). The existence of the code-book adds to storage requirements; however, if the code-book is static, only one copy need exist (usually as a separate file, which quantizers can reference). The indices used in the code-book must have a sufficiently large range to label all the bins present. Thus, using this approach, an eight bit quantizer would allow 256 distinct indices and hence 256 quantized real values to represent the input.

Using the above approach, one cannot achieve a *bit-rate*[4] of below 1 bpp, since one needs at least two bins for quantization to be meaningful. Vector quantization provides a way of bringing the bit-rate below unity.

[4]A measure of compression, given in bits per pixel (bpp). Since each index encodes a particular pixel, a quantizer with 256 partitions, and hence an 8 bit index, implies a bit-rate of 8 bpp.

4.4 Vector Quantization

If one only employs scalar quantization (discussed above), then each pixel must be represented by at least one bit [18, 27, 8, 51, 26]. Hence, the upper bound on the compression ratio is $n : 1$, where n is the number of bits required to represent the grey-scale values of the input image. However, if one uses a single index to refer to a block (or vector) of pixels, the bit-rate can be drawn below unity. The bins (partitions) in such a *vector quantization* (VQ) scheme are now multi-dimensional, and the means of deciding which vectors should be mapped into a particular partition becomes decidedly more expensive. One uses a *distortion metric* to make this decision. Such a metric compares the vector to be quantized with all the vectors in the *reproduction alphabet* (i.e., the set of representative vectors for all partitions) and determines which reproduction vector, and hence partition, minimizes the distortion (the error induced by quantization). Just as with scalar quantization, each reproduction vector resides in a code-book and has an associated index. Now, however, one can achieve a minimum bit-rate of $\frac{1}{n}$ bpp when quantizing an n-dimensional vector. Hence, the compression ratio is determined by the vector dimensionality, n — larger n implies a lower bit-rate and higher compression.

One cannot simply choose a large n and assume that the quantization will be satisfactory. Consider the following example: assume that each pixel in the vector can have 32 distinct quantized states (a fairly conservative number), and assume that the vector is of size 4. The number of possible reproduction vectors is then 32^4. This, in turn, implies that the index would have to contain 20 bits to fully represent this range. This results in a bit-rate of $\frac{20}{4} = 5$ bpp — it seems that scalar quantization would have been as useful and computationally cheaper. Nonetheless, if the vectors to be coded (such as $n \times n$ blocks in an image) have a high degree of correlation, this scheme may be effectively applied, since the number of reproduction vectors can then be small.

The performance of the quantizer (as measured by the distortion metric) will become increasingly degraded as the dimensionality increases, unless one simultaneously increases the number of reproduction levels. However, this leads to a growth in computational time and a decrease in the bit-rate (since each index now requires more bits to represent it).

We are faced with two conflicting requirements:

- to increase the compression ratio (whilst minimizing the distortion) and

- to minimize the computational load of the quantization operation.

The optimal resolution of this conflict depends largely on the nature of the data to be quantized. For image data, the pixel correlation enables one to make do with a small reproduction alphabet and low bit-rates are attainable. However, direct image quantization often results in noticeable image blocking. If one applies VQ to the transformed image, any resulting blocking effects are skewed and distorted by the inverse transform, and are generally less offensive to the eye. One might question the efficacy of applying VQ to a transformed image, since the correlation has supposedly been removed. It is perhaps better, to consider a transform as a 'redundancy extractor', which gives a new representation which is non-redundant, in that it does not contain (much) unnecessary data. The transformed image still has correlations amongst its coefficients, but these are necessary ones (most transforms are continuous, thus there will be no discontinuities when smooth input data is transformed).

How does such a system function in practice? In the case of the WT, most detail coefficients are clustered around zero and those that are not may be coarsely quantized, for reasons referred to earlier. Due to this state of affairs, one can achieve a low bit-rate using VQ with the wavelet transform [1]. Good results may also be obtained using more general sub-band schemes [46].

Having discussed the basic ideas behind VQ, one would like to construct such a quantizer. There are many ways of doing this [8, 18]; for reasons of simplicity and computational economy I chose to implement LBG vector quantization, which is discussed below.

4.4.1 The LBG Algorithm

The Linde-Buzo-Gray (LBG) algorithm [27] provides an interactive method for constructing a k dimensional vector quantizer with n reproduction levels, $k, n \in \mathbb{Z}$. The distortion metric may be of any kind desired, although for reasons referred to earlier, I have opted for the mean squared error (MSE) criterion. The MSE metric, $d(x, \hat{x})$, provides a simple measure of the error between the reproduction vector $\hat{x}$ and the input vector x, and is given by

$$d(x, \hat{x}) = \sum_{i=0}^{k} |x_i - \hat{x}_i|^2. \tag{4.2}$$

The algorithm proceeds from an initial guess and refines this guess until a quantizer (codebook) has been designed which produces an average distortion which is within some specified tolerance. Central to this process, is the introduction of a representative set of *training vectors* i.e., a set of vectors from the input source (in our case, a large number of transformed pixel blocks). The source is assumed to be a random process with an underlying cumulative probability distribution function (which is unknown); the training vectors (as representatives of this source) are used in lieu of explicit distribution information to 'train' the quantizer. There are certain assumptions involved in this iterative development (the stationarity and ergodicity of the random process) which in general will only hold approximately. Nonetheless, it has proven itself practically and is commonly used. In general, the LBG algorithm will converge to a locally optimal quantizer i.e., one which does not necessarily minimize global distortion.

The algorithm is as follows [27]:

1. Given: the initial reproduction alphabet, A_0, a distortion threshold $\epsilon \geq 0$, the sequence of training vectors, $\{\mathbf{x}_j : j = 0, \ldots, n-1\}$, and N (the number of reproduction vectors). Set $D_{-1} = \infty$ and $m = 0$.

2. Find the minimum distortion partition of the training sequence; that is, using the distortion metric, map the vectors to those reproduction vectors which minimize the distortion metric.

3. Compute the average distortion of the newly distributed training vectors w.r.t. the current reproduction alphabet, A_m:

$$D_m = \frac{1}{n} \sum_{j=0}^{n-1} \min_{y \in A_m} d(\mathbf{x}_j, y). \tag{4.3}$$

   If $\frac{D_{m-1} - D_m}{D_m} \leq \epsilon$, stop with the current set of reproduction vectors as your quantizer, otherwise continue.

4. Find the optimal reproduction alphabet, A_m, by taking the centroid of all the vectors in each partition, S:

$$\frac{1}{\|S\|} \sum_{\mathbf{x} \in S} \mathbf{x}, \tag{4.4}$$

where $\|S\|$ indicates the number of vectors in the partition.
Increment m and return to step (2).

The generation of the initial alphabet must be accomplished before the algorithm is applied. There are two methods suggested for this is [27]: uniform quantization and splitting. The former involves constructing a regular lattice of k-dimensional points (the initial reproduction vectors) which contains all the training vectors. This method does not pay any attention to clustering of the training vectors, and consequently generates many empty partitions (partitions into which no vectors are mapped). Empty partitions often persist once they have developed and cause the quantizer to lose efficiency, since they are redundant and occupy space which may have been used to quantize the data more finely. The latter method, splitting, seems to be the method of choice (see, for example, [1]), and it is the one I used. Splitting requires that the number of reproduction levels be a power of two. An initial one level quantizer is obtained by taking the centroid of the entire training set. This vector is then split into two vectors by adding and subtracting a perturbation vector which has small components. This new two level quantizer is then subjected to the LBG algorithm and once a suitably low distortion level has been achieved, both the vectors are then split (by application of the same perturbation vector) and fed back into the LBG algorithm. This splitting process is continued until the required number of reproduction levels has been reached. This then provides the initial alphabet upon which the final LBG iteration will be based. This method generates all lower level quantizers (which are expressible as a power of two) and is better able to approximate clustered input data.

An analysis of the performance of the LBG algorithm, with splitting, and applied to the quadratic spline wavelet transform, is presented below.

4.5 Analysis of Quadratic Spline-Wavelet Compression

4.5.1 General Discussion

The application of VQ to (bi-orthogonal) wavelet image compression [1] and general subband coding schemes [46, 51] has produced promising results. Inspired by this work, I decided to apply the technique to the compression of the spline-wavelet transform subbands. I will refer to the results obtained there when discussing my own. The approach

that was taken (the design of multi-resolution code-books) is based directly on [1].

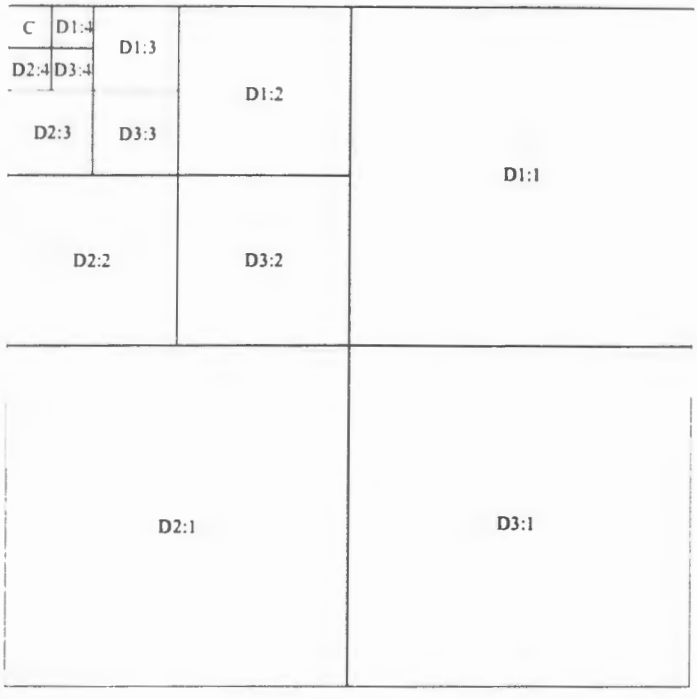
I did not use a dynamic bit-allocation strategy: the available bit-allocation options were hard-coded and remained constant for all images. This allowed for easier analysis of the results, since dynamic bit-allocation (in the context of VQ) can result in the application of widely varying sub-band quantizers to different test images and also increases the compression time (although this is normally negligible).

Only 4x4 and 2x2 blocks were used to quantize the detail sub-bands (as in [1]) while scalar quantization was used on the approximation sub-band. The number of levels of the multi-resolution decomposition was a function of the bit-rate desired: high bit-rates did not require the use of more than 2 levels, while the lowest bit-rates required a four level decomposition. The number of levels permitted for detail quantization were: 256, 512 and 1024 (i.e., 8-10 bits per block). The approximation indices were each coded with eight bits — hence the need to decompose further, since decreasing the coefficient support will improve compression performance. Figure 4.3 shows the available bit-allocations. The positional information for each coefficient block was implicitly encoded in the data stream, since the block decomposition of the sub-bands was implemented sequentially. Blocks which overlapped the sub-band boundaries were coded in full, using the symmetry extension dictated by the sub-band. This resulted in a very small decrease in coding efficiency. The zero coefficient boundaries of the quadratic transform sub-bands were not exploited — such manipulations result in only marginal compression gains and are not worth the effort.

Two training set were used for the tests — See Figure 4.4 and Figure 4.5. The first training set was a collection of some fairly dissimilar images, the second contained only the heads of men and women. The resulting code-books were written to disk and occupied between 17Kb (256 level, 2x2 blocks) to 68 Kb (1024 level, 4x4 blocks) each.

A problem which became apparent during training, was the small number of training vectors produced by blocking the lower levels of the wavelet decomposition: each level of the decomposition decreases the number of blocks by a factor of four. There are two methods that might be used to overcome this problem

- increase the number of training images
- use the existing vectors to expand the training set.



Comp	D1:1	D2:1	D3:1	D1:2	D2:2	D3:2	D1:3	D2:3	D3:3	D1:4	D2:4	D3:4
0	2	2	2	-	-	-	-	-	-	-	-	-
1	2	2	2	2	2	2	-	-	-	-	-	-
2	4	4	4	2	2	2	-	-	-	-	-	-
3	0	0	4	2	2	2	-	-	-	-	-	-
4	0	0	0	2	2	2	-	-	-	-	-	-
5	0	0	0	2	2	2	2	2	2	-	-	-
6	0	0	0	2	2	4	2	2	2	-	-	-
7	0	0	0	4	4	2	2	2	2	-	-	-
8	0	0	0	4	4	4	2	2	2	-	-	-
9	0	0	0	4	4	0	2	2	2	-	-	-
10	0	0	0	4	4	0	2	2	4	-	-	-
11	0	0	0	4	4	0	4	4	2	-	-	-
12	0	0	0	0	0	2	2	2	4	2	2	2
13	0	0	0	0	0	4	2	2	2	2	2	2
14	0	0	0	0	0	4	2	2	4	2	2	2
15	0	0	0	0	0	4	4	4	2	2	2	2

Figure 4.3: Bit-allocation. This table provides a list of the hard-coded quality settings and the corresponding block sizes for the sub-band quantizers (2 = 2x2 etc). The bit-rate decreases monotonically as the compression parameter increases. Hyphens indicate that the sub-band decomposition was terminated before reaching the corresponding level. Zero entries mean that no coefficients from that sub-band were encoded. The approximation coefficient sub-band is always scalar-quantized.

Figure 4.4: Generic Training Set. The training set used for generic quantization training. The images shown here have been scaled down.

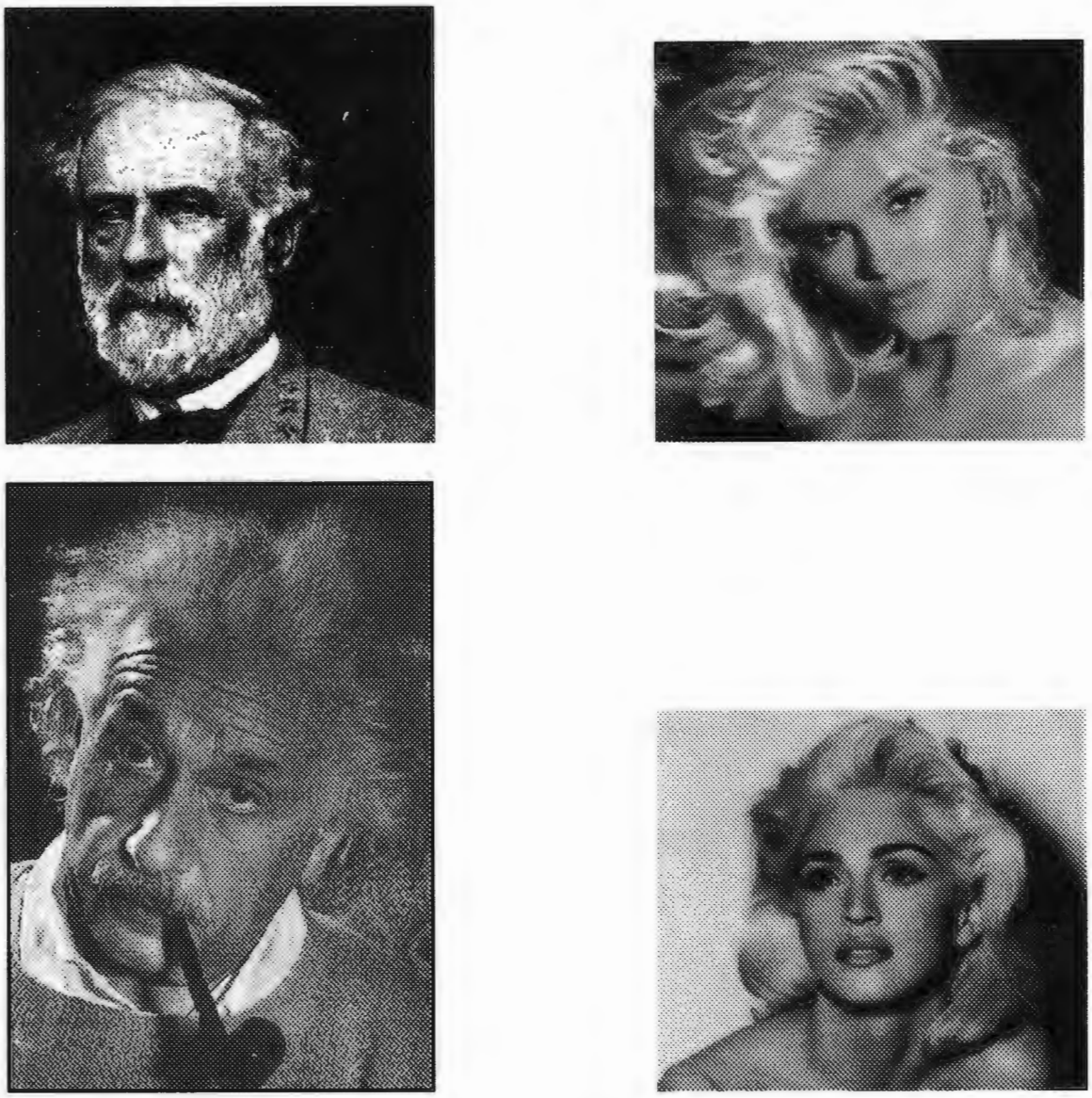


Figure 4.5: Alternate Training Set. This training set was used to examine the effects of subject-specific quantizer training. As before, the images shown are not necessarily to scale.

Increasing the size of the training set is undesirable since one would need a large number of training images to generate the required number of training blocks for, say, a 1024 level, 4x4 block quantization of a fourth level sub-band. The second option involves applying a de-correlating action to the available blocks so that they may be used as new training vectors. The approach I used was suggested in [46]: the wavelet coefficients on each level are shifted one index left, right and diagonally, and each shifted image serves as the source for a new wavelet decomposition. This shift is sufficient to de-correlate the coefficients and consequently provides us with suitable training vectors. The method is applied recursively, until the maximum depth of the decomposition is reached. This ensures that the number of blocks produced on each level of the decomposition is the same; thus, one can make do with a small initial training set.

