Centrum voor Wiskunde en Informatica
REPORT*RAPPORT*

Quadratic vs Cubic Spline-Wavelets for Image Representation and Compression

P.C. Marais, E.H. Blake, A.A.M. Kuijk

Probability, Networks and Algorithms (PNA)

PNA-R9717 November 30, 1997

Report PNA-R9717
ISSN 1386-3711

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Quadratic vs Cubic Spline-Wavelets for Image Representation and Compression

P.C. Marais
Robotic Research Group
University of Oxford, UK

E.H. Blake
Dept. of Computer Science, University of Cape Town
Rondebosch 7700, South Africa

A.A.M. Kuijk
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

ABSTRACT

The Wavelet Transform generates a sparse multi-scale signal representation which may be readily compressed. To implement such a scheme in hardware, one must have a computationally cheap method of computing the necessary transform data. The use of semi-orthogonal quadratic spline wavelets allows one to maintain a suitable level of smoothness in the MRA whilst enabling cheap computation. Among the other advantages afforded by such a scheme are easily implementable boundary conditions and the existence of either linear or generalized linear phase in the wavelet filters, which has important consequences for signal compression. For image representation, a compressed spline MRA i.e., an MRA in which 'redundant' basis coefficients have been ignored, allows us to generate a spline representation of an image which (for low order splines) may be readily displayed on specialised graphics hardware. Such a representation may also be used directly to generate a progressively transmitted image.

1991 Mathematics Subject Classification: 42C15, 68U10.

1991 Computing Reviews Classification System: E.4, I.4.2, I.4.5, I.4.10.

Keywords and Phrases: wavelets, semi-orthogonal, multiresolution analysis, quadratic, cubic, spline, image representation, compression.

Note: Work carried out under project PNA4.5 "Image Generation for Virtual Environments"

1. Introduction

Wavelet Theory has had a significant impact on many areas of engineering and mathematics. The sparse and local representation generated by a wavelet decomposition is fundamental to its rising popularity as a data (in particular, image) compression technique. Directly coupled to the wavelet representation is the concept of a Multi-Resolution Analysis (MRA), in terms of which a signal may be decomposed into a weighted sum of scaled and translated wavelets.

Many of the wavelets which are employed today are defined by means of recursion formulae and do not have a simple closed form expression. The *Spline wavelets* introduced by C. K. Chui [4] and M. Unser [8] are well suited to analytic manipulations, possessing many desirable properties in addition to having a closed form representation. These properties

include rapid computation of the wavelet representation and preservation of certain kinds of phase information, which is important for image compression (to ensure minimal distortion of the reconstructed signal). Furthermore, since the wavelets and scaling functions are either symmetric or anti-symmetric, the boundary conditions are substantially simplified (cf. Section 2.1).

Cubic splines are generally used since they are believed to provide the best compromise between computational cost and smoothness. Low order[1] splines (constant and linear) are unsatisfactory: they contain gradient discontinuities which are reflected in the Multi-Resolution Analyses which they generate. These discontinuities are extremely noticeable (Mach banding) when wavelet compression techniques are used which discard 'redundant' basis coefficients. Consequently, we desire some degree of smoothness in our scheme.

We present theory and substantive data to support our contention that a quadratic based wavelet formalism is superior to a cubic based one, in terms of both implementation cost and representational accuracy.

2. Spline Wavelets and Multi-Resolution Analysis

A wavelet, $\psi(x)$, is a 'well behaved' function (an element of the functional space $L^2(\mathbb{R})$) which is localised in both space and frequency (the Fourier domain) and possesses a zero integral i.e. it oscillates.

It may be shown that any function $f(x) \in L^2(\mathbb{R})$ may be expressed as a weighted sum of scaled and translated wavelets:

$$f(x) = \sum_{j,k} c_{j,k}\psi(2^j x - k) \quad k, j \in \mathbb{Z}.$$

The parameter j controls the scaling of the wavelet, whilst k determines its position. Generally, wavelets are orthogonal (in that their L^2 inner product is zero) across scales and within a particular scale. However, certain benefits accrue if this restriction is relaxed somewhat. In the case of *spline wavelets*, only orthogonality across scales is enforced, and these wavelets are thus called *semi-orthogonal* wavelets.

