IMAGE NOISE CANCELLATION USING B-SPLINE WAVELETS

By

NIKHIL NAGRANI - 061265

ABHILASH SRIVASTAVA - 061401

VAIBHAV SRIVASTAVA – 061459

May-2010

Submitted in partial fulfilment of the Degree of Bachelor of Technology in Information Technology

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING AND INFORMATION TECHNOLOGY

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY-WAKNAGHAT

CERTIFICATE

This is to certify that the work entitled, **"Image Noise Cancellation using B-Spline wavelets"** submitted by **Nikhil Nagrani** in partial fulfilment for the award of degree of Bachelor of Technology in Computer Science Engineering and by **Abhilash Srivastava, Vaibhav Srivastava** in partial fulfilment for the award of degree of Bachelor of Technology in Information Technology submitted in the Department of Computer Science Engineering & Information Technology at JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

(Project supervisor)

[illegible] 25/5/10

Dr. Rajesh Siddavatam

Asst. Professor

Department of Computer Science Engineering & Information Technology,

Jaypee University of Information Technology,

Waknaghat, Solan-173215, India

ACKNOWLEDGMENTS

Computer Science is fast developing subject & new dimensions are being added from time to time. During the last decade many new dimensions have been introduced and many old ones have been redefined. In light of new development and recent findings, we devote the task that was asked from us during our IVth year at Jaypee University of Information and Technology to "Image Noise Cancellation using B-Spline wavelets" and to the needs of computer science.

We would like to thank **Dr. Rajesh Siddavatam** who helped us during the task given to us. We thank him again for providing us with the necessary facility for conducting this task.

Nikhil Nagrani	(061265)
Abhilash Srivastava	(061401)
Vaibhav Srivastava	(061459)

Table of Contents

List of Figures

ABSTRACT

In this project, we propose to develop an approach for image noise cancellation based on the B-spline wavelets. The discussed B-spline based multiscale/resolution representation is based upon a perfect reconstruction analysis/synthesis point of view. We also present a straightforward computationally efficient approach for B-spline basis calculations that is based upon matrix multiplication and avoids any extra generated basis.The B-spline analysis discussed here can be utilized for different imaging applications such as compression, prediction, and denoising.

Chapter I

Introduction

1.1 Objective

In this project, the technique for image noise cancellation based on the B-spline wavelets is proposed and implemented using MATLAB. The proposed B-spline based multiscale/resolution representation is based upon a perfect reconstruction analysis/synthesis point of view. Our proposed B-spline analysis can be utilized for different imaging applications such as compression, prediction, and denoising. We also present a straightforward computationally efficient approach for B-spline basis calculations that is based upon matrix multiplication and avoids any extra generated basis.

1.2 Methodology

The proposed B-spline wavelets are used in Image noise cancellation, as follows:

1) Decompose the received noisy image using the proposed B-spline wavelet to an arbitrary n decomposition level.
Denote the resulting detail and approximation coefficients by *d*1, *d*2, . . ., *dn*,*an,* respectively.

2) As the next step, The Matlab software/function *wavedec2* under 2-D Discrete Wavelets Toolbox category mainly implements the wavelet decomposition of the obtained matrix X representing the noised image at level N, using the wavelet named in string '*wname*' as described below,

[C,S] = wavedec2(X,N,'*wname*')

Outputs are the decomposition vector C and the corresponding bookkeeping matrix S.

3) Finally, **Image noise cancellation** is achieved using the denoised detail coefficients obtained during the second step as given above, together with

the synthesis coefficient matrix obtained at the first step using the proposed B-spline reconstruction system.

1.3 Resources and Limitations

We have used Matlab 7.0 for implementation of our image noise cancellation method. Some of the functions we used are:

- **wavedec2** function : Multilevel 2-D wavelet decomposition
- **wfilters** : Wavelet filters

  [Lo_D,Hi_D,Lo_R,Hi_R] = wfilters('*wname*') computes four filters associated with the orthogonal or biorthogonal wavelet named in the string '*wname*'.

- **appcoef2** : 2-D approximation coefficients

  appcoef2 is a two-dimensional wavelet analysis function. It computes the approximation coefficients of a two-dimensional signal. The syntaxes allow you to give the wavelet name or the filters as inputs.

- **detcoef2** : 2-D detail coefficients

  detcoef2 is a two-dimensional wavelet analysis function. D = detcoef2(O,C,S,N) extracts from the wavelet decomposition structure [C,S] the horizontal, vertical, or diagonal detail coefficients for O = 'h'(or 'v' or 'd', respectively), at level N

Limitations:

We didn't have the B-spline wavelet related functions available in the matlab so we designed our own functions for the same to apply the B-spline filter scheme on the image for the cancellation of the noise.