The splitting process which generates the initial alphabet turned out to be problematic. When an arbitrary (small) splitting vector was chosen, the initial alphabet invariably contained a large number of empty partitions (particularly at lower levels). These partitions had to be removed, to bolster the performance of the quantizer (empty partitions serve no purpose). I devised a strategy to alleviate this problem: rather than splitting each and every partition, only those satisfying some 'splitability' criterion are split. After each iteration of the splitting algorithm, the number of partitions which have still to be split are determined and the required number are selected, based on their splitability, and propagated; the others are left untouched. This means that more than $\log_2 n$ steps might be required, and also introduces a bias into the generation of the initial alphabet; but this is unavoidable if one wishes to eliminate empty partitions. The splitability index for each partition is simply the average distortion of all the vectors that map into that partition. Thus, partitions which are well represented will not be split, while those which are coarsely represented will be. This seems to be the most logical choice; the results are given in the next section. Weighting the splitability based on the magnitude of the reproduction vector (thus implicitly favouring either smooth or detailed regions) proved to be unsatisfactory. One could undoubtedly conceive more cunning measures of splitability, but the impact that this would have on the production of good quantizers is questionable.

The distortion measures used to determine reconstruction performance were the signal-to-noise ratio (SNR) and the normalized mean squared error (NMSE). These were defined

as

$$\text{SNR} = 10\log_{10}\left(\frac{\text{var}(\text{I})}{\text{var}(\text{I}-\text{R})}\right) \text{ dB} \tag{4.5}$$

$$\text{NMSE} = 100\frac{\text{var}(\text{I}-\text{R})}{\text{var}(\text{I})} \% \tag{4.6}$$

where I and R represent the original and reconstructed images, respectively, and var(L) denotes the variance of the image L [26]. The choice of these fidelity metrics was based on simplicity; there are some cases in the following results in which reconstructions with a poor SNR are actually visually more acceptable than those which have a high SNR. The reconstruction error was taken with respect to the projected image — this serves to decouple the errors introduced by quantization from those produced by the input approximation.

The file format for the compressed image is presented in Figure 4.6. Currently, only pgm files (grey-scale) are compressible. 24-bit ppm images could also be incorporated by applying the same techniques to each colour component. Greater compression ratios (for colour images) could be achieved by converting the input to YIQ colourspace and down-sampling the chrominance components before applying the encoding techniques.

4.5.2 Results and Analysis

This section presents data which quantifies the performance of the LBG quantizers produced under the assumptions discussed above. The test images were chosen from both within and without the training sets. These test images contained varying amounts of texture and smoothness. Some, like 'square' did not contain any texture; 'tree', at the other end of the spectrum, was almost exclusively texture.

Figure 4.7 shows SNR and NMSE vs compression ratio for the images 'tree', 'lenna' and 'square'. These images were quantized w.r.t. training set one; that is, 'tree' and 'square' were in the training set, 'lenna' was not. The SNR figures imply that those images which are highly textured ('tree') are not well reproduced, while those with little or no texture are. The inclusion of a test image in the training set is no guarantee of good SNR performance. However, examination of Figure 4.8 shows that 'tree' produces a far more acceptable reproduction (as measured by a human subject) than 'lenna'; this effect is known as 'texture-masking'. 'Lenna' possesses large smoothish regions and it is here where quantization noise is readily discerned. A dramatic increase in HVS distortion of 'lenna' occurs when the number of decomposition levels passes two. This may be partially explained by considering the

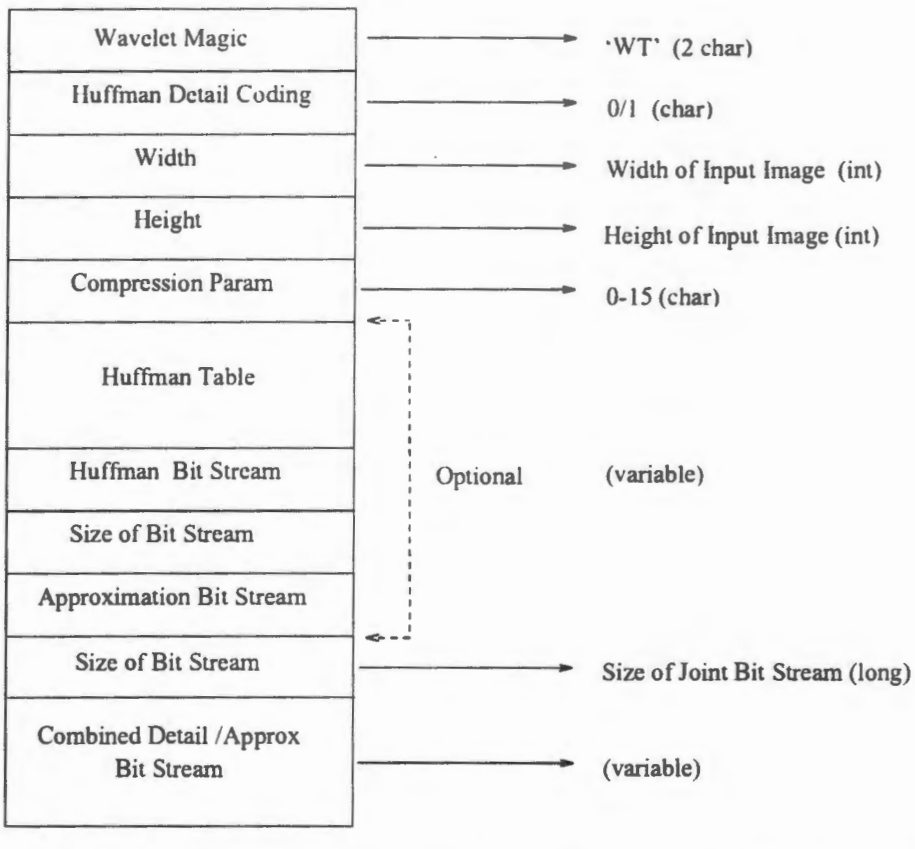


Figure 4.6: The format of the compressed wavelet file. The dotted lines indicate that the Huffman coding of detail block indices is optional. If this option is not chosen, a single unified bit stream is used for both the detail and approximation blocks. The quality parameter (0-15) provides an approximately linear increase in the compression ratio: 0 = best, 15 = worst.

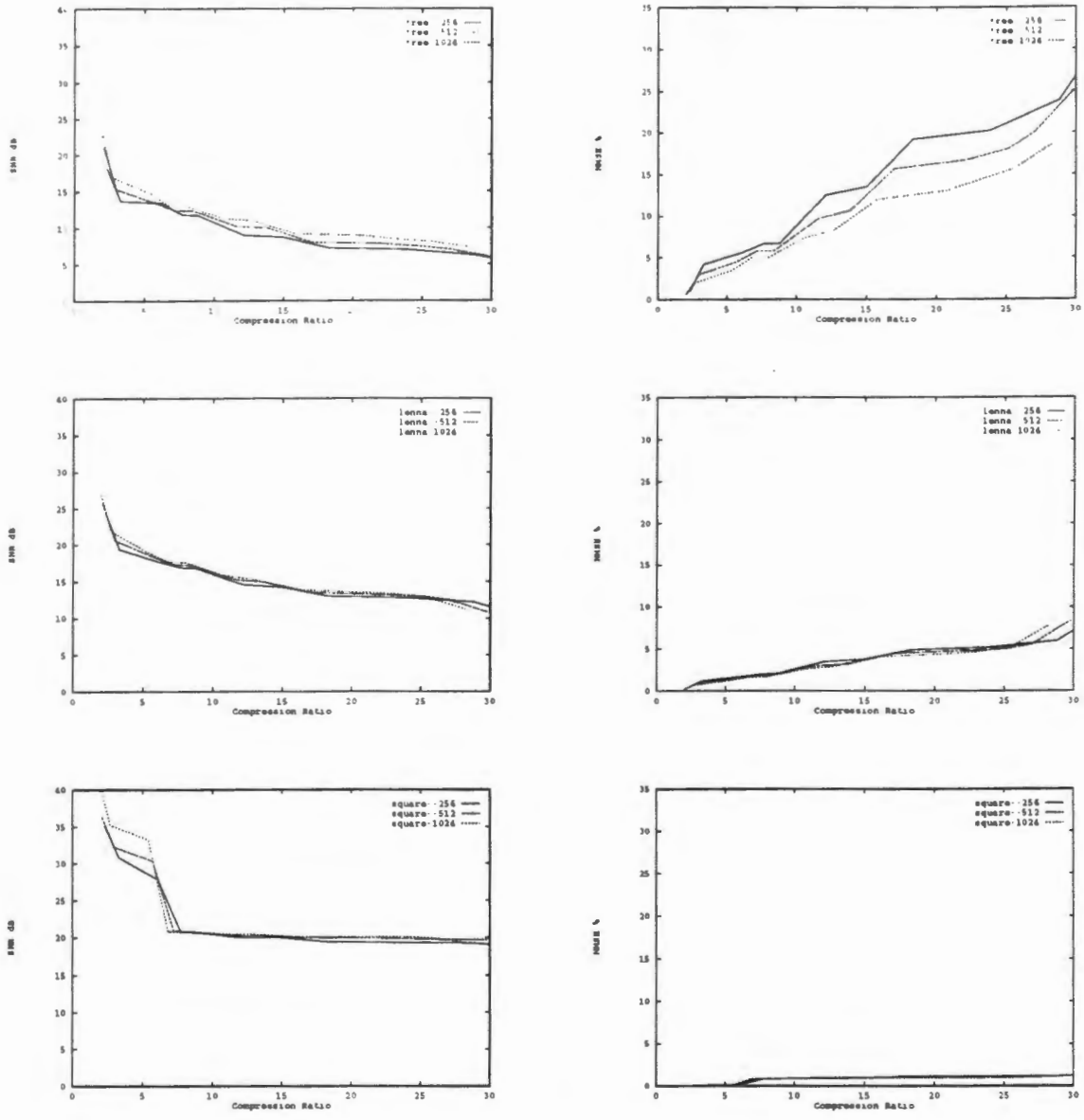


Figure 4.7: SNR and NMSE figures for reconstruction. These images were quantized with quantizers derived from training set one. Despite a poor SNR, 'tree' is by far the most acceptable visually. The different curves in each diagram represent the performance obtained when the indicated number of reproduction vectors is used.

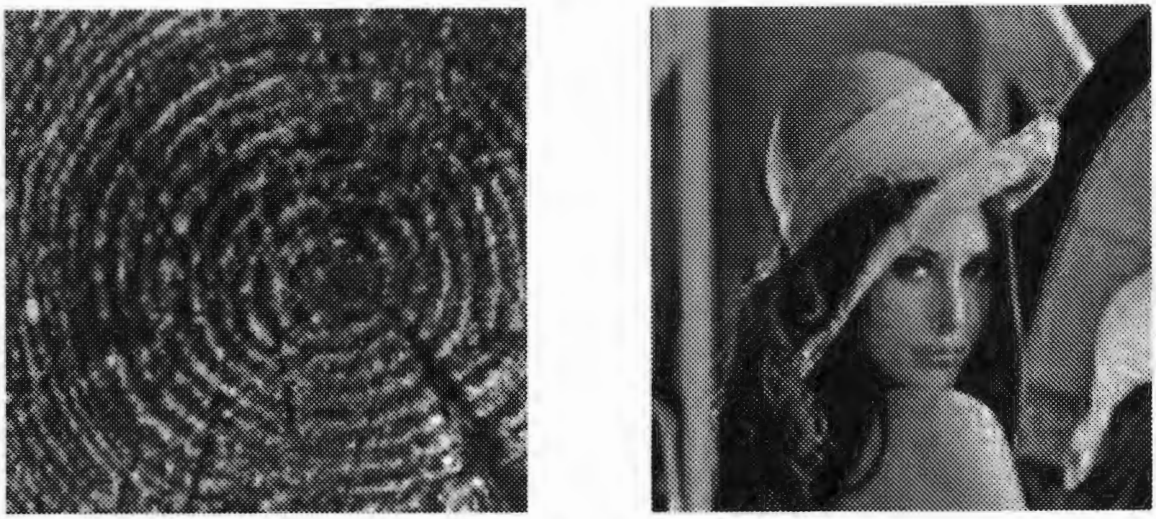


Figure 4.8: SNR vs HVS fidelity metric. **The compression ratio is 15:1 and the signal-to-noise ratios are 14.1 dB (lenna) and 8.5 dB (tree). Nonetheless, the texture in 'tree' serves to mask the (bad) quantization errors, while 'lenna', despite having a substantially better SNR, is heavily distorted.**

nature of the quantization errors induced as a function of level. As one proceeds to lower levels in the multi-resolution pyramid, the support of the wavelet grows from 4 pixels at level 1 (quadratic case) to 78 pixels at level 4. Any bad quantization decision at low levels, will result in a modulation of the corresponding synthesis wavelet which will consequently fail to sum correctly and produce ripples and blotches (of level-dependent size) into the reconstruction. Errors produced in the level 2 detail images, are still small enough to be minimally irritating. However, if the image was highly textured to begin with, these errors may be adequately masked and the reconstruction may still be acceptable ('tree'). The degradation in performance at high compression ratios is also dependent on the block size which underlies the quantizer (for reasons elucidated earlier). Unfortunately, high compression ratios (low bit-rates) necessitate the introduction of large (4x4 in this case) blocks, which cannot adequately describe the coefficients they encode.

The second training set is based exclusively on (the head and shoulders of) human subjects. The test images used in this case were 'tree' and 'lenna' (outside the training set) and 'lee' (inside the training set). The SNR and NMSE results are plotted in Figure 4.9. Once more the acceptability of the reconstruction for 'tree' is belied by the numerical results. The test image 'lee', also performs remarkably well; it is in the training set and has a large region of black, in which contrast errors do not readily show up (See Figure 4.10).

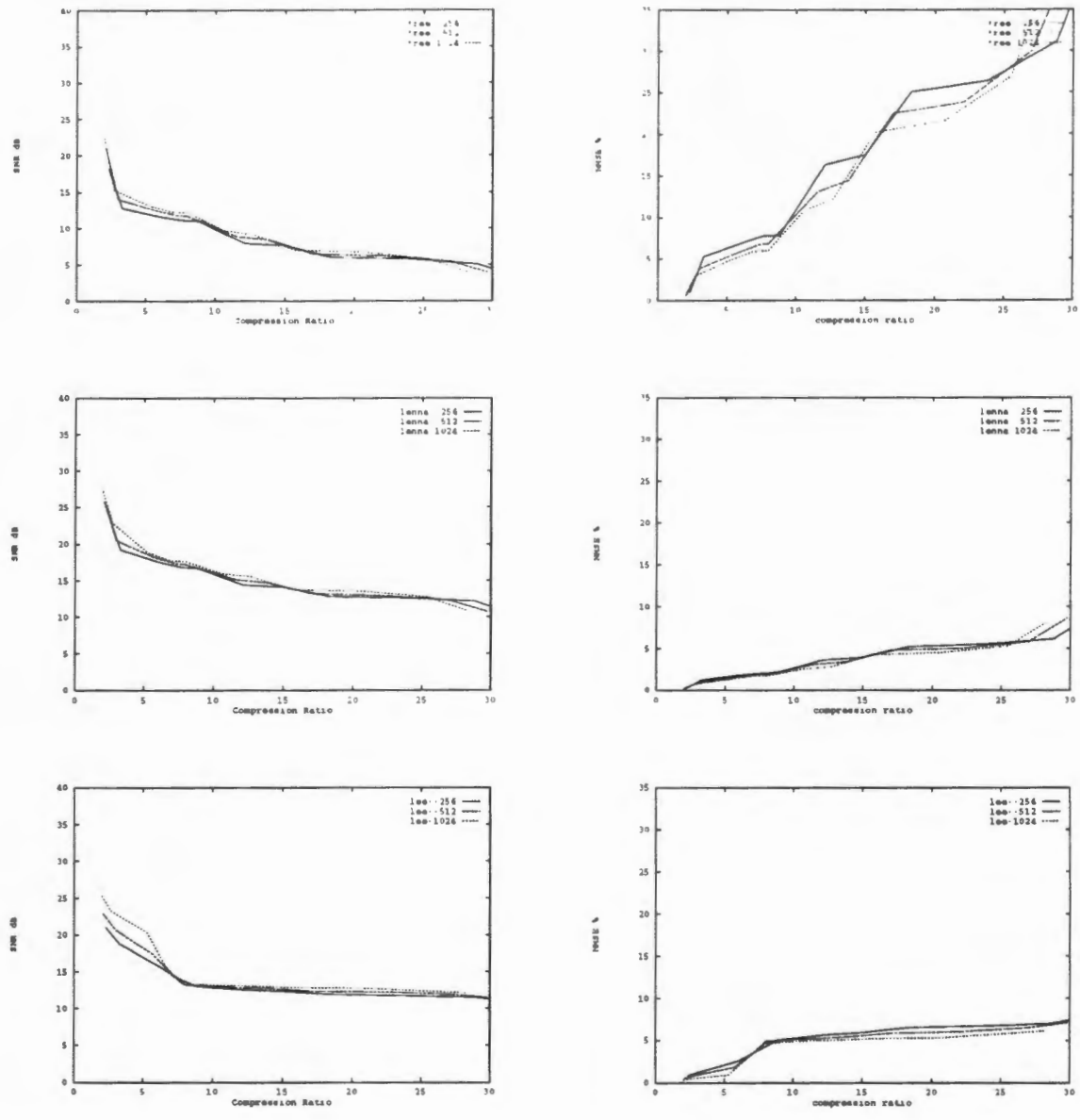


Figure 4.9: SNR and NMSE figures for reconstruction. These results were obtained using training set two. The figures are very similar to those obtained using the generic training set; once more, inspection of the corresponding images revealed that the SNR figures are deceptive.