The wavelet decomposition of a signal implies an infinite expansion over all possible scales. That is, every $f(x) \in L^2(\mathbb{R})$ has a *direct sum* decomposition:

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

with $g_j(x) = \sum_k d_{j,k}\psi(2^j x - k) \in W_j$, for all $j \in \mathbb{Z}$. The *detail spaces* W_j are spanned by the translates of wavelets of a given scale j.

Directly coupled to this expansion is the concept of a *Multi-Resolution Analysis (MRA)* [4], which provides a means of progressively examining a signal at different scales. In the MRA framework, a signal, $f(x)$, may be approximated at a scale j, by evaluating

$$f_j(x) = \sum_k c_{j,k}\phi(2^j x - k), \tag{2}$$

where the *scaling function*, $\phi(x)$, generates a scaled basis

$$\mathrm{span}\{\phi(2^{\mathrm{j}}\mathrm{x} - \mathrm{k}),\ \mathrm{k} \in \mathbb{Z}\},$$

[1] The order of a polynomial is the degree plus one.

for the *approximation spaces*, V_j. Amongst the numerous properties possessed by an MRA, the following is of the most immediate relevance to us

$$f_{j+1}(x) = f_j(x) + g_j(x),\ f_j \in V_j, g_j \in W_j. \tag{3}$$

This states that any (L^2) signal may be decomposed into the sum of a lower resolution approximation and a difference signal which encodes the detail lost in going from the scale $j+1$ to the scale j.

In practice, our signal is approximated at some (input) resolution, lets call it scale 0, and only decomposed across a small number of scales, say M. This results in a residue, $f_{-M}(x)$, which encodes all the information over the scales which have not been expanded.

$$f_0(x) = g_{-1}(x) + g_{-2}(x) + \cdots + g_{-M}(x) + f_{-M}(x). \tag{4}$$

The basis coefficients $\{d_{j,k}, c_{M,k};\ j = -1, \ldots, -M,\ k \in \mathbb{Z},\}$ in this representation constitute the Wavelet Transform of the signal. The Wavelet Transform is a 'sparse' representation, producing many coefficients which may be discarded without noticeably affecting the quality of the reproduced signal. This enables one to achieve signal compression. These coefficients may be obtained through a series of convolutions followed by signal down-sampling, arranged in a cascading fashion to a depth determined by the number of steps in the decomposition.

$$c_{j-1,k} = \sum_l a_{l-2k} c_{j,l}; \tag{5}$$

$$d_{j-1,k} = \sum_l b_{l-2k} c_{j,l}. \tag{6}$$

In a similar fashion, up-sampling followed by convolution, again arranged in a feedback cascade may be used to regenerate the input signal from the wavelet coefficients.

$$c_{j,k} = \sum_l [p_{k-2l} c_{j-1,l} + q_{k-2l} d_{j-1,l}]. \tag{7}$$

The *decomposition sequences*, $\{a_k\}$ and $\{b_k\}$, correspond to discrete versions of the dual scaling function and wavelet, respectively. They are Infinite Impulse Response (IIR) filters. Similarly, the *reconstruction sequences*, $\{p_k\}$ and $\{q_k\}$, correspond to discrete versions of the compactly supported scaling function and wavelet, respectively.

For 2D signals (like images), a *separable* MRA is generated by taking the tensor product of the 1D MRA.

We define *three* 2-D wavelets as follows,

$$\Psi^{[1]}(x,y) = \phi(x)\psi(y) \tag{8}$$

$$\Psi^{[2]}(x,y) = \psi(x)\phi(y) \tag{9}$$

$$\Psi^{[3]}(x,y) = \psi(x)\psi(y). \tag{10}$$

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

$$I_k(x,y) = \sum_{i,j} c_{k;i,j} \Phi(2^k x - i, 2^k y - j). \tag{11}$$

where $\Phi(x,y) \equiv \phi(x)\phi(y)$, is the 2D scaling function.