Our Approach didn't remove the noise completely but to an extent that is sufficient to further process the image.

Chapter II

B-Splines

2.1 SPLINES

Splines are piecewise polynomial curves that are differentiable up to a prescribed order. The simplest example is a piecewise linear C_0 spline, i.e.,a polygonal curve

The name spline is derived from elastic beams, so-called splines, used by draftsmen to lay out broad sweeping curves in ship design. Held in place by a number of heavy weights, these physical splines assume a shape that minimizes the strain energy. This property is approximately shared by the mathematical cubic C_2 splines.

A curve s(u) is called a spline of degree n with the knots $a0$.......am, where ai<=ai+1 and ai < ai+n+1 for all possible i, if

s(u) *is n- r times differentiable at any r-fold knot, and* s(u) *is a polynomial of degree<=n over each knot interval* [ai,ai+1], *for i* = 0......m-1.

It is also common to refer to a spline of degree n as a spline of order n+1. Figures 2.1 and 2.2 show examples of splines with simple knots obtained by Stark's construction.

The inner and end B´ezier points are marked by hollow and solid dots, respectively.

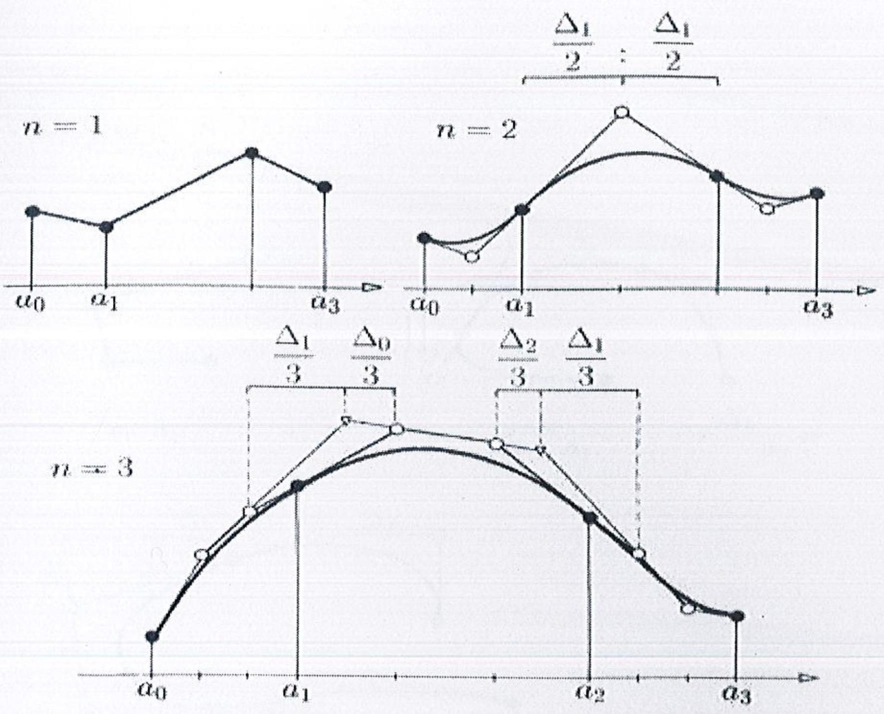


Fig 2.1 : B-Splines Functions of degree 1,2 & 3

2.2 B-SPLINES

As with the B′ezier representation of polynomial curves, it is desirable to write a spline s(u) as an affine combination of some control points c_i, namely

$$s(u) = \sum c_i N_i^n(u) \ ,$$

where the $N_i^n(u)$ are basis spline functions with minimal support and certain continuity properties. Schoenberg introduced the name B-splines for these functions [Schoenberg '67]. Their B′ezier polygons can be constructed by Stark's theorem.