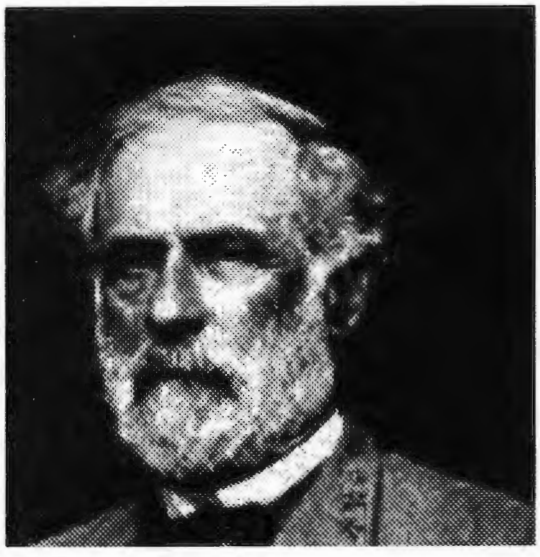


Figure 4.10: HVS vs SNR. The image shown here ('lee') has a preponderance of dark colours which mask quantization errors. The signal-to-noise and compression ratios are 11.8 dB are and 23.8:1, respectively.

Despite the fact that 'lenna' qualitatively falls within the same category as the training set, both sets of quantizers produce very similar performance (Figure 4.11). It may be that the images, while qualitatively the same i.e., under our inherent HVS metric, are not mathematically similar enough to really make any difference.

In general, then, the performance of the spline-wavelet LBG VQ system on images without an excessively high texture content would seem to be somewhat unsatisfactory. Certainly, the results achieved by Mathieu et al [1] were not realised when (semi-orthogonal) spline wavelets were used. The following might be cited as possible reasons for this lack of performance:

1. semi-orthogonal wavelets may be inherently unsuited to image compression
2. the means of producing the initial alphabet may be incorrect or inappropriate.
3. dynamic bit-allocation may be of fundamental importance.

There is no obvious reason why semi-orthogonal wavelets should be any less satisfactory than general bi-orthogonal wavelets: the statistical characteristics (at least those which underlie the compressibility) of all wavelet transforms are very similar. There is, however, a difference between orthogonal and non-orthogonal wavelet schemes as regards their

Figure 4.11: Typical compression results. The left image was compressed to 3.2:1 while the right image was compressed to 8.7:1. The maximum number of reproduction vectors for each sub-band was 256 (Figure 4.3) and no Huffman coding was used. The SNR's are 19.2 dB and 16.8 dB, respectively. Observe that quantization effects are immediately apparent as soon as a sub-band decomposition of two or more levels is used; in this example, both images exhibit these artifacts. The quantizers used were generated from the generic training set i.e., training set one.

compressibility when using a MMSE quantization scheme. Under an orthogonal wavelet transform the L^2 norm is preserved; hence, minimizing the (mean-squared) quantization error in the transform domain ensures that one simultaneously minimizes the reconstruction (L^2) error in the image domain. The fact that non-orthogonal schemes do not possess this property means that they are generally 'sub-optimal' under such a distortion metric. Nonetheless, norm preservation may be assumed provided the wavelets are 'almost' orthogonal. This seems to be the case with bi-orthogonal compression scheme used in [1]. Semi-orthogonal spline wavelets (despite nominally being near orthogonal) do not appear to possess a high enough degree of orthogonality to successfully employ a simple MMSE quantization rule. The question of what (visually relevant) norm might be preserved under such a transformation is not one I have examined. However, [38] offers a method of working around this problem: one employs a 'multi-scale relaxation' algorithm to minimize the L^2 error explicitly, rather than performing this minimization in the transform domain. The alternative is to minimize some other distortion measure, such as the L^1 norm (of the image). However, if this approach is taken, one then has to produce the corresponding analytical expressions to allow the minimization to proceed in the transform domain, which is sometimes impossible.

The support of the wavelet will also play an important role in determining the extent of quantization errors. A wavelet with small extent is desired, since the error is then well localized. The support of the wavelet doubles with each subsequent resolution step which explains the more conspicuous nature of low frequency quantization errors. The support of the quadratic wavelet is $[0, 5]$, which is of intermediate size. While it can certainly be argued that a smaller wavelet would be better (i.e., produce more localized quantization error), the manner in which quantization is undertaken is of greater importance. It would seem that the combination of non-orthogonality and (MMSE) vector quantization is highly inappropriate for the lower levels of the spline-wavelet decomposition.

It seems unlikely that the absence of dynamic bit-allocation could account for the consistently poor results obtained. While performance without 'noise-shaping' will be sub-optimal, this implies a small decrease in coding efficiency — not the dismal results produced here (for low bit-rates).

The issue of initial alphabet generation is one which I find highly unsatisfactory. One must remember that the quantizers produced by the LBG algorithm are only guaranteed to minimize distortion locally; it is quite conceivable that the manner in which the initial alphabet is generated is such that the subsequent iteration does not converge to a good quantizer. If this is the case, then the manner in which empty partitions are discarded and new ones chosen is most likely to be the source of additional distortion.

Section 4.6 describes a method which might be employed to circumvent these problems.

4.5.3 Huffman Coding

The detail coefficients of the wavelet transform have a 'generalized Gaussian distribution', a property which the block indices (approximately) retain after vector quantization [1]. Since this implies that the block indices are not arbitrarily distributed, one may use an entropy coding to exploit this source of redundancy. ***Huffman coding*** is such a method and it generally allows one to approach the entropy of the sequence to be coded. The entropy, H, a measure of the information contained in the sequence, is given by

$$H = -\sum_i p_i \log_2 p_i \quad \text{bpp}, \tag{4.7}$$

where

$$\sum_i p_i = 1 \text{ and } p_i \geq 0. \tag{4.8}$$

Here p_i represents the probability of index i occurring. This value is derived from a histogram of index occurrences.

Many of the transform coding techniques that I examined, follow the quantization phase with some sort of entropy coding. Some go so far as to use the entropy of the index sequence as an objective measure of the attainable bit-rate for each index, implicitly assuming that a perfect entropy coding follows quantization. This is not accurate, however, since

1. one can only approach the entropy asymptotically
2. the amount of header information which inevitably accompanies the compressed file might well be significant
3. for the case of static Huffman coding, at least, the Huffman code sequences must be stored with the file; the size of this information depends on the statistical properties of the sequence, but it averaged about 2 bytes per table entry in my implementation — see Table 4.1. It is quite easy to produce cases in which additional Huffman coding actually increases the size of the compressed file.

In addition to this, one has the overhead of performing the encoding and decoding. While Huffman encoding can be done fairly rapidly, the decoding requires a linear search of all the bit strings and can be comparatively lengthy. Many of these objections can be overcome; for example, dynamic Huffman coding allows one to reconstruct the bit-strings when decoding, rather than having to store them, although one pays for this with a decrease in coding performance. In addition, one may use a scheme such as arithmetic coding, which also performs well and does not have as much accompanying baggage. The size of the header information depends on the complexity of the encoding. The compression ratios I cited were based purely on file size, since I feel that any information which is part of the compressed file is non-redundant and therefore forms an integral part of the new representation. Table 4.1 provides some information which illustrates these points. It would also be possible to provide an 'adaptive' Huffman encoding, in the sense that the program could decide whether the entropy in the coefficients justified the application of additional entropy coding.

File	Size	No-Huff	With-Huff	Table	T-Ent	A-Ent	% Gain	Q-Lvls
lenna	65551	10775	10003	559	6.32	6.36	7	256
house	65551	10775	9506	618	5.65	5.67	11	256
tree	68429	11103	11344	528	7.62	7.64	-2	256
hanna	106315	17954	15699	564	6.37	6.40	9	256

Table 4.1: Huffman coding overhead. This table provides some data on the overheads imposed by static Huffman coding in the proposed compression scheme. Table denotes the size of the Huffman table (in bytes), T-Ent and A-Ent the theoretical and attained entropies (in bpp), Gain is the overall compression gain and Q-Lvls give the max number detail reproduction levels.

4.6 An Alternative Compression Scheme

The above discussion should make it apparent that VQ does not exploit the wavelet transform (or at least the spline-wavelet transform) to its full potential. This failure is, at least in part, due to the manner in which the quantizers are produced viz. by some sort of learning or training mechanism. In the case of VQ, the training process 'averages' the training set and unless one has a very high number of reproduction levels, the results are, in general, less than satisfactory. A far better, approach, in my opinion, would be to consider each image on its own merits, rather than attempting to extrapolate general image characteristics from a training set. This is the approach taken by Lewis et al in [25].

One starts at the lowest level of a four level decomposition, by first encoding the approximation coefficients with eight bits each. Each 2x2 coefficient block in each of the 4th level detail sub-bands is then considered. The visual threshold of each block is computed and if that block is considered to be important (its energy normalized w.r.t. the visual threshold is large enough) then the block is encoded; if not, the block is not encoded. Each coefficient in the block is coded with a linear mid-step quantizer, the step size of which is determined by another HVS calculation. Once a block has been transmitted, the 4 pixel values which correspond to each of the block's pixels in the higher-resolution sub-band (i.e., the one that was down-sampled) are considered. They are subjected to the same sort of HVS thresholding and, if they are deemed important enough, they are transmitted. This process of checking and transmitting coefficients continues recursively, for each sub-band, until all the spawning blocks on level 4 have terminated (reached level 1). The encoding process exploits Marr's observation that important detail persists over multiple resolution levels — if an edge exists at level l, it should also exist at level $l-1$ (one level up) and thus the higher level blocks which give rise to the block under consideration should also be

considered.

Another bonus for the scheme is that the explicit positional information for each block may be inferred from its position in the quantized stream, provided the recursive encoding includes tokens to indicate when a stopping condition (no transmission) is reached. Contrast this to VQ, in which every block has to be coded. Since the quantization is based on characteristics of each image, rather than projected statistics, the coding should be more robust and can certainly be called adaptive. The quantization itself is preferable, since each coefficient is quantized separately allowing for better reproduction of the coefficient values. The coefficients are Huffman coded. There will be some overhead in storing quantizer parameters, but this is minimal — most of the information to describe the quantizer may be regenerated recursively during reconstruction, since this information depends only on information already re-created.

Some final observation are in order. Firstly, the method described will result in an image dependent compression ratio: highly uniform images will fare better than those with much variation. Secondly, the method was implemented using the Daubechies order four orthogonal wavelet, which has smaller support than that of the quadratic spline wavelet. A wavelet with small support ensures that quantization errors are well localized.

4.7 Concluding Comments

Wavelet image compression is capable of producing high compression ratios whilst maintaining important structural information. The efficiency with which the transform can be implemented means that one does not need to apply image-domain blocking operations, thus eliminating blocking effects.

MMSE vector quantization has been successfully used with coding of bi-orthogonal wavelets; however, the use of a mean-squared error distortion measure would seem to be inappropriate for the spline-wavelet compression used in this dissertation: the poor compression performance can be ascribed, at least in part, to the non-orthogonality of the transform and the subsequent non-preservation of the L^2 norm. This should not be seen as an indication of poor compression potential: provided a suitable mechanism can be found to encode the necessary (sparse) information in the transform domain, spline-wavelet coding should achieve

similar performance to it's orthogonal counterparts. Here it was the quantization scheme which was inadequate, *not* the variant of the wavelet transform employed.

The alternative compression method outlined above seems a more natural and appropriate means of attaining the desired performance. However, the choice of an appropriate distortion measure for use in the quantization phase remains problematic. Despite these objection, it was shown that texture-masking effects may permit the VQ compression scheme described above to remain feasible for heavily textured images.

Chapter 5

The Difference Engine and Image Synthesis

Once the wavelet-encoded image has been decompressed, the array of data values must be converted into a visual representation i.e., the pixels corresponding to the reconstructed image must be appropriately illuminated. This phase is usually accomplished by directly mapping the data values into video memory or by means of special display processors, which may perform a variety of higher level pixel manipulations. In both of these cases it is assumed that the value for each pixel will be *explicitly* provided by the front-end. This assumption seems to be axiomatic. This need not, however, be the case. The *Difference Engine* provides a means of circumventing this requirement, provided that the scan-line data are restricted to lie on a polynomial of some specified degree. In this case, all that this display processor requires is a list of the polynomial's forward differences and initial intensity (hence the name). Although the full complement of pixel values is eventually displayed, those which are not specified can be determined very speedily and without the aid of complex logic.

There are a number of issues which must be considered before such a scheme can be effectively implemented and these will be discussed in the following sections.

Section 5.1 provides an overview of the Difference Engine (DE). The manner in which a spline multi-resolution analysis can be used to synthesize an image on the DE is investigated in Section 5.2. Section 5.3 examines methods of improving the efficiency of the synthesis

procedure, while Section 5.4 analyses the performance of the proposed system.

5.1 The Difference Engine

The Difference Engine is the final component in the rendering pipeline of a new display architecture developed at CWI [3]. Originally designed to provide rapid rendering of Phong shaded objects, which have polynomial intensity profiles [3], this display processor has the ability to interpolate an arbitrary length span of such pixels with a single instruction. The interpolatory logic is implemented as a systolic array — each new cycle produces the complete set of values for the specified span.

An nth degree polynomial span may be specified by a starting point, a set of n forward differences and the width of the span. The pth order forward difference of $I(x)$ is

$$(\Delta_p I)(x) = (\Delta_{p-1} I)(x+1) - (\Delta_{p-1} I)(x), \tag{5.1}$$

where

$$(\Delta_0 I)(x) = I(x). \tag{5.2}$$

Once the required differences are computed, using the simple recursive scheme presented above, the polynomial values at uniformly spaced intervals ($\mathbb{Z}$, in this case) may be obtained by using the following simple update rule

$$(\Delta_p I)(x+1) = (\Delta_p I)(x) + (\Delta_{p+1} I)(x), \quad p = 0, \cdots, n-1. \tag{5.3}$$

for consecutive values of x. The 11ns cycle time of this processor means that one can perform these calculations with sufficient speed to ensure pixel production at the display refresh rate.

The proposed architecture does not employ a frame-buffer. Instead, the image is represented as a list of primitives and the objects selected from this list are converted into DE instructions by customized hardware, at a sufficient rate to provide real-time video display. The complexity of the image determines the size of the list and consequently the number of instructions which are produced.

The instruction set for the DE is presented in Table 5.1.

There are two important points which should be noted:

operation	description	cycles
acc_mode	Accumulate mode: if enabled negative intensities are not added to accumulator	1
dis(x,dx)	disable accumulation of intensities from pixel 'x' for 'dx' pixels (cleared after next 'eval*' command)	1
eval0(x,dx,i)	Set pixel (i.e., accumulator) from 'x' for span of 'dx' directly, disable further additions until next refresh.	1
eval1(x,dx,i)	add i to accumulator for span from 'x' for 'dx' pixels	3[a]
eval2(x,dx,i,di)	First order forward difference, starting at pixel 'x' with value 'i' and increment 'di', for 'dx' pixels	4[a]
eval3(x,dx,i,di,ddi)	Second order forward difference — like 'eval2' except now 'di' is also changed by 'ddi' at each step	5[a]
eval4(x,dx,i,di,ddi,dddi)	Third order forward difference, like 'eval3' *mutatis mutandis*	6[a]
eval n	Higher, $n-1$, order forward differences	$n+2$[a]
nop	No operation	1
refresh	Output accumulator value and clear everything	2
setddi(x,ddi)	Set (i.e, override) second difference[b] at the point 'x' with 'ddi' (issued before an 'eval*')	2
setdi(x,di)	Like 'setddi' only it affects the lower forward difference	2
seti(x,i)	Like 'setdi' except that this creates a span of intensities	2
setpddi(x,dx,ddi)	Set (i.e, override) second difference[b] at points 'x', 'x+dx', 'x+2dx', ...in the middle of the next 'eval' command	2
setpdi(x,dx,di)	Like 'setddi' only it affects the lower forward difference	2
setpi(x,dx,i)	Like 'setdi' except that this creates a pattern of intensities	2

Table 5.1: Difference Engine Instructions and Their Costs in Cycles
Note: The costs mentioned above are incurred whether a span is 1 pixel long or covers the whole width of the scan-line.

[a] The cost of this operation can be reduced by 2 cycles in future versions
[b] If there are higher order differences then this sets the highest order difference

- the DE can interpolate arbitrary order polynomials, in time proportional to the degree (currently $n + 2$ cycles for a polynomial of degree $n - 1$).
- the DE provides a scan-line accumulator.

The DE can interpolate polynomial spans accurately up to a length dependent on the degree of the polynomial — currently about 4096 pixels for a quadratic and 512 pixels for a cubic. This limit poses no problems, since the image data can be segmented into several spans if the need arises, which is unlikely if one uses the quadratic scheme proposed in earlier in this dissertation.

The existence of an intensity accumulator is essential if one wishes to use the DE for multi-resolution image synthesis, since one then needs to accumulate several levels of detail for each scan-line. Furthermore, since one would like to produce the pixel stream as quickly as possible, the first point implies that a low order polynomial should be selected. These issues will now be taken up.

5.2 Multi-resolution Image Synthesis on the Difference Engine

The DE provides a means of efficiently displaying images which are based on polynomial patches. Thus, if one can find a means of decomposing an arbitrary input image into a series of polynomial primitives, the DE can put its speed and architecture to good use and (in appropriate conditions) provide performance gains over conventional display hardware. In particular, since only a small set of pixel values are required to interpolate an arbitrarily[1] long span, the computational demands on front-end processor (which must compute the intensity values) may be drastically reduced. This is of particular benefit when one performs image reconstruction after compression, since the process of computing each restored pixel value may be computationally expensive — if the restored values are constrained to lie on a polynomial one need only compute sufficient values to determine the forward differences. The DE will implicitly, and at negligible cost, provide the remainder of the pixel values.

But how does one achieve this polynomial segmentation? The means of providing such a decomposition was introduced and developed earlier in this dissertation: polynomial-spline

[1]Note: numerical accuracy eventually becomes an issue.

multi-resolution analysis. It should now be clear that the semi-orthogonal MRA was chosen for the following reasons:

1. to provide the (reputedly) high compression ratios of wavelet coding
2. to enable the efficient synthesis of compressed images on the DE.