Similarly, just as we did in 1-D, we may write

$$I_{k+1}(x,y) = I_k(x,y) + g_k(x,y). \tag{12}$$

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

$$g_k(x,y) = \sum_{i,j,p=1}^{3} {d_k}_{ij}^{[p]} \Psi^{[p]}(2^k x - i, 2^k y - j). \tag{13}$$

The 2D decomposition relation becomes

$$c_{j-1;k,l} = \sum_m \sum_n a_{m-2k} a_{n-2l} c_{j;m,n} \tag{14}$$

$${d^{[1]}}_{j-1;k,l} = \sum_m \sum_n a_{m-2k} b_{n-2l} c_{j;m,n} \tag{15}$$

$${d^{[2]}}_{j-1;k,l} = \sum_m \sum_n b_{m-2k} a_{n-2l} c_{j;m,n} \tag{16}$$

$${d^{[3]}}_{j-1;k,l} = \sum_m \sum_n b_{m-2k} b_{n-2l} c_{j;m,n}. \tag{17}$$

while the reconstruction relation is now

$$\begin{aligned} c_{j;k,m} = \quad & \textstyle\sum_l \sum_p p_{k-2l} p_{m-2p} c_{j-1;l,p} + \\ & \textstyle\sum_l \sum_p p_{k-2l} q_{m-2p} {d^{[1]}}_{j-1;l,p} + \\ & \textstyle\sum_l \sum_p q_{k-2l} p_{m-2p} {d^{[2]}}_{j-1;l,p} + \\ & \textstyle\sum_l \sum_p q_{k-2l} q_{m-2p} {d^{[3]}}_{j-1;l,p}. \end{aligned} \tag{18}$$

The reader is referred to [4] for additional explanations and references. The values of the quadratic decomposition and reconstruction sequences may be found in [5, 6].

2.1 Boundary Conditions

In order to correctly compute the Wavelet Transform of a finite signal, such as an image, the boundaries of the signal must be extended to enable correct evaluation of the boundary convolutions. Arbitrary signal extension will induce boundary artifacts which will contaminate the wavelet representation and, consequently, the associated MRA. This issue is of crucial importance for both the quadratic and cubic schemes, since the decompostion filters, which are used to generate the representation, are (truncated) IIR filters with large support: incorrect boundary extensions can have wide ranging affects on the decomposition.

Since the filters used are either symmetric or anti-symmetric it is possible to use symmetric boundary extensions (cf. Appendix A.1 for details) in such a way that signal symmetries persist (in some form or another) even after down-sampling.

2.1.1 Convolution of symmetric signals and filters. A signal may have several points of symmetry, all with different characteristics. We need 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 in Appendix A.1 (whole-sample and half-sample (a)symmetry).

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

$$\begin{aligned} a_k &= (s * f)_k \\ &= \sum_n \pm s_{k-n} f_n \\ &= \sum_n \pm s_{l-k+n} f_{t-n} \\ &= \sum_{n'} \pm s_{l+t-k-n'} f_{n'} \\ &= \pm a_{l+t-k}. \end{aligned} \tag{19}$$

Thus, the centre of symmetry will be at $\frac{l+t}{2}$ 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 we 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 $\{\overline{b}_k^m\}$ were 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:

$$\begin{aligned} p_k^m &= p_{m-k}^m \\ q_k^m &= q_{3m-2-k}^m \\ \overline{a}_k^m &= \overline{a}_{-m-k}^m \\ \overline{b}_k^m &= (-1)^m \overline{b}_{-(3m-2)-k}^m \end{aligned} \tag{20}$$

2.1.2 Computing with the boundary extensions. To determine the boundary conditions (BCs), we need only consider a one level decomposition — the procedure discussed below can be used for all subsequent levels. Our 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 the sequence start on index 0. The input boundary extensions we chose are given in Table 1 and Table 2 and are presented in terms of the start and end-points of the input sequence and the type of symmetric

	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 1: 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).

extension required. The output values stated represent the new points of symmetry etc. after filtering and decimation with the indicated filters. By using the symmetry extensions indicated in the Tables for each subsequent step of the decomposition, we can ensure that the output sequence retains some form of boundary symmetry, even after convolution (with the appropriate symmetric filter) and down-sampling. One simply notes the filter to be applied and the values of the first and last sequence indices of the input signal and uses the Tables to apply the indicated extension.