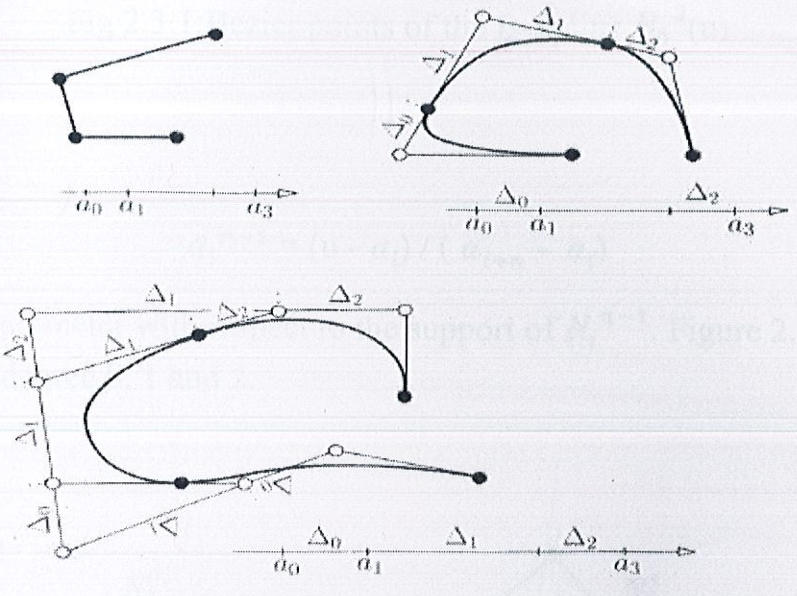

a_0 a_1 a_3
Δ_0 Δ_1 Δ_2
a_0 a_1 a_3
Δ_0 Δ_1 Δ_2
a_0 a_1 a_3

Fig 2.2:Parametric splines of degree 1,2 and 3.

2.3 A recursive definition of B-splines

To define B-splines, let (a_i) be a, for simplicity, biinfinite and strictly increasing sequence of knots, which means $a_i < a_i$ +1, for all *i.* We define the B-splines N_i^n with these knots by the recursion formula

$$N_i^0(u) = \begin{cases} 1 & \text{if } \quad u \in [a_i, a_{i+1}) \\ 0 & \text{otherwise} \end{cases}$$

And

$$N_i^n(u) = \alpha_i^{n-1} N_i^{n-1}(u) + (1 - \alpha_{i+1}^{n-1}) N_{i+1}^{n-1}(u)$$

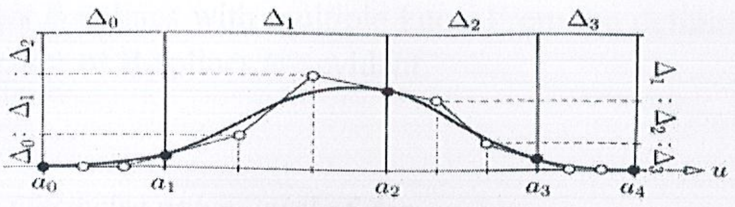


Fig 2.3.1:Bezier points of the B-Spline $N_0^{\ 3}$(u)

where

$$\alpha_i^{\ n-1} = (u - a_i) / (a_{i+n} - a_i)$$

is the local parameter with respect to the support of $N_i^{\ n-1}$. Figure 2.3.2 shows B-splines of degree 0, 1 and 2.

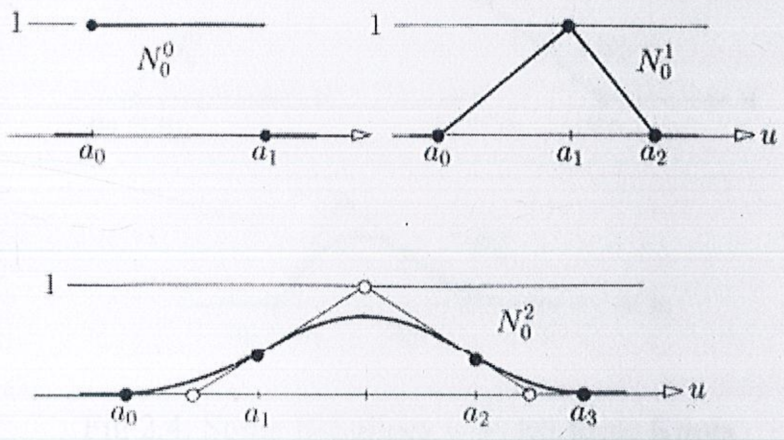


Fig 2.3.2:B-Splines of degree 0,1,2

In case of multiple knots, the B-splines $N_i^{\ n}$ are defined by the same recursion formula and the convention

$$N_i^{\ r-1} = N_i^{\ r-1} / (a_{i+r} - a_i) = 0 \ if \ a_i = a_{i+r}$$

Figure 2.4 shows B-splines with multiple knots.From the definition above, the following properties of B-splines are evident.

- $N_i^n(u)$ *is piecewise polynomial of degree n,*
- $N_i^n(u)$ *is positive in* (a_i, a_{i+n+1})
- $N_i^n(u)$ *is zero outside of* (a_i, a_{i+n+1})
- $N_i^n(u)$ *is right side continuous*

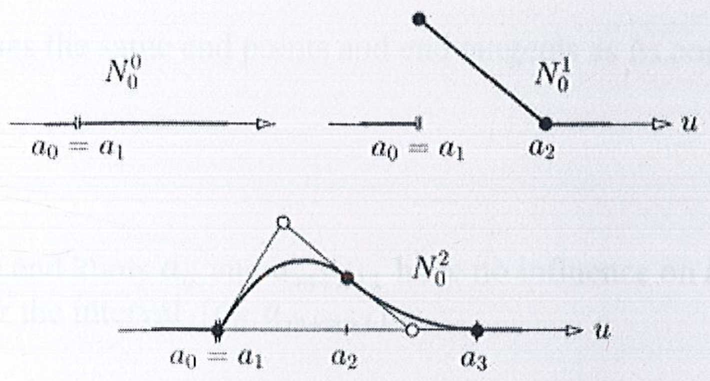


Fig 2.4: Some B-Splines with Multiple Knots

2.4 B-SPLINE PROPERTIES

- The B-splines of degree n with a given knot sequence that do not vanish over some knot interval are linearly independent over this interval.