The reconstructed image consists of a series of detail images and an approximation image, as discussed in earlier chapters. On level j, the approximation image satisfies the following relationship

$$I^j(i,l)|_{[k2^j,(k+1)2^j]^2} \in \pi^2_{m-1}; \quad i,j,k,l \in \mathbb{Z} \tag{5.4}$$

That is, the restriction of the approximation function, $I^j(x,y)$, to the cardinal integer knot-sequence indicated provides us with a polynomial of degree $m-1$ in two variables. The detail image, $g^j(x,y)$, on level j satisfies a similar relationship, but with $j-1$ substituted for j. Thus one arrives at a segmentation of the image in terms of polynomial patches (of order m). Once the bounding dimensions of a patch have been determined (Equation (5.4)), the 2-D polynomial can be interpolated using successive horizontal sweeps of a 1-D forward difference scheme, since the x cross sections of such a (tensor product) patch are themselves polynomials in one variable. One could also employ a scheme based on 2-D finite differences, but such an approach is rather cumbersome, prone to inaccuracy and introduces additional overheads — this issue will be taken up in a later section.

Thus, to recreate an image from its multi-resolution (MR) decomposition, we have the following algorithm (for each scan-line):

1. Determine the size of the polynomial spans
2. For each scan-line in the detail image and approximation image
   2.1 For each span on the scan-line
      2.1.2 compute the required differences
      2.1.3 compose the appropriate DE instruction to interpolate 2^j pixels and send.

The pixel values which must be computed are determined from the approximation and detail image formulae presented earlier. The latter, in particular, is expensive since there are three detail signals involved at each point. Some comments are in order.

Firstly, since the goal of the scheme is to achieve more efficient production of pixel values by only computing essential information, one does not wish to compute difference information unnecessarily. For an nth degree polynomial we need n pixels to compute the required differences; if the spans are of this length or shorter, one should set each pixel in the scan-line directly (using the 1 cycle eval0 instruction, cf. Table 5.1).

Secondly, although the spans are of length $2^{\{j;j-1\}}+1$ pixels, only the first $2^{\{j;j-1\}}$ of these are set; the last pixel serves as the starting point for the next span and is set by the appropriate eval$n$ instruction. Of course, the instruction which interpolates the last span on the scan-line *will* set the last pixel. The semi-colon notation is used to distinguish between the indices for the approximation and the detail span lengths, respectively.

Thirdly, the full MR synthesis is more expensive than simply setting each pixel directly, in terms of both pixel evaluation costs and the number of cycles required to interpolate each scan-line. The number of *function evaluations* (i.e., $g_k(i,j)$, $I_k(i,j)$) required for multi-resolution synthesis of a scan-line is

$$F(m,L) = m\frac{|x|}{\max(2^L,m)} + m\sum_{j=1}^{L}\frac{|x|}{\max(2^{j-1},m)}, \tag{5.5}$$

where $L \geq 0$ is the number of levels in the decomposition, $|x|$ is the length of the scan-line (in pixels) and m is, as usual, the order[2] of the polynomial. This cost function does not tell one anything about the actual computational cost of each function evaluation. The approximate number of *multiplications* required[3] (a fair indicator of computational complexity — the number of additions is roughly the same) is

$$M(m,L) = \sum_{p=1}^{3}\sum_{j=1}^{L}\left\lceil\frac{m|x|}{\max(2^{j-1},m)}\right\rceil W^d(p,m) + \left\lceil\frac{m|x|}{\max(2^L,m)}\right\rceil W^a(m), \tag{5.6}$$

where the the weighting functions W^d and W^a provide the approximate number of multiplications required to evaluate a pixel in the detail and approximation images, respectively. These weights are as follows:

$$W^d(p,m) = \begin{cases} 2m^2 & p = 1,2 \\ 4m^2 & p = 3. \end{cases} \tag{5.7}$$

[2]Recall that an order m polynomial is of degree $m-1$.

[3]This approximation is based on direct evaluation of only *necessary* pixel values.

$$W^a(m,j) = m^2. \tag{5.8}$$

The *baseline complexity count (BCC)*, the number of multiplications required to directly determine each pixel on a scan-line, is then

$$\text{BCC} \equiv M(m,0) = |x|m^2. \tag{5.9}$$

However, a 4 level, $m = 3$ MR reconstruction would require

$$M(m,4) \approx 25|x|m^2. \tag{5.10}$$

One should, however, bear in mind that this estimate is based on direct application of the formulae for the detail and approximation images. The only optimizations used in these computations were a) the calculation of the minimal set of pixel values (those required to produce the initial differences), b) the use of pre-initialized lookup tables to avoid repeated evaluation of the wavelet and scaling function formulae. Closer inspection of these formulae, however, reveals that they are separable 2-D convolutions with wavelet/scaling function kernels. Hence, if one used optimized convolution hardware these values could be determined speedily. However, doing so would mean that *all* the pixel values would have to be produced, rather than just those we need. All the subband convolutions can be computed in parallel; in fact, because of the separability of the wavelet kernels, the computational cost is only slightly more expensive than doing a direct IWT and applying the zeroth level expansion (normalization). If one uses optimized FFT hardware, the difference should be negligible.

The number of DE cycles required to produce a scan-line will depend on the number of eval$n$ instructions which are issued. The *Baseline Cost* (BC) is the number of cycles required to produce (without the use of MR synthesis) a full scan-line worth of data. This value is merely $|x|$ — that is, one cycle to set each pixel (using eval0, cf. Table 5.1). Full MR synthesis (in which an eval$n$ instruction is issued for each span) would require

$$\text{MRCC}(m,L) = |x| \left[\sum_{j=1}^{L} \frac{c_{j-1,m}}{k_{j-1,m}} + \frac{c_{L,m}}{k_{L,m}} \right] \tag{5.11}$$

where

$$c_{j,m} = \begin{cases} (m+2) & \text{if } 2^j > m \\ (1+2) & \text{otherwise.} \end{cases} \tag{5.12}$$

and

$$k_{j,m} = \begin{cases} 2^j & \text{if } 2^j > m \\ 1 & \text{otherwise.} \end{cases} \tag{5.13}$$

The function MRCC(m, L) (Multi-Resolution cycle count) provides an estimate of the number of cycles required to perform full MR scan-line synthesis, under the following assumptions

1. an mth order polynomial can be interpolated in $m + 2$ cycles (see Table 5.1)
2. spans which are too short to provide all the difference information have each pixel set explicitly by means of 'eval1' instructions. Note: one can't use 'eval0' because this turns off accumulation and would overwrite the information for previous resolution levels .

This number is clearly higher than BC, which implies that it is more economical (in terms of instruction cycles) to perform a direct display of the image, rather than using the full multi-resolution reconstruction.

When $L = 4$ and $m = 3$, it requires $\frac{|x|(3+2)}{16}$ cycles to produce a scan-line of the approximation image, about a third of the time required for the full direct approach. Of course, this approximation would be unacceptably blurred, unless the source image did not possess detail on the levels which were ignored (such as a high variance Gaussian or some simple, smooth, geometric shape). However, since such shapes are unlikely to feature prominently amongst the images one wishes to process, simply displaying the approximation image would not be a viable alternative (except perhaps, for a simple overview search in a compressed image database/archive).

If one added only visually significant detail to the approximation image, then there is a chance that the cycle cost would remain below the baseline cost, since many images possess regions in which little variation exists and these could be exploited to lower the instruction count. In addition, since the computation of detail pixels is expensive, the fewer one has to compute the more rapidly the pixel instruction stream can be produced. The issues surrounding such an approach are now discussed.

5.3 Adaptive Synthesis

The cost of computing the full MR synthesis is significantly more expensive than setting each pixel directly. Nonetheless, the MR scheme can provide savings (in terms of DE instruction cycles required to interpolate a scan-line) over the direct approach, provided it is applied in an intelligent manner. By this, I means that, rather than blindly applying the reconstruction, a suitable metric is devised to determine whether such a reconstruction is desirable. The metric proposed here is simply the cycle count (CC) — the precise number of cycles required to interpolate the scan-line using adaptive MR synthesis. The way such a value is derived will be discussed shortly. The CC provides an immediate indication of the desirability of using MR synthesis *for a particular scan-line.* The emphasis on scan-line measures is not only a consequence of the DE's architecture; an image may have regions which contain no detail and others which are highly detailed. If one used a global cycle count, the textured regions might convince one (through the global CC value) to use direct reconstruction on *all* pixel values. However, use of the scan-line measure would ensure that these empty regions were filled at low cost, while the textured regions were set directly. The use of such a criterion ensures that the scan-line cycle count never exceeds the baseline cost — in the worst case, each pixel on the scan-line will be set directly (requiring $|x|$ cycles), and one gains no benefit over conventional display processors.

There are three methods one can use to produce a viable MR synthesis algorithm:

- adaptive generation of detail information
- merging of instructions
- using the constraints imposed by C^1 continuity.

The first of these involves the generation of DE instructions only for those detail regions which a human subject would perceive as relevant.

Instruction merging is appropriate whenever the splines which comprise the various approximation/detail images show little variation from span to span. For example, a scan-line with near uniform intensity could be interpolated in one cycle using a simple zero-degree polynomial. This would provide a tremendous gain over issuing set pixel instructions for each pixel in the scan-line, since each set pixel also requires one clock cycle. One must

be careful, however, that the optimizations made are not too time consuming or lead to inaccuracies. In particular, since the accumulation of several small pixel values can lead to a discernible and important output values, it seems prudent that one only apply such instruction merging to the approximation image and not the consecutive detail tiers. In any event, if the detail selection functions properly, there should be no redundancy worth exploiting in the detail images.

The constraint that the quadratic splines be elements of C^1 has an important implication for us: only the second difference need be changed as we cross a span boundary. The entire scan-line can then be interpolated by a single 'eval3'. Since the the set difference instruction is very cheap (2 cycles, currently), this provides a distinct advantage over issuing 5-cycle 'eval3's for each span.

5.3.1 Adaptive Detail Generation

If the object to be synthesized has little texture or edge information, there will be few wavelet transform coefficients and, consequently, little information in the detail images. Given this scenario, one would like to generate instructions to produce the small information bearing regions in the detail images only. Of course, heavily textured images will have few 'quiet' regions and will be less likely to yield instruction compression using such an approach. In this case, the CC must reflect this and the default (direct) construction method should be invoked.

Assuming, then, that one wishes to perform such an *adaptive synthesis* operation, how does one do this? The answer, not surprisingly, is to use the decoded wavelet coefficients — or more precisely, those which are large. These large coefficients will correspond to prominent texture/edge features in the original image (which is why they were preserved). Having selected a coefficient, the extent of the 2-D wavelet basis which that coefficient weights is determined (from the expressions for their supports, see Equations (5.14)—(5.17)) and a structure is built for each scan-line which contains information on the resulting span extents.

$$\mathrm{supp}\Psi^{[1]}_{j;k,l} = [k2^j, (m+k)2^j] \times [l2^j, (2m-1+l)2^j], \tag{5.14}$$

$$\mathrm{supp}\Psi^{[2]}_{j;k,l} = [k2^j, (2m-1+k)2^j] \times [l2^j, (m+l)2^j], \tag{5.15}$$

$$\mathrm{supp}\Psi^{[3]}_{j;k,l} = [k2^j, (2m-1+k)2^j] \times [l2^j, (2m-1+l)2^j], \tag{5.16}$$

$$\mathrm{supp}\Phi_{j;k,l} = [k2^j, (m+k)2^j] \times [l2^j, (m+l)2^j], \tag{5.17}$$

The structure is established in such a way that overlapping basis elements are merged to produce a single span, where necessary (See Figure 5.1). The structure contains information

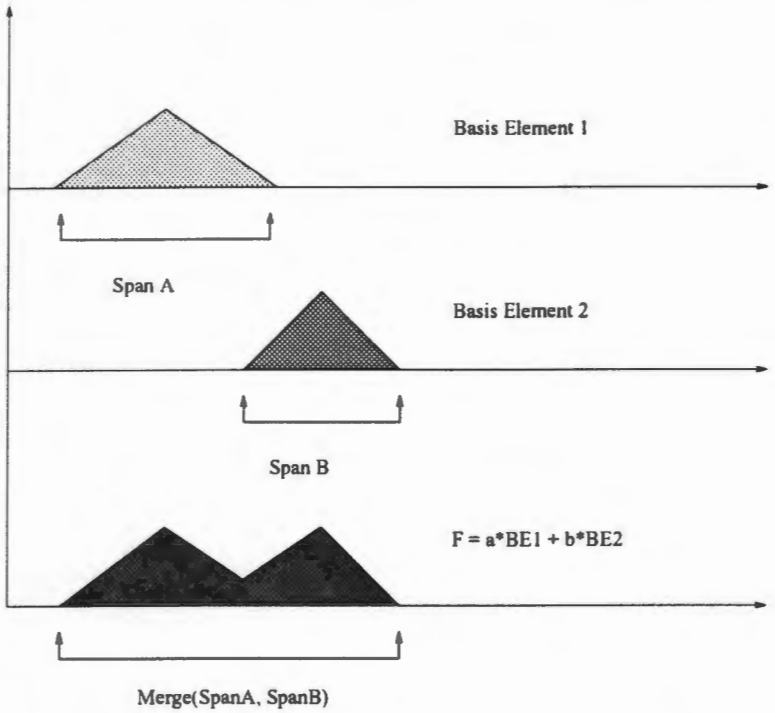


Figure 5.1: Merging of basis elements. When basis functions are summed (to arrive at the output image) the resulting basis elements generally overlap. In this example, two linear basis elements are weighted and summed; the extent of the output function is just the union of the basis element spans. Since many basis coefficients may be zero, the merging operation will generally yield a number of disjoint non-zero regions. If MR synthesis is chosen, DE instructions are produced to generate the non-zero regions intersecting the current scan-line.

on the cost of building each scan-line using the MR approach — it is this information which governs the choice of reconstruction method for each scan-line. No actual function evaluations occur at this stage, since one may wish to opt for direct construction, which does not require these values. If MR reconstruction is selected, based on the CC, then the span lists for each level are traversed and the appropriate instructions generated. For direct construction, the level 0 image is computed from the level 0 approximation coefficients (derived from the inverse wavelet transform). Since some scan-lines may require direct synthesis, the approach used is to perform a complete inverse WT (which can be done quickly) and to proceed from that point.

5.3.2 Instruction Merging

The merging of instructions is dependent on the variation of the differences as one passes across span boundaries. One might also group 'instruction demotion' with this operation i.e., deciding that the order of a spline segment should be lowered because the differences indicate that it has degenerated into some lower order polynomial. In this case one could output a lower cost DE instruction. In the extreme case of a zero-degree polynomial with zero intensity, one would not output an instruction at all.

The span merge proceeds as follows:

1. Compute the differences for the two spans under consideration
2. If the differences are the same within the prescribed tolerance
   2.1 Create a new span with starting point of the old one and
   the extent of both combined.
   else
   2.3 Emit processor instruction to interpolate the (accumulated) span.
3. Fetch next span.
4. If no is span available emit last one and exit else return to 1.

Another reason for only using the span merge on the approximation level is that the likelihood of there being mergeable spans is significantly greater (the image has been low-pass filtered). The possibility of merging instructions is another reason to employ the MR approach: the only merging one can easily do when pixels are set directly is zero-degree merging i.e., merging neighbouring pixels with more or less the same value. This is so because the pixel data may be arbitrarily irregular, unlike the pixel values in the spline MRA which are constrained to lie on a polynomial of known degree — one can merge very large spans provided the differences are more or less the same. For example, if one decomposed, say, $\Phi(\frac{x}{128}, \frac{y}{128})$ (a very dilated quadratic scaling function!) the synthesis/merging procedure would ensure that very few instructions were needed to reproduce this image. The direct approach would have no way of exploiting the special nature of this test image, and would require many wasteful operations in reproducing this simple shape. This is, admittedly, a highly contrived example, but it serves to illustrate the point that the characteristics of the spline MRA allow one to eliminate this kind of 'redundancy' with great ease.

5.3.3 Calculation of the Cycle Count

The decision as to whether a scan-line should be set directly or synthesized from the multi-resolution analysis is based on the value of the Cycle Count (CC). This number provides the precise number of DE instruction cycles required to build the scan-line using adaptive synthesis and is derived as follows:

1. The number of cycles required to interpolate the current scan-line in the approximation image is computed. Each span will require one instruction, at a cost reflected in Table 5.1 (except in the case where the span has insufficient pixels to produce the requisite differences — in this case each pixel is set separately). To reduce the count, 'setddi' instructions are issued on all adjacent spans except the first and an 'eval3' is then emitted to interpolate the entire scan-line.

2. The wavelet bases corresponding to the retained (or largest) wavelet coefficients are examined, and the support of the bases which intersect the current scan-line are found.

3. The number of cycles required to interpolate these (arbitrarily sized) spans are computed as before.

4. The sum of these two values provides the Cycle Count

It should be observed that merging operations are not taken into account in these calculations. Although this seems to be a serious oversight, it was a necessary one: as soon as merging takes place, function evaluations are required to compute difference information and these are very costly in relation to determining span supports. Thus, in cases where merging in the approximation image would serve to lower the CC below the baseline cost, the system would opt for direct reconstruction. However, subsequent experiments revealed that this is unlikely to occur. In general, approximation merging will only be beneficial if the input has substantial smooth regions; when this occurs, the CC will generally fall well below the BC and adaptive reconstruction is invoked. In any event, zero-order merging is always applied to the reconstructed scan-lines when direct reconstruction is used, which will ensure that near uniform regions which slip through are generated more efficiently.