The boundary extensions for reconstruction are simply those resulting from the previous levels filtering and decimation. They are determined by examining Table 1 or Table 2 after processing each levels input sequences. These BCs must either be stored on re-created; for compression purposes, the latter option is more suitable.

To re-create the BCs, one just simulates the filter bank operation, without actually performing the filtering operations — this requires very few calculations (one just needs to compute the starting end terminating sequence indices). This approach is in contrast markedly to the methods 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 is as follows:

	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 2: 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 $\{\overline{b}_k^4\}$.

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

2.2 Image Representation on a Spline Basis

Given the computational simplicity of polynomial spline representations it may be desirable to express an image as the sum of polynomial primitives. This may be achieved by performing a spline MRA on a given image.

The first step towards implementing this analysis is to project the input image onto the 'best' approximating spline. This may, however, be an expensive task, often requiring the inversion of a matrix system with the same dimensions as the image. A cheap and satisfactory alternative is *quasi-interpolation* [4, 6], which uses local information to determine the interpolant. As the degree of non-locality is increased, the accuracy of the fit improves — in the limit the signal (image) is interpolated exactly (Section 3.3).

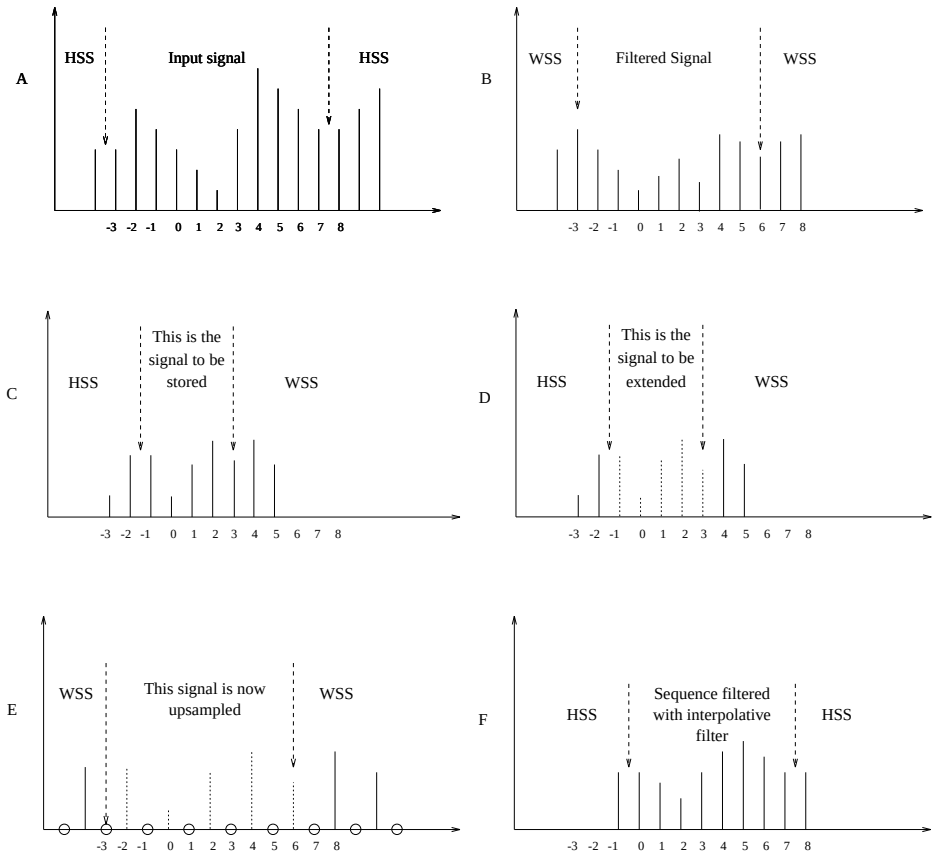

Figure 1: Partial decomposition and reconstruction. This example illustrates the ideas referred to above, by working through the filtering steps for the (quadratic) decomposition filter $\{a_k\}$ and the reconstruction filter $\{p_k\}$. 'A' shows the input signal and its extension (HSS, as required by Table 1). 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.