- A dimension count shows that the B-splines $N_0{}^n$..........$N_0{}^m$ with the knots a_0.........a_{m+n+1} form a basis for all splines of degree *n* with support $[a_0, a_{m+n+1})$ and the same knots.

- Similarly, the B-splines $N_0{}^n$..........$N_0{}^m$ over the knots a_0.........a_{m+n+1} restricted to the interval $[a_0, a_{m+n+1}]$ form a basis for all splines of degree *n* restricted to the same interval.

- The B-splines of degree *n* form a partition of unity, i.e.,

$$\sum_{i=0}^{m} N_i^n(u) = 1, \quad \text{for} \quad u \in [a_n, a_{m+1})$$

- A spline s$[a_0, a_{m+1}]$ of degree *n* with *n*-fold end knots,

$$(a_0 =)a_1 = \ldots = a_n \quad \text{and} \quad a_{m+1} = \ldots = a_{m+n}(= a_{m+n+1})$$

  has the same end points and end tangents as its control polygon.

- The end knots a_0 and a_{m+n+1} have no influence on $N_0{}^n$ and $N_m{}^n$ over the interval $[a_0, a_{m+n+1}]$

- The B-splines are positive over the interior of their support

$$N_i^n(u) > 0 \quad \text{for} \quad u \in (a_i, a_{i+n+1})$$

- The B-splines have compact support :

$$\text{supp} N_i^n = [a_i, a_{i+n+1}]$$

- The B-splines satisfy the de Boor, Mansfield, Cox recursion formula

$$N_i^n(u) = \alpha_i^{n-1} N_i^{n-1}(u) + (1 - \alpha_{i+1}^{n-1}) N_{i+1}^{n-1}(u)$$

where

$\alpha_i^{\,n-1}$ = (u - a_i) / (a_{i+n} - a_i) represents the local parameter over the

support $of\ N_i^{\,n-1}$

- The derivative of a single B-spline is given by:

$$\frac{d}{du} N_i^n(u) = \frac{n}{u_{i+n} - u_i} N_i^{n-1}(u) - \frac{n}{u_{i+n+1} - u_{i+1}} N_{i+1}^{n-1}(u)$$

- The B-spline representation of a spline curve is invariant under affine Maps

- Any segment $s_j[a_j, a_{j+1})$ of an *n*th degree spline lies in the convex hull of its n + 1 control points $c_{j-n} \ldots\ldots c_j$

2.5 B-spline basis functions

Bézier basis functions are used as weights. B-spline basis functions will be used the same way; however, they are much more complex. There are two interesting properties that are not part of the Bézier basis functions, namely: (1) the domain is subdivided by knots, and (2) basis functions are not non-zero on the entire interval. In fact, each B-spline basis function is non-zero on a few adjacent subintervals and, as a result, B-spline basis functions are quite "local".

Let U be a set of m + 1 non-decreasing numbers, $u_0 <= u_2 <= u_3 <= \ldots <= u_m$. The u_i's are called *knots*, the set U the *knot vector*, and the half-open interval [u_i, u_{i+1}) the *i-th knot span*. Note that since some u_i's may be equal, some knot spans may not exist. If a knot u_i appears k times (*i.e.*, $u_i = u_{i+1} = ... = u_{i+k-1}$), where $k > 1$, u_i is a *multiple knot* of *multiplicity* k, written as $u_i(k)$. Otherwise, if u_i appears only once, it is a *simple knot*. If the knots are equally spaced (*i.e.*, $u_{i+1} - u_i$ is a constant for $0 <= i <= m - 1$), the knot vector or the knot sequence is said *uniform*; otherwise, it is *non-uniform*.

The knots can be considered as division points that subdivide the interval $[u_0, u_m]$ into knot spans. All B-spline basis functions are supposed to have their domain on $[u_0, u_m]$. In this note, we use $u_0 = 0$ and $u_m = 1$ frequently so that the domain is the closed interval [0,1].