5.4 Results and Discussion

The test images used here were presented in Chapter 4, with the exception of 'circle', 'sugarbowl' and 'hanna', Figure 5.2. In order that one might clearly see the effects of

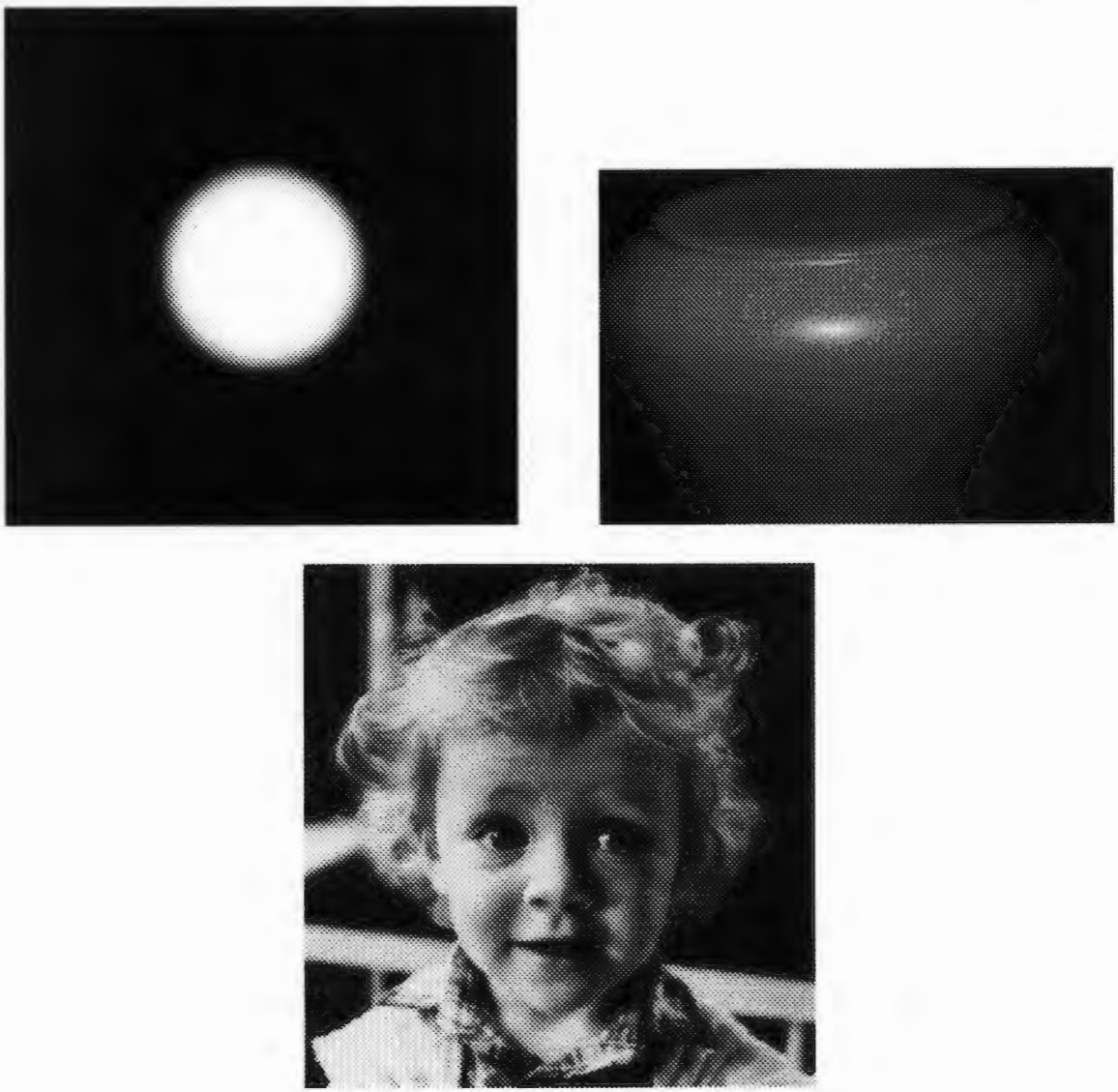


Figure 5.2: The test images 'circle', 'sugarbowl' and 'hanna' (scaled versions).

the MR synthesis operation, which depends on the number of large wavelet coefficients present on each level, the wavelet compression phase was simulated by zeroing all wavelet coefficients below a selected threshold. This threshold was arbitrarily chosen to ensure that

most images were well reconstructed: once a value was selected it was used with *all* the experimental images. The proportion of wavelet coefficients eliminated was recorded, as well as the effects (in terms of instruction cycle reduction) of the multi-resolution synthesis and merging. To provide a quantitative measure of instruction cycle reduction, we define

Image	Kept	MR used	GF1	GF2	Description
Lenna	17.2	No/Yes	1.23	1.22	Complex, texture and edges
hanna	14.5	No/No	1.33	1.33	Complex, uniform regions, edges + texture
square	1.0	Yes/Yes	5.89	4.66	Simple, uniform regions, sharp edges
house	10.3	Yes/Yes	1.63	1.57	Complex, uniform regions, edges + texture
bowl	1.0	Yes/Yes	3.43	3.87	Simple, Phong shaded convex surface

Table 5.2: Compressibility vs image type. The field 'Kept' refers to the percentage of wavelet coefficients retained (by the 'compression'). 'MR used' indicates whether MR synthesis was invoked for any scan-lines; the first entry gives the result for the old instruction costs, while the second provides this information for the new instruction costs. The fields 'GF1' and 'GF2' give the gain factors for the old and new instruction costs, respectively. A three level decomposition was used.

the 'Gain Factor' (GF) as

$$\mathrm{GF} = \frac{|y| \times \mathrm{BC}}{\sum_{i=1}^{|y|} \mathrm{CC}_i}, \tag{5.18}$$

where $|y|$ is the number of scan-lines, BC is the baseline cost and CC_i is the number of instruction cycles required to produce the ith scan-line. For example, a GF of 2.0 means that only half the number of instruction cycles were required to produce the image when compared to direct synthesis. Because of the manner in which reconstruction takes place, $\mathrm{GF} \geq 1.0$.

Examination of Table 5.2 reveals that as the complexity of an image increases, as measured by the proportion of detail coefficients retained for a given threshold, the GF decreases. This observation is only approximately true: an image which has sub-regions with widely varying characteristics may yield a larger GF than an image of 'lower' complexity (fewer detail coefficients above the given threshold). The smoothness of an image determines the degree to which it may be efficiently synthesized since such images possess very little detail. The *nature* of the smoothness is also of importance: if the image is in fact a spline (particularly a low resolution one), then it can be rendered very cheaply. In Low resolution approximation images can be generated very cheaply — this is of interest if one intends to use the DE for progressive transmission. To sum up, then: the gain factors were between

1.2 and 1.7, for images with heavy textures and between 3.0 and 5.0 for images with a high degree of smoothness.

% Kept	Setddi		Eval0		Eval1		Eval3		GF		MR
17.2	0.0	1.5	100.0	97.4	0.0	1.0	0.0	0.1	1.23	1.22	N/Y
10.7	0.0	3.9	100.0	93.1	0.0	2.7	0.0	0.3	1.22	1.20	N/Y
6.9	0.0	4.5	100.0	91.8	0.0	3.3	0.0	0.5	1.20	1.24	N/Y
1.0	62.0	65.7	32.9	14.0	0.2	14.7	4.9	5.7	3.43	3.87	Y/Y
0.3	65.9	79.1	26.5	0.0	3.7	14.7	3.9	6.23	3.39	4.31	Y/Y
0.2	73.4	81.2	16.6	0.0	5.3	12.1	4.7	6.7	3.08	4.57	Y/Y

Table 5.3: The effect of detail thresholding (compression) on MR synthesis. The left and right columns of each entry provide data for the old and proposed instruction costs, respectively. A 3 level decomposition was used with the thresholds 0.01, 0.05 and 0.07. The first sub-table gives the data for 'Lenna', while the second provides information for 'Sugarbowl'.

Table 5.3 provides a clearer indication of the role which compression plays in rendering gains. One would expect the GF to rise steadily as the number of detail coefficients decreased — however, this behaviour is not immediately apparent. The reason for this somewhat counter-intuitive state of affairs can be found in the transition of directly rendered scan-lines to those generated by MR synthesis. This occurs because the synthesis decision calculations do not take merging into consideration, and often the directly rendered scan-lines (which have zero-degree merging applied to them) can be produced more efficiently than those generated by MR synthesis. However, the discrepancy is normally quite small; the trade-off between function evaluations and accuracy seems acceptable. Of course, if all the MR images are computed then additional merge checking could be implemented without great cost. In the case of 'sugarbowl', when the cheaper instructions are used, the GF does indeed increase as the number of wavelet coefficients decreases.

It should be remembered that the gain factors given above are for reconstruction of the *entire* image; as already stated, approximation images can be computed very efficiently.

Since the DE was not designed for the express purpose of manipulating MR analyses, there are some inefficiencies in the manner in which the algorithms used render this structure. In particular, 'set*' instructions cannot be reset once they have been issued, and this has the effect of forcing one to issue unnecessary 'set' instructions.

There is a strong connection between the GF and the number of levels involved in the decomposition. Table 5.4 provides some data to quantify this observation. There are two

Cost	2	1	1	1	3	1	5	3			
Level	Setddi		Eval0		Eval1		Eval3		GF		MR
2	0.0	1.4	100.0	96.9	0.0	1.6	0.0	0.1	1.24	1.24	N/Y
3	0.0	1.5	100.0	97.4	0.0	1.0	0.0	0.1	1.23	1.22	N/Y
4	0.0	0.9	100.0	98.7	0.0	0.4	0.0	0.0	1.23	1.23	N/Y
2	75.3	74.7	22.9	3.5	0.1	19.7	1.7	2.1	2.62	3.38	Y/Y
3	62.0	65.7	32.9	14.0	0.2	14.7	4.9	5.7	3.43	3.87	Y/Y
4	58.3	63.2	32.2	16.5	0.2	11.3	5.6	9.0	3.1	3.51	Y/Y

Table 5.4: The effect of decomposition level on MR synthesis. The cost indicated above is the time required (in clock cycles) to execute the indicated instruction: the first column provides the old instruction cost, while the second provides the proposed instruction cost. Data is given for both cases. The values in each column are the percentages that each instruction type contributed towards the total instruction cycle cost. 'GF' stands for gain factor (as described in the text) and 'MR' indicates whether MR synthesis was invoked during image reconstruction. Note: if MR is 'Y', this does *not* imply that MR synthesis was used for every scan-line — only those for which it was more economical. 'N' means that no scan-line could be synthesized more cheaply. The same threshold — 0.01 — was used to generate all the data. The first sub-table gives data for 'Lenna', while the second gives data for 'Sugarbowl'.

conflicting costs which have to be balanced when one decides on the optimal level of the decomposition:

- as one increases the number of levels, more detail has to be accumulated, with a commensurately higher instruction cost
- when few levels are used, the approximation image will require many more instructions to generate (because the polynomial spans are smaller) and there is little chance of polynomial merging occurring.

Of course, the complexity of the image will also have an important role to play. It is evident from the data in Table 5.4 that Lenna and Sugarbowl (images with a great complexity difference) respond differently to the number of levels used in the decomposition. For Sugarbowl, a 3 level decomposition seems best. For Lenna, however, a zero level decomposition seems optimal — because of the preponderance of texture. Although one cannot extrapolate from two examples, experiments with a wider class of images indicated that a level 3 or 4 decomposition was preferable to one with fewer levels.

Direct pixel merging plays an important role in the magnitude of the GF for most images. There are several reasons for this. Most of the images have sizable uniform regions, which

are easily exploited by (direct) zero-degree merging. Higher-order span merging is less successful because the complexity inherent in the images leads to widely varying differences across neighbouring spans on all but the lowest levels. Natural images are generally too complex to allow any sort of real gain from a pure multi-resolution approach, having too many wavelet coefficients at lower levels. However, if the image to be decomposed consists of polynomial primitives, such as the Phong shaded images referred to earlier, the gains can become significant since the high resolution detail tiers may be entirely discarded. In addition, there is also opportunity for higher level span merging, resulting from uniform polynomial segments intersecting several consecutive spans. Since these are precisely the kind of images the DE was designed to synthesize, the usefulness of MR synthesis and merging should be evident. Furthermore, such images will have few wavelet coefficients and will not require many detail evaluations, thus lowering the computational burden and allowing more rapid instruction generation.

Merging is not without its pitfalls, however. In particular, if neighbouring spans with dissimilar differences are merged, the resulting interpolation errors can become acute. This problem is exacerbated by consecutive span merges, since the resulting span may be very long. For the purposes of the polynomial merge, the difference were required to be identical within the accuracy provided by a single precision float, except in the case of direct pixel merging. In this case, all pixel values within one gray-scale of the of the chosen pixel would be approximated by that pixel, using an 'eval0' of the appropriate length.

5.4.1 Alternative Architectures

A 2-D Difference Engine

The 2-D spline multi-resolution analysis ensures that detail and approximation images on level j have constant polynomial character over patches with with support

$$[2^{\{jj-1\}}k, 2^{\{jj-1\}}(k+1)] \times [2^{\{jj-1\}}k, 2^{\{jj-1\}}(k+1)]. \tag{5.19}$$

The semicolon is used to differentiate between the approximation and detail images, respectively. This coherence is not exploited by the synthesis algorithm, because the DE is inherently one dimensional. One could, however, introduce a 2-D Difference Engine i.e., one which performed interpolation across scan-lines as well. Of course, the output would still

have to be a 1-D pixel stream — one could thus maintain the DE and precede it by some sort of 'Y-processor', which would utilize similar logic to provide scan-line interpolation. The 2-D DE would interpolate a rectangular region given the necessary x and y differences and input intensity (for the top left hand corner of the rectangle), outputting a scan-line's worth of DE instructions after processing every new y value. To interpolate a quadratic patch, one would need to specify *nine* differences,

$$\Delta_x,\ \Delta_y,\ \Delta_{xy},\ \Delta_{yy},\ \Delta_{xx},\ \Delta_{xxy},\ \Delta_{yyx},\ \Delta_{xyy},\ \Delta_{xyx},$$

the starting co-ordinates, the starting intensity value and the x and y dimensions — 14 pieces of information in all. To interpolate a square of size $n \times n$, we would require $5n$ pieces of interpolation using the DE with quadratic splines. The gain becomes more significant with larger block sizes. Of course, the amount of information one has to provide is not the only measure of gain that should be employed. The number of cycles required to execute the instruction would be the real indicator of success. Obviously, for small patches, a straight DE interpolation would perform better — one could devise some sort of pass through mechanism, which would allow the previous instruction set to function in the way it had before, just forwarding the instruction straight to the scan-line DE.

The major limitation with such a scheme is the rapid build-up of interpolation error, resulting from the lack of additional precision bits. Quadratic interpolation error in 1-D grows as $O(n^2)$ — in 2-D this would be $O(n^4)$, limiting the size of the region one could accurately interpolate (with 24 precision bits) to a square of size 64x64. Of course, one could increase the number of precision bits, but if the MR decomposition does not proceed beyond $j = 4$, this would be unnecessary — the patches on this level are of size 16x16 (this discounts merging operations, though, which could increase the effective span support substantially). Assuming that the number of precision bits (and the precision of the input) were up to the task, one could (in theory) interpolate the entire image with *one* Y-processor instruction (once again, this is a contrived but illustrative example). Before one could deliver a meaningful decision as to the viability of such a scheme, a full analysis would have to be undertaken.

Direct Basis Accumulation

The expressions for the support of the wavelets and scaling function (Equations (5.14)—(5.17)) show that there are only a limited number of basis functions for each resolution — the three wavelets and the scaling function. The basis values can be generated by multiplying the appropriate functions on the appropriate resolution level and the resulting intensity information for each x cross-section can be entered into a table indexed by resolution, orientation and the position of the cross-section in the 2-D basis. A special instruction could be implemented which could multiply each intensity value in a specified cross-section and accumulate it at a given location in the scan-line accumulator — like an 'eval0', but with span data determined from the table. This basis data could be operated on in parallel (the multiplications, shifting and accumulation would be done in parallel).

Image synthesis would proceed in a similar fashion to that proposed earlier in this chapter:

1. the wavelet coefficients would be used to generate a scan-line list containing information about bases which overlap each scan-line (i.e., the resolution, type and cross-section identifier, the basis coefficient and location of the basis on the scan-line.)

2. each scan-line would be traversed and each basis cross-section would be generated by issuing the special 'eval' instruction, which would accumulate the weighted basis at the appropriate location in the accumulator in a very short period of time.

The new instruction would probably require 3 clock cycles, one to do the multiplications, another to shift the intensity data to the appropriate point and the last to accumulate the data. In addition, there are *three* wavelets which have to be accumulated when generating detail, which means that 3 of these instructions would have to be issued to accumulate detail which was present in all orientations. Nonetheless, since the time required to generate each basis is independent of the resolution, and the low resolution bases will be far more numerous than high resolution bases, there would be marked gains (Most of the high resolution bases would be removed by the compression phase). In addition, it would be even cheaper to produce approximation images than in the scheme proposed earlier.

In this scheme, there is no need to interpolate splines — or even do interpolation, for that matter. Indeed, the scheme is more general in that arbitrary wavelet (or other) basis can reside in the table: orthogonal, bi-orthogonal etc., provided they have compact support.

An option could be included to load the table with new wavelet data, to allow maximum flexibility.

5.5 Concluding Comments

Adaptive multi-resolution synthesis offers a means of exploiting the structure of the wavelet compressed images — or more precisely, the associated MRA — and the architecture of the Difference Engine. While the gains attainable using this approach depend on the complexity of the image involved, the inclusion of a span-merging procedure ensures that one can *always*[4] achieve some sort of reduction over the baseline cost. While the computational overheads involved in direct computation of the various multi-resolution images seems, at first glance, to be excessive, fast convolution schemes can be designed (and implemented in hardware) to ensure that these calculations can be performed efficiently.

The decision governing the choice of image synthesis (direct or multi- resolution) is made independently for each scan-line, providing a simple but effective kind of image adaptivity. The number of processor cycles required to render a scan-line is bounded above by the baseline cost: under no circumstances will the BC be exceeded. In addition, to ensure computational efficiency, no function evaluations occur until the relevant type of reconstruction has been selected. The calculations required to make this determination are straightforward and do not require much CPU time.

To improve rendering performance, the spline continuity is used to ensure that low cost *set difference* instructions are used rather than full interpolation instructions, where possible. Furthermore, the possibility exists to improve rendering efficiency by exploiting scan-line coherence i.e., to introduce a processor which performs rapid 2-D interpolation. Alternatively, one could opt for a more general image synthesis approach, which would not tied to any particular image representation. While there are many issues which would have to be examined before such a device could become a reality, the ideas raised above certainly bear closer scrutiny.

[4] Except in extreme pathological cases!