3. The Case For A Quadratic MRA

Having established the theoretical base for our discussion, we are now in a position to compare the relative merits of the quadratic and cubic based MRA's. There are three issues which bear discussion.

3.1 Computational cost

Quadratic wavelets and scaling functions possess a smaller support than their cubic counterparts. This means that the synthesis of the input signal from its wavelet decomposition (as is the case after compression) may be achieved more speedily in the former case.

The number of non-zero components in the relevant filters is indicated in Table 3. It can be seen from this table and Equation 7, that the number of multiplications required to compute the (up-sampled) convolutions is proportional to $(\#\{p_k\} + \#\{q_k\})$. Thus, assuming that 2D signals are computed using the separability property, the cubic schemes requires

Order	$\#\{p_k\}$	$\#\{q_k\}$
Quadratic	4	8
Cubic	5	11

Table 3: Comparison of reconstruction filter lengths for the cubic and quadratic schemes

approximately 33% more multiplications and additions than the quadratic scheme (for both 1D and 2D signals). Of course, if the separability of the 2D convolutions is not exploited, the balance swings even further in favour of quadratics (with cubics then requiring almost twice as many multiplications and additions).

If one wishes to compute the values of the interpolated signal at any real point then, once again, the quadratic computation is cheaper: the recursive formula

$$N_m(x) = \frac{x}{m-1} N_{m-1}(x) + \frac{m-x}{m-1} N_{m-1}(x-1). \tag{21}$$

where m is the order of the polynomial, and

$$N_1(x) = \chi_{[0,1)}(x) = \begin{cases} 1 & \text{if } x \in [0,1); \\ 0 & \text{otherwise.} \end{cases} \tag{22}$$

may be used to efficiently compute the value of the spline basis at any point with a complexity proportional to the order of the spline.

3.2 Wavelet Representation Error

Since the dual wavelet and scaling function have infinite support, the decomposition sequences they engender (which are used to determine the wavelet representation coefficients) must be truncated. However, this truncation cannot be arbitrary — it must preserve the symmetry of the decomposition sequences or else one loses the benefit of symmetric signal extension and reconstruction artifacts will occur.

The decomposition filters must, not unexpectedly, obey certain pass-band conditions if they are to filter out the right information [4]:

$$\begin{aligned} \sum_k a_k &= 1; \\ \sum_k b_k &= 0. \end{aligned} \tag{23}$$

These formulae correspond to the requirement that the scaling function and the the wavelet have unit and zero integral, respectively.

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, $\{b_k^4\}$, 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.

	$\#\{a_k\}$	$\#\{b_k\}$	$\mathrm{O}(\sum a_k - 1)$	$\mathrm{O}(\sum b_k)$
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 4: 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, Equation (23).

Table 4 illustrates the point very well. Examination of the data shows that a quadratic filter with only 14 non-zero entries satisfies the pass-band conditions to the same order of magnitude as a cubic filter with 27 non-zero components (in the b_k sequence). Consequently, a quadratic decomposition may be computed more cheaply than a cubic decomposition, assuming that a similar level of accuracy is required.

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 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 2 show the effect of this failure: the incorrectly registered frequencies add ripples to the signal.

This lack of robustness on the part of the cubic spline wavelet scheme has evinced surprisingly little comment in the literature.