To define B-spline basis functions, we need one more parameter, the degree of these basis functions, p. The i-th B-spline basis function of degree p, written as $N_{i,p}(u)$, is defined recursively as follows:

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \le u < u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u)$$

The above is usually referred to as the Cox-de Boor recursion formula. This definition looks complicated; but, it is not difficult to understand. If the degree is zero (i.e., $p = 0$), these basis functions are all step functions and this is what the first expression says. That is, basis function $N_{i,0}(u)$ is 1 if u is in the i-th knot span $[u_i, u_{i+1})$. For example, if we have four knots $u_0 = 0$, $u_1 = 1$, $u_2 = 2$ and $u_3 = 3$, knot spans 0, 1 and 2 are [0,1), [1,2), [2,3) and

the basis functions of degree 0 are $N_{0,0}(u) = 1$ on [0,1) and 0 elsewhere, $N_{1,0}(u) = 1$ on [1,2) and 0 elsewhere, and $N_{2,0}(u) = 1$ on [2,3) and 0 elsewhere. This is shown below:

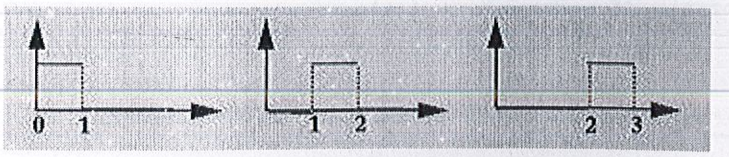


Figure 2.5.1

To understand the way of computing $N_{i,p}(u)$ for p greater than 0, we use the triangular computation scheme. All knot spans are listed on the left (first) column and all degree zero basis functions on the second. This is shown in the following diagram.

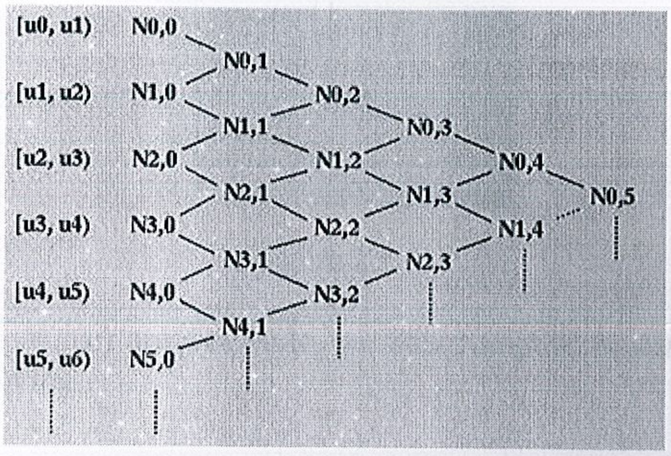


Figure 2.5.2

To compute $N_{i,1}(u)$, $N_{i,0}(u)$ and $N_{i+1,0}(u)$ are required. Therefore, we can compute $N_{0,1}(u)$, $N_{1,1}(u)$, $N_{2,1}(u)$, $N_{3,1}(u)$ and so on. All of these $N_{i,1}(u)$'s are written on the third column. Once all $N_{i,1}(u)$'s have been computed, we can compute $N_{i,2}(u)$'s and put them on the fourth column. This process continues until all required $N_{i,p}(u)$'s are computed.

In the above, we have obtained $N_{0,0}(u)$, $N_{1,0}(u)$ and$N_{2,0}(u)$ for the knot vector U = { 0, 1, 2, 3 }. Let us compute $N_{0,1}(u)$ and $N_{1,1}(u)$. To compute $N_{0,1}(u)$, since i = 0 and p = 1, from the definition we have

$$N_{0,1}(u) = \frac{u - u_0}{u_1 - u_0} N_{0,0}(u) + \frac{u_2 - u}{u_2 - u_1} N_{1,0}(u)$$

Since $u_0 = 0$, $u_1 = 1$ and $u_2 = 2$, the above becomes

$$N_{0,1}(u) = uN_{0,0}(u) + (2 - u)N_{1,0}(u)$$

Since $N_{0,0}(u)$ is non-zero on [0,1) and $N_{1,0}(u)$ is non-zero on [1,2), if u is in [0,1) (*resp.*, [1,2)), only $N_{0,0}(u)$ (*resp.*, $N_{1,0}(u)$) contributes to $N_{0,1}(u)$. Therefore, if u is in [0,1), $N_{0,1}(u)$ is $uN_{0,0}(u) = u$, and if u is in [1,2), $N_{0,1}(u)$ is $(2 - u)N_{1,0}(u) = (2 - u)$. Similar computation gives $N_{1,1}(u) = u - 1$ if u is in [1,2), and $N_{1,1}(u) = 3 - u$ if u is in [2,3). In the following figure, the black and red lines are $N_{0,1}(u)$ and $N_{1,1}(u)$, respectively. Note that $N_{0,1}(u)$ (*resp.*, $N_{1,1}(u)$) is non-zero on [0,1) and [1,2) (*resp.*, [1,2) and [2,3)).