Chapter 6

Second Generation Image Coding

There are a variety of methods for achieving image compression, ranging from the exotic to the trivial — some of these were discussed in Chapter 4. These 'Classical' image coding schemes are known as *first generation* techniques. Included in this category are (differential) pulse-code modulation, delta modulation, predictive coding and transform coding. Such methods make little or no attempt to use any but the most trivial properties of the human visual system (HVS). These techniques may be lossy or lossless. When low bit-rates are required, lossy coding must be employed, since lossless techniques are limited by their requirement of perfect reconstruction.

Second generation methods are those which are specifically designed to exploit the inadequacies of the HVS, such as its insensitivity to high-frequency noise. Wavelet sub-band coding, as discussed in [1, 25, 15] falls into this category. Such compression schemes attempt to simulate the action of visual processing i.e., to extract and retain only visually relevant information. Wavelet-based schemes are not alone is in this regard; Watson [50] has developed another such scheme, which uses the suggestively named *cortex transform* to model the action of the visual cortex, the region of the brain in which visual information is processed. There are many other examples of HVS models (see, for example, [44, 17]); one might almost say that any worthwhile image compression scheme will be built around some sort of HVS model. Kunt et al [23, 24] explores this topic in great detail.

The method used to compress the wavelet transform coefficients — MMSE vector quantization — proved unsatisfactory in that it failed to adequately exploit the compressibility of the wavelet representation (Section 4.5.2). This can be attributed, at least in part, to

the use of too simple an HVS model. The following sections are aimed at developing a satisfactory means of exploiting both the redundancy inherent in the transform and the nature of the HVS to achieve a higher level of compression performance. The discussions presented here should be considered as preliminary to future work.

Section 6.1 provides a simple introduction to edge detection techniques. The manner in which edges may be constructed from these points, and efficiently represented, is discussed in Section 6.3. Edge-based image reconstruction is covered in Section 6.4. An alternative coding scheme, geared towards the Difference Engine, is suggested in Section 6.5 — Section 6.6 presents the results of some simple investigations into facets of the proposed scheme.

6.1 The multi-scale edge characterization of images

There is wide consensus on the important role that edges play in our interpretation of visual information. Marr [34] suggested that the brain extracts a skeletal edge representation of an image as a prelude to higher visual processing — the so-called *primal sketch* [34, 35]. *Marr's conjecture*, which holds that all the information necessary to fully describe an image may be extracted from the edge information present on all scales, is a formalization of this idea — see Section 6.4.2. The Conjecture is intuitively appealing: images are made up of edges and texture — however, upon closer inspection, texture is subject to the same analysis i.e., it too is composed of edge and 'texture' information. This process may be repeated until no additional detail is discernible. Such an edge-map encoding would seem to offer a means of obtaining a very compact image code, given the sparse nature of such a representation. Of course, one first has to extract the edges — a task which will be examined in the following sections.

6.2 Edge detection methods

Many of the algorithms used to detect edge-points are unexpectedly simple. It is this very simplicity which underlies the major difficulty inherent in edge detection viz., which edge points lie on 'legitimate' edges and which are merely the by-products of computational or image noise? In fact, the problem may be worse, particularly if one is aiming to develop a

compact image code. In this case, besides the legitimacy issue, the whole question of *relevant* edge selection comes to the fore. After all, one does not wish to maintain unnecessary edge information in such a representation. Unfortunately, the latter problem is, in general, one which cannot be automated i.e., it calls for some measure of subjective judgment. One can attempt to classify edges based on obvious criteria such as edge length, the average grey-scale value along the edge etc. but all of these fail under appropriate conditions. Nonetheless, while a general solution is (presently) unattainable, one can still use the standard methods to obtain a generally satisfactory edge decomposition.

Edge detection is a two phase procedure consisting of

1. the determination of edge points and
2. linking these points into edges or contours.

The latter problem will be considered in Section 6.3; the former is addressed now. In the following sections the phrase 'edge detection' is used rather than 'edge point detection', since this is the usage most prevalent in the literature.

6.2.1 Standard approaches to edge detection

Before any meaningful attempt can be made to extract the edges from an image, one must have a clear idea of what an edge is. An *edge point* is one at which the intensity undergoes maximum change as one moves in a prescribed direction. The totality of all such simply[1] connected points, for the selected direction, is said to constitute an *edge* with a direction *perpendicular* to the direction of maximum change. (See Figure 6.1). This kind of *directional* edge detection is accomplished (in the continuous domain) by taking the *directional derivative* of the image function, and checking to see whether the point under consideration maximizes this function for the given direction. If so, it is a point in the edge with the prescribed gradient. For example, $\frac{\partial I}{\partial x}$ provides a means of detecting vertical edges.

In many cases, however, one wishes to use an edge-detector which is *isotropic* i.e., which does not perform directional edge detection. In this case, one can use a *gradient-based* operator. This involves the computation of the magnitude of the 2-D gradient operator, $(\nabla I)(x, y)$ defined as

[1] By this I mean that an edge point will have a maximum of *two* other points connected to it. Naturally, the end point of an edge will only have one neighbouring edge point.

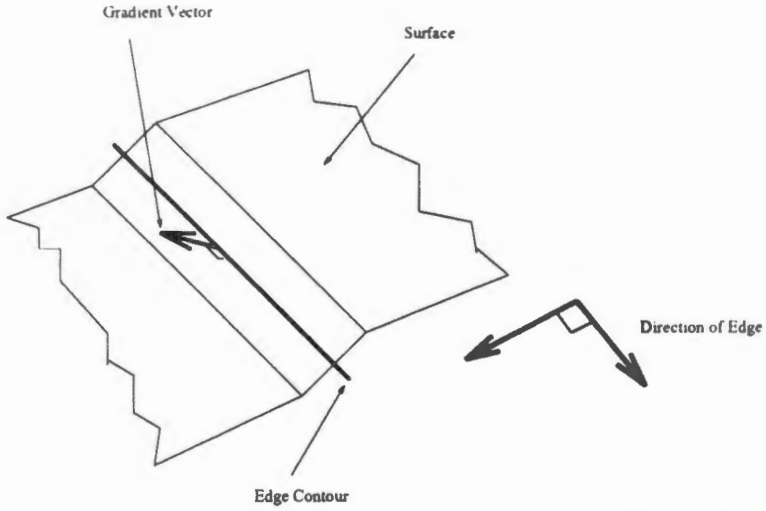


Figure 6.1: Edge detection. The directional derivative attains its maximum in the direction of the gradient vector (which is perpendicular to the surface under consideration). The edge direction is perpendicular to the direction of this vector. Edge points are located at the centre of an edge, since this is the point at which the intensity change attains its maximum.

$$(\nabla I)(x,y) \equiv \frac{\partial I(x,y)}{\partial x}\hat{i} + \frac{\partial I(x,y)}{\partial y}\hat{j} \tag{6.1}$$

The magnitude value is then thresholded, and all points above a prescribed threshold are considered edge points. The problem with this approach is that of broad edges: all the edge points above the threshold are selected, not only the maximal ones. This problem can be resolved by *edge-thinning* techniques [26].

Since the images one works with are composed of discrete samples, the partial derivatives have to be replaced with finite difference approximations. The precise manner in which the derivatives are approximated determines how successful the edge detection phase will be. A number of edge detectors [26] have been developed for images with varying requirements.

An alternative to computing the first (directional) derivative is to use a second derivative formulation. In this case, an edge point is one which produces a zero in the second derivative, rather than maximizing the first derivative [6, 35]. One can use the *Laplacian operator*,

$$(\nabla^2 I)(x,y) \equiv \frac{\partial^2 I(x,y)}{\partial^2 x} + \frac{\partial^2 I(x,y)}{\partial^2 y} \tag{6.2}$$

to provide an isotropic second derivative.

Because the finite difference operations are sensitive to noise, Canny [6] precedes the gradient operation by Gaussian smoothing. This serves to de-sensitize the detection process and leads

to more accurate edge extraction. Marr and Hildreth [35] use a similar approach, although in this case the Laplacian formulation is used instead of maxima detection. Now however, in keeping with Marr's conjecture, the edge detection becomes a multi-scale process. Gaussians of differing variance are used to smooth the image before the detection phase; important edges are considered to be those which persist over several scales. The detection and smoothing phases may be combined and treated as one filtering operation. The impulse response of the resulting convolutional operator then has the shape of a 'mexican hat'.

6.2.2 Wavelet based edge detection

The connection between wavelet transform coefficients and edges has been stressed repeatedly throughout this dissertation. These coefficients are large in the vicinity of rapidly changing intensity values, precisely the sort of behaviour exhibited by edges and texture regions. However, one cannot construct a wavelet edge detector in an arbitrary manner. The construction in [33] is based on a cubic spline wavelet. The scaling function is chosen so that its integral is unity; the wavelet is obtained from the scaling function by differentiation:

$$\int \phi(x)dx = 1, \tag{6.3}$$

$$\psi(x) = \frac{d\phi(x)}{dx} \tag{6.4}$$

Since the wavelet transform may be re-written as a convolution [10], and convolution 'commutes' with differentiation, one may write

$$(W_\psi f)(s,x) = (f * \psi_s)(x) = s\frac{d}{dx}(f * \phi_s)(x) \tag{6.5}$$

where s is the scale and $\phi_s(x) \equiv \frac{1}{s}\phi(\frac{x}{s})$. Thus, in 1-D, the wavelet transform defined w.r.t. to the wavelet $\psi(x)$, is equivalent to a Canny edge detector [6] (the scaling function assumes the role of the smoothing gaussian). To find edge points one searches for wavelet transform maxima. In fact, it is advantageous to use the modulus of the wavelet transform for this detection, since this detects only sharp variations in the signal.

The above generalizes in an obvious way to 2-D, where we have

$$\begin{pmatrix} (W_{\psi^1} f)(s,x,y) \\ (W_{\psi^2} f)(s,x,y) \end{pmatrix} = s \begin{pmatrix} \frac{\partial}{\partial x}(f * \phi_s)(x,y) \\ \frac{\partial}{\partial y}(f * \phi_s)(x,y) \end{pmatrix} = s\nabla(f * \phi_s)(x,y) \tag{6.6}$$

with $\psi^1(x,y) = \frac{\partial \phi(x,y)}{\partial x}$ and $\psi^2(x,y) = \frac{\partial \phi(x,y)}{\partial y}$.

Thus, edge point can be detected from the components of (an appropriately defined) 2-D wavelet transform. This formulation has the advantage that the wavelet transform components also give explicit information about the *orientation* of the edge [33]. The modulus of the component vector serves as an indication of of *edge strength*. This information is very useful when attempting to resolve the difficulties referred to at the beginning of Section 6.2.

6.3 Contour coding

If one wishes to derive an image coding scheme based on edges, then detection of edge points alone is insufficient. These points must be associated with the edges to which they belong, in order that one may code these edges. If this is not done effectively, the resulting contours may be highly fragmented. Fragmentation of contours has two major consequences:

Inefficiency the effectiveness of subsequent contour representations is degraded,

Information Loss there may be an unacceptable loss of information, since short edges are often discarded in favour of a more compact edge coding.

There are a number of situations in which contour coding will be desirable [20, 22]. The most obvious is when the image itself consists of line/contours only — for example, circuit diagrams, iso-surface maps etc. In these cases, the kind of redundancy in the image makes this sort of coding a natural choice.

6.3.1 Edge point Connectivity

An edge point may be *four-connected* or *eight-connected*. In the former case, the point may only bind to the four neighbours which are in direct contact with it. In the latter case, it can also connect to the four points which only touch it diagonally (Figure 6.2:A). The choice of connectivity can affect both the accuracy of the contours produced and the amount of processing required to construct them. If one opts for four-connectivity, then a sequence of diagonal edge points would not be merged (Figure 6.2:B), unless 'filling' edge points were inserted to connect them. An eight-connected chaining scheme would have no problem with

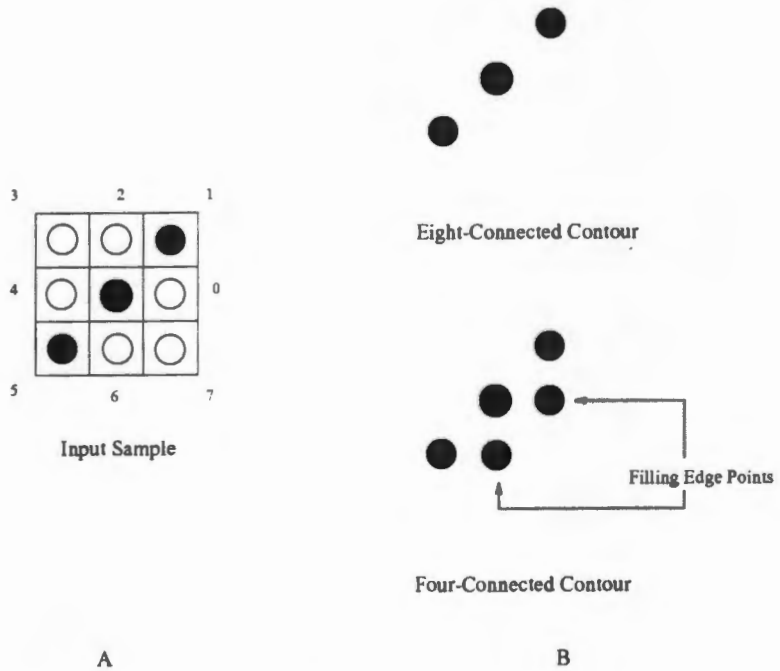


Figure 6.2: Connectivity. A point is a digital image is surrounded by eight neighbours (A). Points 0, 2, 4 and 6 are four- connected to the centre point. Points 0 through 8 are eight-connected to the centre point. The implications of connectedness are easy to see: a sequence of diagonal points can be easily joined when they have eight-connectedness. When the contour building algorithm is based on four-connected techniques, however, special 'filling' points have to be inserted before such a contour can be built. This introduces a small measure of inaccuracy and makes edges seem thicker (B).

such a sequence of edge points. However, one then has to do more work, since additional directions have to be searched to determine potential edge points.

6.3.2 Edge point chaining

If there are several points to which a given edge point can be connected, some sort of logic must be invoked to ensure that the most likely candidate is chosen. There is no fool-proof method of doing this. In general, the only information available to assist in making this decision is the spatial position and grey-scale value attached to the edge pixels. Such information is of limited use in a noisy environment. Clearly, some other means of identifying legitimate edge points must be employed.

Such a scheme is presented in [33]. In this system, candidate points are only chained if their respective strengths and directions are within a prescribed tolerance. The magnitude of the wavelet transform is used as the measure of edge strength. The direction of the (tangent to the) edge is obtained by noting that the gradient vector is perpendicular to the edge contour. The direction of the gradient vector can be easily determined from the wavelet coefficients — see Section 6.2.2. Thus, by determining the disparity in the direction of the gradient vectors of the two candidate points, and checking the wavelet magnitude associated with each point, a choice can be made as to whether they are logically connected.

A problem which arises in this scheme (and in general) is that *quantization* of edge directions occurs, regardless of whether the edge points are eight- or four-connected. In other words, while the tangent to the 'true' contour may vary smoothly, the number of edge directions is (generally — see [22]) limited to four or eight. However, since one can only extract approximate edge positions anyway, this objection is more aesthetic than practical — provided one uses 8-connected chaining. As mentioned above, four-connected chaining imposes some irritating constraints. Nonetheless, there may be situations in which a four-connected chaining scheme is preferable.

6.3.3 Edge-Chain coding

Once the edges (or acceptable approximations thereof) have been extracted, some means must be found to encode this information in as compact a form as possible.

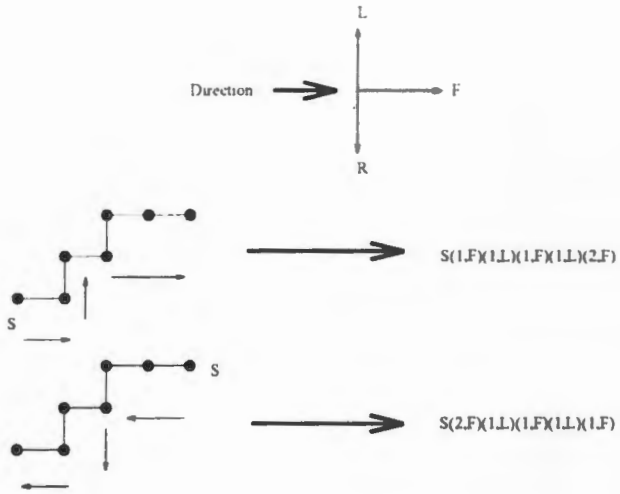


Figure 6.3: A four-connected chain code. A contour may be described non-uniquely. In the first diagram, the contour is traversed left to right, while in the second it is traversed right to left. Three (relative) directions are used, rather than four absolute directions.

The standard approach is *contour run-length coding* (CRLC) [20, 22]. As the name suggests, this method steps along the contour, emitting pairs of the form (n, dirn), where n in the number of occurrences of the unit interval in the direction 'dirn'. Of course the starting point of the contour must also be stored. The directions referred to are *relative* i.e., they are given in relation to the previous contour direction vector. For example, using four-connected chaining, one would have *three* relative directions; to go 'backwards', you would start with a reversed direction vector and proceed as before. See Figure 6.3. In many cases, the edge chaining and contour coding are combined in one step. In this case, edge searching algorithms [22] are employed to resolve ambiguities (although somewhat contentiously).