3.3 Quadratic Image Representation

The last few years have seen the emergence of increasingly sophisticated display hardware. An example of such is the Difference Engine [1] which has the ability to generate a polynomially interpolated span of (arbitrary length) pixel data with a single cheap instruction. This paves the way for novel image synthesis techniques, such as the reconstruction of an image expressed on a 'compressed' polynomial spline basis obtained by applying a data compression technique to the Wavelet Transform corresponding to a spline-based MRA [6]. However, before such a representation can be generated the initial signal (image) must be approximated by a spline of the appropriate degree.

To reduce costs, one may employ a *quasi-interpolation* (cf. Appendix) scheme [6], which uses local image data to compute the approximation. Again the choice of polynomial must

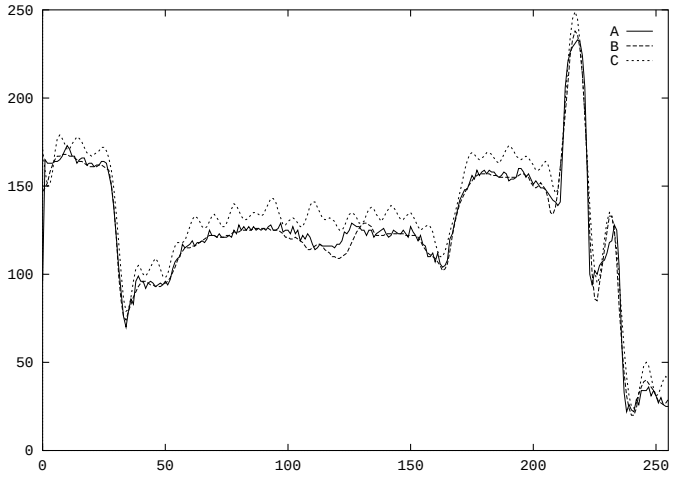

Figure 2: Failure of the cubic filters at low truncations. 'A' gives the input data, 'B' the 2nd resolution level cubic decomposition approximating the image and 'C' the reconstruction to level 2 after decomposing with the over truncated cubic decomposition sequences. When the cubic filters are not over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

be decided — do we wish to use a cubic or a quadratic based MRA? Surprisingly, the cubic quasi-interpolation turns out to be *less* accurate than quadratic quasi-interpolation! This provides yet another incentive to use a quadratic scheme.

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 5: 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 $k = 1$.

Table 5 shows that for a second order ($k = 2$) quasi-interpolation scheme (which employs a 5x5 convolution mask), the mean interpolation error is almost zero, with a standard deviation of under a pixel value (in the quadratic case) and a maximum error of 10 pixels. The maximum error occurs where we have a very sharp intensity gradient (like a spike — see Figure 3). In contrast, the same order of cubic quasi-interpolation yields a significantly more noisy fit.

This result is supported by the theory of quasi-interpolation [4, 6]: we have the following relationship

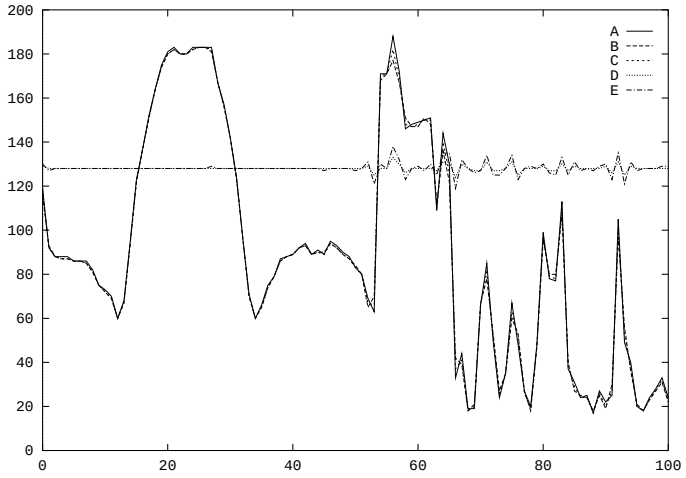

Figure 3: Quasi-interpolation Error Effects. The interpolation error is biased by 128. The graph 'A' gives the input data. 'B' gives the cubic quasi-interpolation ($k = 2$), 'C' gives the quadratic quasi-interpolation ($k = 2$) of the scan-line. Graph's 'D' and 'E' give the interpolation error for the 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.