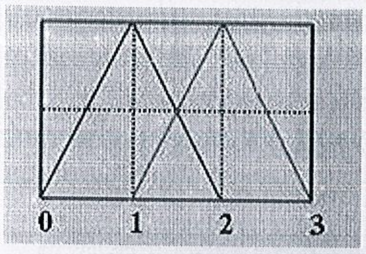


Figure 2.5.3

Once $N_{0,1}(u)$ and $N_{1,1}(u)$ are available, we can compute $N_{0,2}(u)$. The definition gives us the following:

$$N_{0,2}(u) = \frac{u - u_0}{u_2 - u_0} N_{0,1}(u) + \frac{u_3 - u}{u_3 - u_1} N_{1,1}(u)$$

Plugging in the values of the knots yields

$$N_{0,2}(u) = 0.5uN_{0,1}(u) + 0.5(3 - u)N_{1,1}(u)$$

Note that $N_{0,1}(u)$ is non-zero on [0,1) and [1,2) and $N_{1,1}(u)$ is non-zero on [1,2) and [2,3). Therefore, we have three cases to consider:

1. u is in [0,1):
In this case, only $N_{0,1}(u)$ contributes to the value of $N_{0,2}(u)$. Since $N_{0,1}(u)$ is u, we have ,

$$N_{0,2}(u) = 0.5u^2$$

2. u is in [1,2):
In this case, both $N_{0,1}(u)$ and $N_{1,1}(u)$ contribute to $N_{0,2}(u)$. Since $N_{0,1}(u) = 2 - u$ and $N_{1,1}(u) = u - 1$ on [1,2), we have ,

$$N_{0,2}(u) = (0.5u)(2 - u) + 0.5(3 - u)(3 - u) = 0.5(-3 + 6u - 2u^2)$$

3 u is in [2,3):
In this case, only $N_{1,1}(u)$ contributes to $N_{0,2}(u)$. Since $N_{1,1}(u) = 3 - u$ on [2,3), we have ,

$$N_{0,2}(u) = 0.5(3-u)(3-u) = 0.5(3-u)^2$$

If we draw the curve segment of each of the above three cases, we shall see that two adjacent curve segments are joined together to form a curve at the knots. More precisely, the curve segments of the first and second cases join together at $u = 1$, while the curve segments of the second and third cases join at $u = 2$. Note that the composite curve shown here is smooth. But in general it is not always the case if a knot vector contains multiple knots.

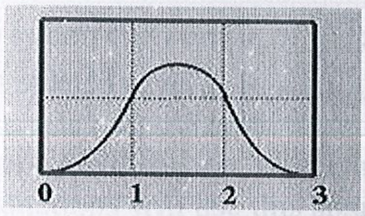


Figure 2.5.4

2.6 Two Important Observations

Since $N_{i,1}(u)$ is computed from $N_{i,0}(u)$ and $N_{i+1,0}(u)$ and since $N_{i,0}(u)$ and $N_{i+1,0}(u)$ are non-zero on span $[u_i, u_{i+1})$ and $[u_{i+1}, u_{i+2})$, respectively, $N_{i,1}(u)$ is non-zero on these two spans. In other words, $N_{i,1}(u)$ is non-zero on $[u_i, u_{i+2})$. Similarly, since $N_{i,2}(u)$ depends on $N_{i,1}(u)$ and $N_{i+1,1}(u)$ and since these two basis functions are non-zero on $[u_i, u_{i+2})$ and $[u_{i+1}, u_{i+3})$, respectively, $N_{i,2}(u)$ is non-zero on $[u_i, u_{i+3})$. In general, to determine the non-zero domain of a basis function $N_{i,p}(u)$, one can trace back using the triangular computation scheme until it reaches the first column. The covered spans are the non-zero domain of this basis function. For example, suppose we want to find out the non-zero domain of $N_{1,3}(u)$. Based on the above discussion, we can trace back in the north-west and south-west directions until the first column is reached as shown with the blue dotted line in the following diagram. Thus, $N_{1,3}(u)$ is non-zero on $[u_1, u_2)$, $[u_2, u_3)$, $[u_3, u_4)$ and $[u_4, u_5)$. Or, equivalently, it is non-zero on $[u_1, u_5)$.