CRLC, and its generalizations (See, for example, [22]), work well for edge segments with long runs, such as those encountered in circuit diagrams. However, for the arbitrary kinds of edges one is likely to encounter in natural images, the overhead of storing a number with each direction symbol is likely to cause a loss of efficiency (even if one employs an entropy coding to reduce the size of the data). An alternative approach is suggested in [7]. Rather than coding runs directly, the contour is blocked into into segments of B edge points. The possible (relative) directions are enumerated and the direction symbols generated when traversing the contour are Huffman coded (Figure 6.4). Although this method will not, in general, achieve the same performance as the algorithm used in [22], it is computationally cheaper and easier to implement. For small block-sizes the limitations on contour torsion (imposed

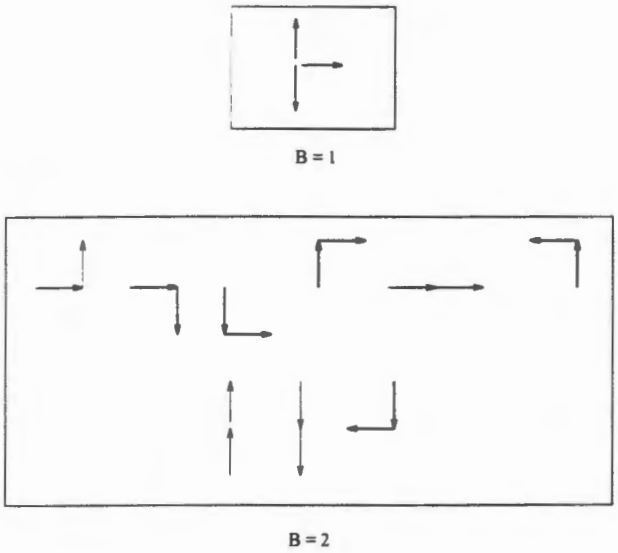


Figure 6.4: Edge blocks for block sizes $B = 1, 2$. When using a four-connected code, there are $\sum_{i=1}^{B} 3^i$ possible blocks codes for a given size B, since a coding scheme based on B-sized blocks needs all the smaller block codes to deal with contour end-points.

by the small blocks) allow rates of around 1.3 bits per contour pixel to be attained (using $B = 3$). This performance is, however, for four-connected contours. For eight-connected contours there are many more possibilities and coding efficiency rapidly decreases.

6.4 Contour reconstruction

The above process generates a series of (compactly) represented edges or contours. These contours may be used as the sole image representation, in which case one is still able to extract important contextual information (providing the encoding has not deleted too many edge segments). Alternatively, some sort of interpolative method may be used to produce a smoother, more visually appealing image.

Two methods will be described here. The first, developed by S. Carlsson [7], uses only directly available information (i.e gray-scale and geometric information) to perform image reconstruction. The second method, that of Mallat and Zhong [33], uses wavelet edge detection and chaining, as discussed above. In addition, rather exotic use is made of the information contained in the wavelet transform to enable image reconstruction.

6.4.1 Carlsson's approach

The contours extracted in [7] are four-connected (because of the edge chaining algorithm used). The edge points themselves are located on half-pixel points; rather than storing this non-integral value, the grey-scale value is encoded by keeping the two pixel values which lie on either side of the 'virtual' edge point. The difference in these values is a measure of edge strength.

The geometry of the edges is stored using the block coding scheme referred to above. Grey-scale information (edge strength) is compressed by exploiting the slow variation of grey-scale information along a contour. The starting positions of edges are coded using a form of run-length coding: the distance between consecutive starting positions is coded, rather than the actual x, y co-ordinates. These runs are then Huffman coded.

The contour interpolation is posed as a variational problem: minimize the variation of the surface constrained by the grey-scale values of the contours. This turns out to be equivalent to solving of a non-linear quadratic PDE [7, pg. 61]. A finite difference equation can be set up and solved via *successive over-relaxation*. This is an iterative approach, in which the grey scale values on the contours (which retain their value throughout) 'diffuse' outward to fill the regions between the contours. In general, eight iterations is sufficient to achieve an acceptable level of smoothness.

Unless the edge representation is very rich (in which case compression performance will be degraded) the image reproduced will suffer from a major loss of texture information as well as minor, but visually irritating, edge loss. To overcome this, Carlsson uses Laplacian image coding to code the residual (the difference between the input image and the reconstructed one). Because much of the high-frequency information (the strong edges) has been subtracted from the image, one can code the residual fairly compactly without sacrificing intelligibility. The price paid for this gain will be the loss of small scale texture. The overall compression ratios are not particularly noteworthy when good reproduction is required, seldom exceeding 10:1.

6.4.2 Multi-scale edge reconstruction

While the method proposed in [33] is conceptually similar to that described in Section 6.4.1, the implementation is radically different. The edge detection and chaining algorithms are

based on the modulus and direction of the wavelet (gradient) vector, produced by the special two-wavelet transform mentioned earlier. This approach already serves to distinguish the two methods; the conceptual correspondence becomes even more hazy when the reconstruction algorithm is examined.

Taking the message of Marr's conjecture to heart, Mallat and Zhong derive an inverse (reconstruction procedure) based on the multi-scale edge information extracted, which produces a very close approximation to the original image. That is, the inverse wavelet transform is modified to deal with the edge representation.

In their system, a series of angle and modulus images are generated from the input image. These 'images' represent the decomposition of the gradient vector (the wavelet transform) into its polar components. The extracted edges have their geometry encoded using the method introduced in Section 6.4.1. To make the representation efficient, only the modulus/angle values along edges should be stored. The modulus values are coded using predictive coding (they are assumed to vary slowly along the edge). The angle values are not stored at all — they are only used in the edge chaining process. To recreate them, the tangent to the edge curve at each point is approximated (using simple geometry) and the gradient vector direction is assumed to be perpendicular to that of the tangent vector. Since the wavelet transform is only taken over a finite set of scales (as discussed previously) the information at smaller scales will reside in the approximation image, which is then also part of the representation and must be stored to enable reconstruction.

Since the detail images are now encoded in the edge representation, some means must be developed to perform an inverse WT based on this representation. Analysis of the problem reveals that, in order to perform the inverse transform, one must find the element, of a sub-space of wavelet transforms, which minimizes a smoothness criterion (i.e., it must minimize a particular 'Sobolev' norm on this sub-space). It turns out that one can define two projection operators, P_V and P_Γ to determine this element. The former is merely the composition of the wavelet transform/inverse transform operators, while the latter modifies its argument by adding on sections of exponential curves (See the Appendix of [33]). When these projection operators are alternated, using the zero function as the initial 'point', the output converges (with a fair degree of certainty[2]) to the desired element of the wavelet sub-space.

[2]Certain approximations are made which affect the stability of the convergence; however, the authors insist that they have yet to encounter an image for which their system would not function adequately.

When image compression is desired, the edge representation has to be made significantly smaller. In [17], a three level wavelet edge decomposition is used, but rather than coding the edge maps for all three levels, only the second level map is retained. This is used to approximate the other two. As in Section 6.4.1, edge culling leads to loss of texture and edge information. The problem is resolved in much the same way as before — a residue is generated and subjected to an image compression scheme. Now, however, orthogonal wavelet compression is applied, a method noted for its texture preservation at fair compression ratios. Intelligible results are obtained at very high compression ratios (of the order of 100:1), although the image is then of poor quality.

6.5 An alternative proposal for edge interpolation

There are a number of problems with the edge reconstruction techniques discussed above. These may be summarized as follows:

Multi-scale edge reconstruction

1. The number of calculations required is excessive. Although the projective operators are $O(n \log n)$ these operators must be iterated several times in order to get good convergence. Each iteration requires a wavelet transform and inverse transform as well as a series of exponential calculations. Running on a reasonably powerful workstation (SGI R4000 based Indigo^2) it takes several minutes to complete the recommended eight iterations. Admittedly, the operations could be made more efficient, but even if one could improve performance, the complexity of the calculations makes this approach unsuitable for real-time hardware implementation.

2. The existence of the approximation image places a fundamental limit on the achievable compression ratios. This image is required to make the edge representation complete, given the finite nature of the wavelet decomposition. There is a tradeoff involved here: the more levels one decomposes, the smaller the approximation image, but the more sensitive the resulting wavelet coefficients are to quantization. It should be remembered that wavelet coding the residual will introduce another approximation image.

3. It is a non-trivial problem to determine which edges are visually relevant (for reconstruction). The more edges one retains, the better the reconstruction, but the less compact the code. This is a problem common to all such edge representations. It seems best to allow a user-specified threshold (on edge length, say) and experiment until a suitable compromise is reached.

4. The storage requirements are fairly substantial, since each detail image is now replaced by an angle image and a modulus image. One can improve memory utilization, but this complicates matters, particularly for hardware implementation.

Edge interpolation

1. The use of a four-connected chaining algorithm leads to the problems discussed in Section 6.3.1.

2. The approach used to code the starting positions of edges (a run-length scheme) is unsatisfactory, since there is no guarantee that edge starting points will be nicely distributed across the image — Huffman coding could then prove extremely inefficient, given the overhead of maintaining the Huffman table.

3. Without the information provided by the wavelet transform, it is more difficult to chain the correct points together.

4. The residual is compressed using a Laplacian compression scheme, rather than wavelet compression.

From the above one can begin to see what form the alternative approach should possess. At the very least, the residual should be subjected to wavelet compression, preferably augmented by some sort of HVS image model — the scheme discussed at the end of Chapter 4 would fulfill this requirement.

Since the existence of the approximation image in the multi-scale edge representation places a fundamental limit on the attainable compressibility, and the computations involved in image reconstruction are excessive, the multi-scale compression scheme envisaged in [17, 33] is not viable (for us). Although Carlsson's approach utilizes a very simple edge interpolation

scheme, it is also iterative and multiple sweeps are required before an acceptable approximation is produced. Ideally, one would like some sort of simple, non-iterative interpolative technique.

An important part of this new approach, will be the manner in which it produces edge-points and chains them together. From the above, it is clear that Carlsson's recommended way of coding chain starting point information is less than satisfactory: it is not clear that Huffman coding the sequence of runs for an arbitrary image will produce a more compact representation; perhaps it would be best to exclude this aspect of the coding. Wavelet edge chaining techniques seem more intuitively appealing and accurate. The most important requirement, however, is that the strong edges are extracted from the image — even a simple scheme should be able to manage this. Nonetheless, given the advantages of a wavelet edge detector, it remains the method of choice. In any event, the existence of the residual image ensures that we will catch any coding errors which emerge in this stage.

Since the interpolated image will not be used as a stand-alone visual representation, the interpolation does not have to be accurate. However, to ensure that we do not introduce sharp edges in the residual, the interpolation across edges must maintain a certain amount of continuity.

An important concern in our case is the manner in which such a compression scheme would affect Difference Engine performance. The performance gains reported earlier were based on the properties of the spline MRA. Unless the interpolation scheme also exhibits some sort of polynomial character, these gains will be sacrificed. The obvious approach is to settle for a (quadratic) spline interpolation scheme. Ideally, one would like to interpolate the region between knot points (the irregularly spaced contour points) with a single quadratic patch, but continuity conditions might not allow this. Nonetheless, it should be possible to arrive at a simple interpolation (perhaps using Beta splines, because they are more general) which will give a near-optimal decomposition, given the lack of constraints (except at edge points).

Thus, the proposed scheme should provide the following (more or less):

- edge point detection (via wavelet transform)
- edge chaining (using wavelet transform info)
- grey-scale and geometry coding using Carlsson's approach

Figure 6.5: Edge point map. **The edge points extracted from the second level of the wavelet decomposition, using the method of [33].**

- simple (polynomial) interpolation scheme for an irregular set of knot-points (the contour pixels)
- compress the residual using the semi-orthogonal wavelet transform, as before.

The final form of the decomposition/synthesis algorithm would depend on hardware requirements.

6.6 Some preliminary tests

As a prelude to deeper investigations, a raster-based edge chaining algorithm, built around the public domain software of S. Mallat, was developed. The algorithm used the wavelet modulus/angle information to decide which points to chain. The edge chaining was an eight-connected process, however, in order to use Carlsson's contour block coding, (for contour geometry), the increment list produced was down-graded to a four- connected increment representation, which was then coded using a block size of 3. The starting points of the edges were coded in full. No compression was done on the modulus values — the intention was to check the effectiveness of the wavelet chaining and edge geometry representation.

The following observations were made:

Figure 6.6: Extracted edge maps. The edge map on the left has 860 edges, while that on the on on the right has 416. A edge length threshold was used for both — the left had all edges containing only one point removed. All edges with less than six edge points were removed from the second edge map.

1. The wavelet edge point detection works well — Figure 6.5.

2. While the chaining is generally acceptable, one must carefully choose the tolerance between neighbouring angle and modulus values along a contour. If the angle tolerance, in particular, is too fine, the contour will be fragmented. However, choosing the value too coarsely allows unwanted points to be linked into the edge.

3. The edges exhibit the thickness associated with a four-connected coding scheme.

4. Various thresholds on contour length and average modulus value along the contour were tried as a means of culling edges. (See Figure 6.6).

The scan-line edge-chaining scheme was based on linked lists (of evolving chain sections — See Figure 6.7) and was a little inefficient. There was some overhead involved in 'fixing' edge chains to make them suitable for contour block coding. One could certainly use an eight-connected contour code, although the efficiency of Carlsson's approach would then drop — nonetheless, it is computationally less demanding than, say, Generalized CRLC [22], so this might be acceptable. Alternatively, a simpler but less desirable contour coding method might be chosen.

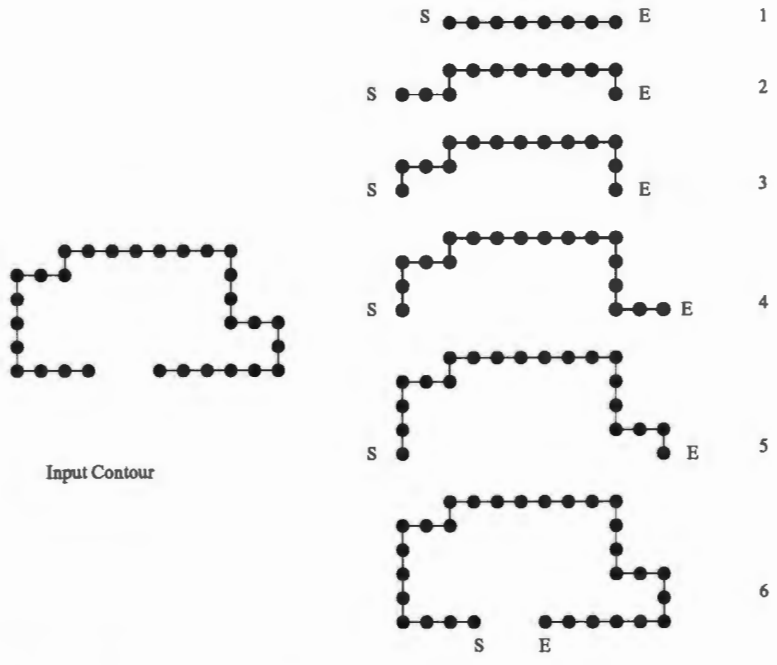


Figure 6.7: Scan-line edge chaining. The start and end of the evolving edge are represented by 'S' and 'E', respectively. Since the algorithm proceeds down the screen, only the directions above the current scan-line need be searched. When an edge point is encountered, a check is made to see whether it is already linked to an edge, if not it is bound to the appropriate edge. If an edge segment can be merged with an existing edge, this is done.

6.7 Concluding comments

This final chapter has touched on some ways in which the spline wavelet image code can be improved. Besides using a more effective (HVS) model to exploit the redundancy in the wavelet transform, one can employ edge interpolation techniques to reduce the amount of high-frequency information in the image. This information increases the number of wavelet coefficients and thus decreases the efficiency of wavelet compression. All that the edge interpolation need do is provide a smooth enough transition across edge regions so that subtracting the input and interpolated versions (to arrive at the lower energy residual) doe not introduce any discontinuities at the edge locations — this would defeat the whole purpose of the scheme. Provided the edge representation is compact, such a method should yield gains over straight wavelet compression.

Image synthesis involves the reconstruction of the residual (via an inverse wavelet transform) and the addition of the regenerated contour image. Given our desire to use the DE, it seems natural that we opt for an interpolation method which can use the chip's architecture — otherwise, we risk losing the advantages gained from spline MR synthesis. Bearing this in mind, the best alternative would be a spline interpolation method based on quadratics. The method should allow easy decomposition into DE instructions and be as general as possible — Beta splines have variable skew and tension parameters and would seem to be an ideal candidate for the job.

The above suggestions are only tentative. However, they offer some direction, and with sufficient refinement, they may form the basis for an enhanced compression/synthesis scheme.

Chapter 7

Conclusion

7.1 Overview of Main Results

The image coding system presented here consists of two parts:

- an image compression system, based on the quadratic spline-wavelet transform,
- and an image reconstruction or synthesis system, generated from the associated Multi-Resolution Analysis (MRA).

The choice of the quadratic MRA, rather than one based on cubics, was justified in Section 3.4: quadratics are cheaper to evaluate than cubics but still maintain a degree of smoothness, a property which is absent from lower order schemes. They are also less costly to produce on the Difference Engine and generate less noticeable quantization errors in the image domain (the wavelets are smaller). Most importantly, they are a viable alternative — the problems associated with this scheme were shown to be tractable.

It was clear that the distortion measure used in the quantization (minimum mean-squared error) was not appropriate (Section 4.5.2) — only images with few or no edges exhibited acceptable reconstruction fidelity for rates below 1 bpp. This was a consequence of using a metric which did not preserve edge coefficients and the fact that, in the semi-orthogonal framework, the L^2 norm is not preserved — see Section 7.1.1.

Image synthesis starts by performing an inverse DWT. After this is accomplished, the algorithms developed in Chapter 5 are invoked to render the decompressed image. These

algorithms enable one to decrease the rendering time for most images when compared to directly setting each pixel — the gains can be very substantial for smooth images (a factor of 4+). The rendering algorithms are designed to exploit the architecture of the Difference Engine, which can render splines very rapidly, and the continuity imposed by the choice of spline. In particular, (lower level) multi-resolution approximation images can be rendered very cheaply, since they consist of quadratic spline patches and contain no high-resolution detail.

There are a number of areas for future research. The quantization algorithm must certainly be improved — at least to deal with edges. Further improvements should be possible using second generation coding techniques, particularly those based on edge-coding. Finally, the architecture of the Difference Engine can be modified to improve performance or entirely an new architecture may be developed specifically to render quadratic spline images.