$$\max_{l\in\mathbb{Z}} |(Q_k f - J_m f)(l)| \leq (\max_{l\in\mathbb{Z}} f(l) + \min_{l\in\mathbb{Z}} f(l))\frac{1}{2}\beta_m^{k+1}, \quad \beta_3 = \frac{1}{2} \text{ and } \beta_4 = \frac{2}{3}, \tag{24}$$

where the sequence $\{f_k\}_{k\in\mathbb{Z}}$ is bounded and m is the order of the underlying polynomial. In the above expression, $(Q_k f)(x)$ is a kth order quasi-interpolant, while $(J_m f)(x)$ is an mth order (true) interpolant. It is easy to see that, for any order k of quasi-interpolation, the 3rd order (quadratic) quasi-interpolant will provide a better fit.

An explanation for this somewhat counter-intuitive result may be gleaned by examining the weights used to compute the quasi-interpolant (cf. Appendix). The cubic weighting function decays more slowly; consequently more importance is attached to non-local data and this results in the suppression of local features.

4. Conclusion

The choice of polynomial on which to base a spline MRA plays a fundamental role in determining the speed and accuracy with which the wavelet representation may be determined. In the case of the semi-orthogonal spline wavelets we employed the quadratic MRA was shown to be superior to the cubic one in both decomposition and reconstruction times, while also permitting greater accuracy for a corresponding number of terms in the truncated IIR filters used to perform the decomposition. Furthermore, for applications in which one desires a cheap method of maximally (smoothly) approximating an arbitrary image, one may employ a local quasi-interpolation scheme under which it is both cheaper and more accurate to perform a quadratic approximation than a cubic one.

We believe that the arguments stated above provide compelling reasons to favour the quadratic over the cubic scheme: the quadratic representation possess a sufficient degree of smoothness to enable accurate approximations of smooth data, while requiring substantially less computation than its cubic counterparts.

A. Appendix

Details are provided on several important implementational issues. For additional information, the reader is directed to [4, 5, 6].

A.1 Symmetric signal boundary extensions

There are two major classes of signal extensions [2]:

- 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 4 and Figure 5).

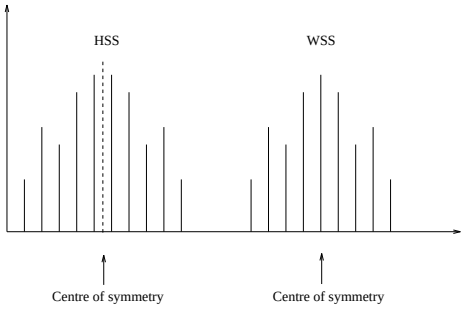

Figure 4: Symmetric extensions. The dashed line indicates the centre of symmetry.

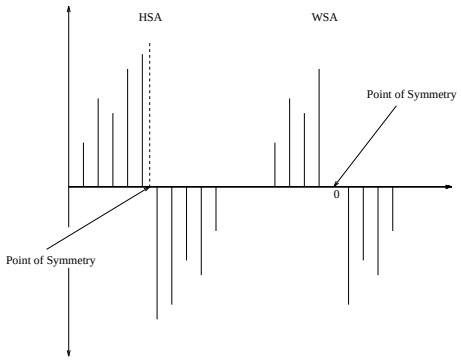

Figure 5: Anti-symmetric extensions. The centre of symmetry is indicated by a dashed line. Observe that WSA forces the sequence to assume the value zero at the point of symmetry. HSA enforces this zero condition at a half-sample point of symmetry i.e., an 'index' of the form $k + \frac{1}{2},\ k \in \mathbb{Z}$.

It is assumed that the sequences are extended periodically using the given boundary conditions. Conceptually, one extends the sequence (using the indicated BCs) 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 BCs have been chosen correctly, this periodic extension ensures that we can deduce the correct sequence value of any index.

Definition 1 *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.*

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.