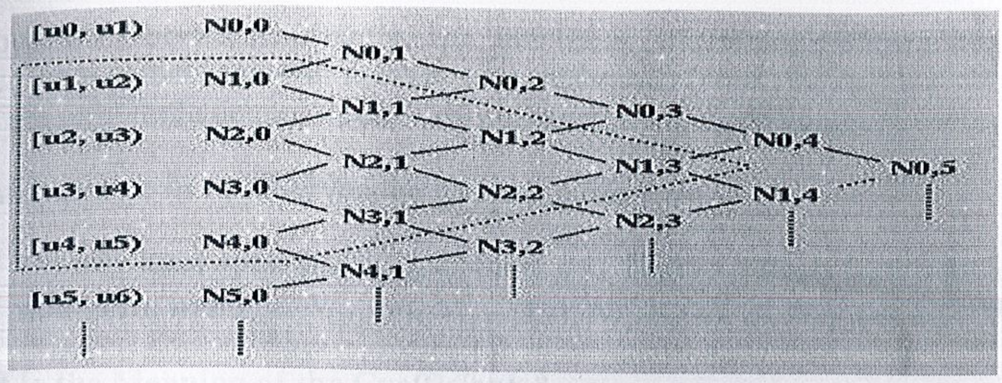


Figure 2.6.1

In summary, we have the following observation:

> Basis function $N_{i,p}(u)$ is non-zero on $[u_i, u_{i+p+1})$. Or, equivalently, $N_{i,p}(u)$ is non-zero on $p+1$ knot spans $[u_i, u_{i+1})$, $[u_{i+1}, u_{i+2})$, ..., $[u_{i+p}, u_{i+p+1})$.

Next, we shall look at the opposite direction. Given a knot span $[u_i, u_{i+1})$, we want to know which basis functions will use this span in its computation. We can start with this knot span and draw a north-east bound arrow and a south-east bound arrow. All basis functions enclosed in this wedge shape use $N_{i,0}(u)$ (why?) and hence are non-zero on this span. Therefore, all degree p basis functions that are non-zero on $[u_i, u_{i+1})$ are the intersection of this wedge and the column that contains all $N_{i,p}(u)$'s. In fact, this column and the two arrows form an equilateral triangle with this column being the vertical side. Counting from $N_{i,0}(u)$ to $N_{i,p}(u)$ there are $p+1$ columns. Therefore, the vertical side of the equilateral triangle must have at most $p+1$ entries, namely $N_{i,p}(u)$, $N_{i-1,p}(u)$, $N_{i-2,p}(u)$, ..., $N_{i-p+2,p}(u)$, $N_{i-p+1,p}(u)$ and $N_{i-p,p}(u)$.

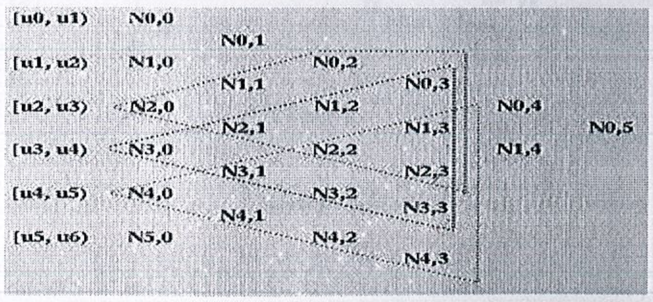


Figure 2.6.2

Let us take a look at the above diagram. To find all degree 3 basis functions that are non-zero on $[u_4, u_5)$, draw two arrows and all functions on the vertical edges are what we want. In this case, they are $N_{1,3}(u)$, $N_{2,3}(u)$, $N_{3,3}(u)$, and $N_{4,3}(u)$. This is shown with the orange triangle. The blue (*resp.*, red) triangle shows the degree 3 basis functions that are non-zero on $[u_3, u_4)$ (*resp.*, $[u_2, u_3)$). Note that there are only three degree three basis polynomials that are non-zero on $[u_2, u_3)$.

In summary, we have observed the following property.

> **On any knot span $[u_i, u_{i+1})$, at most $p+1$ degree p basis functions are non-zero, namely: $N_{i-p,p}(u)$, $N_{i-p+1,p}(u)$, $N_{i-p+2,p}(u)$, ..., $N_{i-1,p}(u)$ and $N_{i,p}(u)$,**

What Is the Meaning of the Coefficients?

Finally, let us investigate the meaning of the coefficients in the definition of $N_{i,p}(u)$. As $N_{i,p}(u)$ is being computed, it uses $N_{i,p-1}(u)$ and $N_{i+1,p-1}(u)$. The former is non-zero on $[u_i, u_{i+p})$. If u is in this half-open interval, then $u - u_i$ is the distance between u and the *left* end of this interval, the interval length is $u_{i+p} - u_i$, and $(u - u_i) / (u_{i+p} - u_i)$ is the ratio of the above mentioned distances and is always in the range of 0 and 1. See the diagram below. The second term, $N_{i,p-1}(u)$, is non-zero on $[u_{i+1}, u_{i+p+1})$. If u is in this interval, then $u_{i+p+1} - u$ is the distance from u to the *right* end of this interval, $u_{i+p+1} - u_{i+1}$ is the length of the interval, and $(u_{i+p+1} - u) / (u_{i+p+1} - u_{i+1})$ is the ratio of these two distances and its value is in the range of 0 and 1. Therefore, $N_{i,p}(u)$ is a linear combination of $N_{i,p-1}(u)$ and $N_{i+1,p-1}(u)$ with two coefficients, both linear in u, in the range of 0 and 1.