7.1.1 The Spline Wavelet Transform

A quadratic spline wavelet transform was employed to de-correlate the image — a variant which has been neglected because of certain implementational complications. This thesis has, however, demonstrated that such a wavelet is preferable to the (more widely used) cubic spline wavelet from a computational point of view, and that the 'problems' referred to are easily dealt with (Section 3.4). It is also more suited to multi-resolution image synthesis than a linear spline wavelet, since quantization errors in the latter would be more visible (as Mach banding). Spline wavelets *per se* were chosen since they are eminently suited to the architecture of the Difference Engine (DE). In addition, the formulation of the approximation and detail spaces in this system allows for easy generation of DE instructions, something that other splines schemes (bi-orthogonal, for example) do not readily permit. The quadratic wavelet has short reconstruction filters, which implies that one can decompress the coded image rapidly — an important aspect for the display of compressed video material. FFT processors can be used to implement the filtering operations in hardware.

Vector Quantization

Minimum Mean-Squared Error (MMSE) vector quantization (Section 4.4.1) was used to quantize the transform coefficients i.e., the reproduction vector which minimizes the mean

squared error when compared with the input is selected. The results obtained (Section 4.5.2) showed that a MMSE distortion metric was *not* sufficient for the proposed semi-orthogonal wavelet scheme. Poor reproduction of edges was evident, as well as transform domain blocking effects. Besides the dubious visual relevance of such a measure, the lack of complete orthogonality within the semi-orthogonal system poses some theoretical problems: non-orthogonality implies that one no longer has equivalent norms for the transform and image domains. Thus, minimizing the transform quantization error does not ensure that one simultaneously minimizes the reconstruction error in the image domain (under this MSE metric).

The effects of quantization errors were amplified at low resolutions because the quadratic spline wavelet has a fairly large support, which in turns implies that the dilated wavelets on the lowest levels cover a significant area — if we had used a cubic wavelet the problem would have been worse, since the cubic wavelet has a larger support. However, simple thresholding tests (on the detail coefficients) indicate that the vector quantization did not exploit the sparseness of the representation very well — image fidelity was seldom acceptable below bit-rates of 1 bpp, unless there was a large measure of texture masking, while the number of wavelet coefficients which could be discarded (for a comparable level of reconstruction fidelity) resulted in a far lower effective bit-rate.

While the results obtained can be partially blamed on the lack of dynamic bit-allocation (pre-allocated quality settings were hard-coded and remained constant for all images), the uniformly poor showing for all non-trivial test images seems to rule this out as a major contributing factor. Examination of the transform domain representation showed obvious blocking effects around edge coefficients — further evidence of the inappropriateness of the MMSE measure. The number of reproduction levels in the VQ code-books was generally limited to 256, which does not enable much scope for good vector matching. Nonetheless, this number is of the same order as that used in other VQ sub-band schemes [1, 46]. However, even larger code books only provided marginal gains (Section 4.5.2).

Entropy Coding

Optional Huffman coding was used to exploit the redundancy in the VQ indices, a carryover from the non-uniform distribution of wavelet coefficients. However, static Huffman coding imposes the overhead of storing the Huffman code strings (or probability information),

which can overwhelm the gains achieved by applying this entropy coding.

All the images tested in this work derived some benefit from a perfect entropy coding; however, in many cases the additional data required by the Huffman table actually *increased* the size of the compressed information. The theoretical compression gain resulting from Huffman coding was generally in the range of 10-30%. Very large images experienced a better practical gain than small images, since the table size is independent of the image size. It should be noted that the compression results cited in Chapter 4 exclude entropy coding.

7.1.2 Multi-Resolution Synthesis

The synthesis scheme exploits the Difference Engine's architecture and the properties of the spline multi-resolution analysis, to reduce the number of instruction cycles needed to produce an image. This is a two phase process, consisting of:

- adaptive reconstruction using the MRA of the image and
- the merging of similar adjacent polynomials.

In this context, 'Adaptive' means that only important detail from the MRA is used to rebuild the image. Polynomial merging only takes place in the approximation image, since it is here where such 'redundancy' is likely to exist. Experimentation revealed that most images can be rendered more efficiently using this approach (Section 5.4).

A series of simple computations are performed to determine whether the adaptive synthesis/merging operations reduce the time to render the image below that required when each pixel must be set individually. If this is not the case, instructions are generated to explicitly set each pixel, otherwise multi-resolution synthesis is used to rebuild the scan-line. It is important to note that no expensive function evaluations occur during this testing — only if multi-resolution synthesis is chosen, will the required detail and approximation image functions be evaluated. The calculations which are used to evaluate these image functions can be recast as separable 2-D convolutions and implemented in hardware using FFT processors.

To further improve rendering performance, the continuity constraint (C^1) is used to lower the number of separate interpolation instructions which one requires to generate a scan-

line: to interpolate a sequence of adjacent spans, one sets the second difference for each and issues *one* interpolation instruction.

Noteworthy rendering gains were obtained for images with a high degree of smoothness. For such images, the reduction in rendering time can exceed a factor of 4. Natural, textured images typically experience gains of between 10-60% (this is a consequence of naturally occurring pixel correlations). For many images, the application of the adaptive reconstruction alone (i.e., with no subsequent merging) was insufficient to ensure a gain over the baseline count. Nonetheless, there was a dramatic decrease in cost compared to that required for full MR synthesis, since only small patches of the detail images were accumulated. The inclusion of the merging process in the approximation image was thus necessary for improved performance for a more general class of images.

7.1.3 Second Generation Coding: edge extraction and coding

A wavelet-based edge-point extraction and chaining system was developed as a precursor to a fully fledged second generation edge coding scheme. The edge extraction followed the implementation in [17]. The edge-point chaining algorithm, developed during this work, was based on the information extracted from the wavelet transform. In general, the edge chains are satisfactorily extracted (Section 6.3.3). However, for such a chain code to be of use, only longer edge chains must be maintained; there is a fair degree of difficulty involved in making an optimal decision. If one keeps too many edges, compression performance is sacrificed. On the other hand, if too many edges are thrown out, the reproduction becomes highly distorted. There is still much debate about the usefulness of current HVS models — this is an ongoing area of research which is unlikely to yield a satisfactory model until the processes which underlie vision are more clearly understood. Fortunately, in the context of the edge coding proposed in [17], one need only ensure that the dominant edges are extracted — the error image (the difference between the input and the approximation generated from the edge representation) will then be more amenable to wavelet transform coding. The results obtained indicate that this minimal requirement is not difficult to satisfy.

7.2 Future Work

7.2.1 Edge Extraction and Coding

The last chapter of this dissertation makes a case for edge coding as a means to enhance compression performance. The idea is to remove the edges from an image and to code them in some compact manner [7]. One then generates an image from this edge map, which contains much of the information needed by the HVS. The residual image will generally contain texture information, which can be compressed using a scheme such as wavelet compression, which can represent such information well. The HVS's insensitivity to high-frequency error implies that one can attain fairly low bit-rates for this data. Thus, provided the edge image can be coded compactly, one should achieve a higher compression ratio. Initial experimentation with this kind of coding has been promising [7, 17].

7.2.2 Alternate Quantization Schemes

It is evident from the results obtained earlier that Vector Quantization with a MMSE distortion measure did not function well for the given wavelet. Possible reasons for this poor showing have already been discussed above. There are several options which could be considered to improve quantizer performance

- use a perceptually relevant distortion measure during VQ training and quantization
- use an adaptive VQ quantization scheme which uses more bits to qualify edge/texture information
- abandon VQ and use some sort of run-length encoding.

The question of relevance of a particular metric to quantization is generally considered to be an open one. There is, however, fairly wide consensus that the MSE metric is not a good measure of visual fidelity. It seems that the use of some sort of adaptive (predictive) VQ would be the most fruitful course to choose. Run-length coding has the advantage of simplicity, and may be useful if applied to the detail coefficients after appropriate thresholding. Of course, the means of determining an appropriate threshold is problematic, requiring some sort of HVS model.

7.2.3 Alternate Spline Schemes

It might prove beneficial to examine other spline MRA's in detail, to determine whether they offer any advantages over the present scheme. For example, the bi-orthogonal spline wavelets seem to be amenable to MSE quantization [1]. Another possibility, and one which was also previously discarded, was the idea of using a spline of lower order than a quadratic. It may be that the decrease in the support of the wavelet can be traded off against the loss of smoothness and the less appealing nature of quantization errors. Such a choice might well be implementation specific or be forced upon one by hardware limitations.

7.2.4 Alternative Architectures for Spline Image Synthesis

The ideas presented below were introduced and discussed in Section 5.4.1. The reader is referred there for a more thorough treatment.

A 2-D Interpolation Engine

The Difference Engine is an inherently one dimensional device. Consequently, it cannot exploit the vertical correlations which exist within the 2-D spline MRA. A way around this problem would be to develop a 2-D Difference Engine, i.e., one which performed interpolation in two dimensions — across scan-lines as well as along them. Among the problems one faced would be

- the rapid growth of (2-D) interpolation error
- the overhead imposed by the additional information required to interpolate a rectangle
- the increased complexity of the logic required to handle the 2-D interpolation and the consequent decrease in speed.

While these problems can probably be overcome, the gains that such a system could provide would only be worthwhile at low resolutions, and as such the benefits are not immediately apparent.

Direct Basis Accumulation

Since there are only a small number of basis elements for each resolution and orientation, one could build a dictionary of these waveforms and use them to reconstruct the image: the image (represented in terms of its multi-resolution analysis) is just the weighted sum of a series of appropriately scaled and shifted basis functions. To reconstruct a scan-line (assuming a scan-line wide accumulator), the orientation and resolution of the 2-D bases which intersect that scan-line are determined and this information is used to identify the basis cross-sections which cut the scan-line (it is these cross-sections which reside in the dictionary). The cross-sections are accumulated into the buffer after the components in each waveform have been multiplied by the weighting factor and shifted into position. Since each of these operations (multiplication, shifting and accumulation) can be done in parallel, the scan-line could be rapidly reconstructed. A further bonus of this approach would be the ability to use *arbitrary* basis elements in the dictionary — a function could be provided to load a new set of waveforms. Thus, different image representations would be supported. The logic to determine the basis intersections etc. would have to be done externally, since this is highly dependent on the kind of representation. Nonetheless, the speed of the parallel operations and the configurability of the device would make it an attractive alternative to the Difference Engine.

Bibliography

[1] M. Antonini, M. Barlaud, M. Mathieu, and I. Daubechies. Image coding using wavelet transforms. *IEEE trans. on image processing*, 1(2):205–220, April 1992.

[2] M. Barnesly. *Fractals everywhere.* Academic Press, 1988.

[3] E. H. Blake and A. A. M. Kuijk. A difference engine for images with applications to wavelet decomposition. Proceedings of the Second International Conference on Image Communications (IMAGE'COM), pages 309–314, 1993.

[4] C. Brislawn. Classification of symmetric wavelet transforms. Technical report, Los Alamos National Laboratories, March 1993.

[5] P. Burt and E. Adelson. The laplacian pyramid as a compact image code. *IEEE trans. on comm.*, 31(4):532–540, 1983.

[6] J. Canny. A computational approach to edge detection. *IEEE trans on patt anal and mach intell*, 8:679–698, 1986.

[7] S. Carlsson. Sketch based coding of grey level images. *Signal Processing*, 15:57–83, 1988.

[8] P. Chou, T. Lookabaugh, and R. M. Gray. Entropy constrained vector quantization. *IEEE trans. ASSP*, 37(1):31–42, January 1989.

[9] C. Chui and H. Diamond. A natural formulation of quasi-interpolation by multi-variate splines. *Proceedings of the American Mathematical Society*, 99(4), 1987.

[10] C. K. Chui. *An Introduction to Wavelets: Wavelet Analysis and its Applications*, volume one. Academic Press, Boston, 1992.

[11] C. K. Chui and J. Z. Wang. Computational and algorithmic aspects of cardinal-spline wavelets. Technical Report 235, TAMU, Centre for Approximation Theory (CAT), December 1990. Note: There are some transcription errors in the (quadratic) decomposition sequences given here.

[12] M. Cody. The fast wavelet transform. *Dr. Dobb's Journal*, 17(4):16–28, April 1992.

[13] I. Daubechies. Ten lectures on wavelets. volume 61 of *CBMS-NSF series in applied mathematics*. SIAM, 1992.

[14] P. Desarte, B. Macq, and D. Slock. Signal-adapted multiresolution transform for image coding. *IEEE Trans. on Info. Theory*, 38(2):719–746, March 1992.

[15] R. K. DeVore, B. Jawerth, and B. Lucier. Wavelet image compression through wavelet transform coding. *IEEE Trans. Information Theory*, 38(2):719–746, March 1992.

[16] Y. Fisher. Fractal image compression. Siggraph course notes, 1992.

[17] J. Froment and S. Mallat. Second generation compact image coding with wavelets. In C.K. Chui, editor, *Wavelets: A Tutorial in Theory and Applications*, volume two, pages 655–678. Academic Press, Boston, 1992.

[18] R. M. Gray. Vector quantization. *ASSP Magazine*, pages 4–28, April 1984.

[19] H.J. Heijman. Discrete wavelets and multiresolution analysis. In Koornwinder [21], pages 49–79. This article originally appeared in CWI Quarterly, Vol. 5, No. 1, March 1992.

[20] T. Huang. Coding of two-tone images. *IEEE trans. on comm.*, COM-25(11):1406–1424, November 1977.

[21] T. H. Koornwinder, editor. *Wavelets: An Elementary Treatment of Theory and Applications*, volume 1 of *Approximations and Decompositions*. World Scientific, 1993.

[22] M. Kundu, B. Chaudhuri, and D. Majumder. A generalized digital contour coding scheme. *Computer vision, graphics and image processing*, 30:269–278, 1985.

[23] M. Kunt, A. Ikonomopoulos, and M. Kocher. Second generation image coding techniques. *Proc IEEE*, 73(4):549–574, 1985.

[24] Kunt, M., et al. Recent results in high-compression image coding. *IEEE trans on circuits and systems*, pages 1306–1336, 1987.

[25] A. Lewis and G. Knowles. Image compression using the 2-d wavelet transform. *IEEE trans on image processing*, 1(2):244–250, April 1992.

[26] J. Lim. *Two dimensional signal and image processing.* Prentice-Hall International, 1990.

[27] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE trans. on comm.*, com-28(1):84–95, January 1980.

[28] S. Mallat. Multi-frequency channel decompositions of images and wavelet models. *IEEE Trans. on Acous. Speech and Signal processing.*, 37(12):2091–2110, 1989.

[29] S. Mallat. A theory of multiresolution signal decomposition: The wavelet representation. *IEEE Trans. on Patt. Ana. and Mach Intell.*, 11:674–693, 1989.

[30] S. Mallat. Zero crossings of a wavelet transform. *IEEE trans. on Inf. Theo.*, 37(4):1014–1033, July 1991.

[31] S. Mallat. Zero-crossings of a wavelet transform. *IEEE trans on inf theory*, 37(4):1019–1033, July 1991.

[32] S. Mallat and S. Zhong. Singularity detection and processing with wavelets. *IEEE Trans. Inf. Theo*, 38(2):617–643, March 1991.

[33] S. Mallat and S. Zhong. Characterization of signals from multiscale edges. *IEEE Trans. on Patt. Ana. and Mach. Int.*, 14(7):710–732, July 1992.

[34] D. Marr. *Vision.* W.H. Freeman, 1982.

[35] D. Marr and E. Hildreth. Theory of edge detection. *Proc of royal soc of London*, 207:187–217, 1980.

[36] Y. Meyer. Un contre-exemple a la conjecture de Marr et a celle de S. Mallat. University of Paris preprint, 1991.

[37] Y. Meyer. *Wavelets: algorithms and applications.* SIAM, 1993.

[38] P. Moulin. A multi-scale relaxation algorithm for snr maximization in 2-d non-orthogonal sub-band coding. Technical report, Bell communications research, March 1994.

[39] P. Nacken. Image compression using wavelets. In Koornwinder [21], pages 81–91. This article originally appeared in CWI Quarterly, Vol. 5, No. 1, March 1992.

[40] T. Peli and D. Malah. A study of edge detection algorithms. *Computer graphics and image processing*, 20:1–20, 1982.

[41] D. Rioul and P. Duhamel. Fast algorithms for discrete and continuous wavelet transforms. *IEEE Trans. on Info. Theory*, 38(2):569–585, March 1992.

[42] A. Rosenfeld. Digital straight line segments. *IEEE trans. on computers*, C-23(12):1264–1269, December 1974.

[43] M. Ruskai, editor. *Wavelets and their applications.* Jones and Bartlett, Boston, 1992.

[44] R. J. Safranek and J. D. Johnston. A perceptually tuned subband image coder with image dependent quantization and post-quantization data compression. volume 3, pages 1945–1948. IEEE ASSP, 1989.

[45] A. Segall. Bit-allocation and encoding for vector sources. *IEEE trans on inf theory*, IT-22:162–169, March 1976.

[46] T. Senoo and B. Girod. Vector quantization for for entropy coding of image subbands. *IEEE trans of image processing*, 1(4):526–533, October 1992.

[47] M. Unser, A. Aldroubi, and M. Eden. Fast b-spline transforms for continuous image representation and interpolation. *IEEE Trans. Pattern Anal.and Mach. Intell.*, 13(3):277–285, 1991.

[48] M. Unser, A. Aldroubi, and M. Eden. A family of polynomial spline wavelet transforms. *Signal Processing*, 30:141–162, 1993.

[49] G. Wallace. The jpeg still image compression standard. *Communications of the ACM*, 34(4):31–44, April 1991.

[50] A. Watson. Efficiency of a model human image code. *J. Opt. Soc. Amer.*, 4:2401–2417, December 1987.

[51] P. Westerink, D. Boekee, and J. Biemond. Sub-band coding of images using vector quantization. *IEEE trans on comm.*, 36(6):713–719, June 1988.

[52] J. Woods and S. O'Neil. Subband coding of images. *IEEE trans. ASSP-34*, 34, 1986.

[53] L. Wu. On the chain code of a line. *IEEE trans. on patt. anal. and mach. intell.*, PAMI-4(3):347–353, May 1982.

[54] R. K. Young. *Wavelet theory and its applications.* Kluwer Academic Publishers, 1993.