A.2 Quasi-interpolation
Quasi-interpolation may be used on a data set provided

1. the data is bounded and continuous and
2. the data satisfies the Fix-Strang conditions.

We assume that our pixel values may be considered a sampling, on $\mathbb{Z} \times \mathbb{Z}$, of a continuous image function, $I(x, y)$.

Definition 2 *If a 2-D, symmetric, origin-centred piece-wise polynomial function $\Phi(x, y)$ satisfies the* Fix-Strang *conditions [3]*

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

then one may define the kth order quasi-interpolant, $(Q_k I)(x, y), \;\; k \in \mathbb{Z}^+$, *of $I \in C^s(\mathbb{R}^2)$ as*

$$(Q_k I)(x, y) \equiv \sum_l \sum_m (\lambda_k I)(l, m)\Phi(x - l, y - m). \tag{25}$$

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

$$\{(\lambda_k I)(i)\} = (\delta - m + \cdots + (-1)^k \underbrace{m * \cdots * m}_{k \text{ times}}) * I^0(i), i \in \mathbb{Z}^2, \tag{26}$$

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

$$m_{i,j} = \begin{cases} \Phi(0, 0) - 1 & \text{for i, j} = 0; \\ \Phi(i, j) & \text{for i, j} \neq 0. \end{cases} \tag{27}$$

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 $\{(\lambda_k I)(i, j)\}$

$$\begin{bmatrix} O & x & X & x & O \\ x & Q & I & Q & x \\ X & I & \# & I & X \\ x & Q & I & Q & x \\ O & x & X & x & O \end{bmatrix}$$

$m=3$	$k=2$	$k=1$
Q	-0.0117187	-1/64
I	-0.1699218	-3/32
$\#$	1.665	23/16
x	0.00293	-
X	0.0092773	-
O	0.0002441	-

Table 6: The arrangement of coefficients of the quasi-interpolation operator

- 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 coefficients of the λ sequences for the quadratic case are given in Table 6.

In Table 6, the matrix represents the support (i.e. grid-points)over which the coefficients of the intensity samples I_{ij} are non-zero. The centre of the matrix represents the coefficient of I_{ij}.

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

$$\Phi(x,y) \equiv N_m(x+\frac{m}{2})N_m(y+\frac{m}{2}) \tag{28}$$

where the shift is required to centre the cardinal B-spline functions. Equation 25 then looks like

$$(Q_k I)(x,y) = \sum_l \sum_n (\lambda_k I)(l,n) N_m(x+\frac{m}{2}-l) N_m(y+\frac{m}{2}-n). \tag{29}$$

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:

$$(Q_k I)(x,y) = \sum_l \sum_m (\lambda_k I)(l+\frac{m}{2}, m+\frac{m}{2}) N_m(x-l) N_m(y-l). \tag{30}$$

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

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? We 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.

References

1. 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.
2. C. Brislawn. Classification of symmetric wavelet transforms. Technical report, Los Alamos National Laboratories, March 1993.
3. C. Chui and H. Diamond. A natural formulation of quasi-interpolation by multi-variate splines. *Proceedings of the American Mathematical Society*, 99(4), 1987.
4. C. K. Chui. *An Introduction to Wavelets: Wavelet Analysis and its Applications*, volume one. Academic Press, Boston, 1992.
5. P. C. Marais. Spline wavelet image coding and synthesis for a vlsi based difference engine. Master's thesis, University Of Cape Town, 1994.
6. P.C. Marais, E.H. Blake, and A.A.M. Kuijk. A spline-wavelet image decomposition on a difference engine. *CWI Quarterly*, 6(4), December 1993.
7. P.C. Marais, E.H. Blake, and A.A.M. Kuijk. Adaptive spline wavelet image encoding and real-time synthesis. In *Proceedings the IEEE conference on Image Processing*, volume 3, pages 368–372, 1994.
8. M. Unser, A. Aldroubi, and M. Eden. A family of polynomial spline wavelet transforms. *Signal Processing*, 30:141–162, 1993.