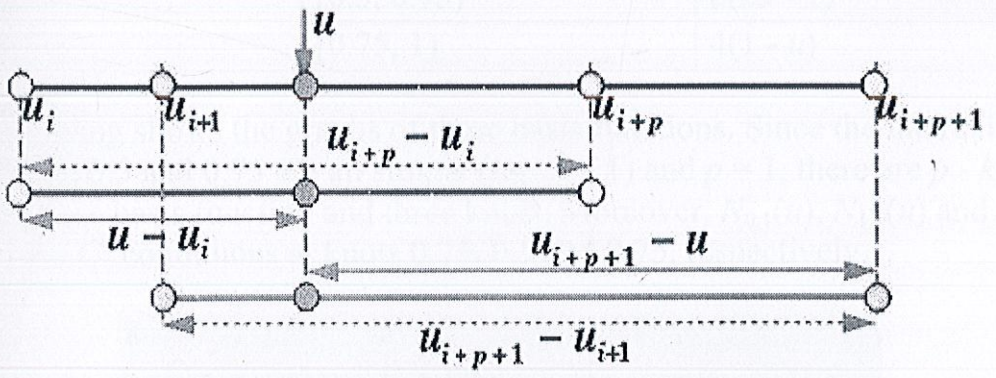


Figure 2.6.3

2.7 B-spline Basis Functions: Computation Examples

2.7.1 Simple Knots

Suppose the knot vector is $U = \{ 0, 0.25, 0.5, 0.75, 1 \}$. Hence, $m = 4$ and $u_0 = 0$, $u_1 = 0.25$, $u_2 = 0.5$, $u_3 = 0.75$ and $u_4 = 1$. The basis functions of degree 0 are easy. They are $N_{0,0}(u)$, $N_{1,0}(u)$, $N_{2,0}(u)$ and $N_{3,0}(u)$ defined on knot span [0,0.25,), [0.25,0.5), [0.5,0.75) and [0.75,1), respectively, as shown below

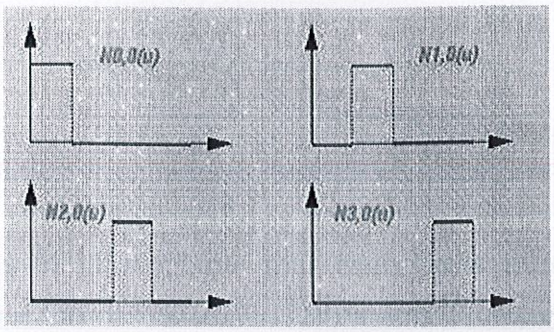


Fig 2.7.1.1

The following table gives the result of all $N_{i,1}(u)$'s:

Basis Function	***Range***	***Equation***
$N_{0,1}(u)$	[0, 0.25)	$4u$
	[0.25, 0.5)	$2(1 - 2u)$
$N_{1,1}(u)$	[0.25, 0.5)	$4u - 1$
	[0.5, 0.75)	$3 - 4u$
$N_{2,1}(u)$	[0.5, 0.75)	$2(2u - 1)$
	[0.75, 1)	$4(1 - u)$

The following shows the graphs of these basis functions. Since the internal knots 0.25, 0.5 and 0.75 are all simple (*i.e.*, $k = 1$) and $p = 1$, there are $p - k + 1 = 1$ non-zero basis function and three knots. Moreover, $N_{0,1}(u)$, $N_{1,1}(u)$ and $N_{2,1}(u)$ are C^0 continuous at knots 0.25, 0.5 and 0.75, respectively.

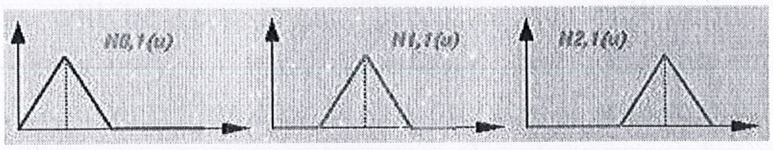


Fig: 2.7.1.2

From $N_{i,1}(u)$'s, one can compute the basis functions of degree 2. Since $m = 4$, $p = 2$, and $m = n + p + 1$, we have $n = 1$ and there are only two basis functions of degree 2: $N_{0,2}(u)$ and $N_{1,2}(u)$. The following table is the result:

Basis Function	***Range***	***Equation***

$N_{0,2}(u)$	[0, 0.25)	$8u^2$
	[0.25, 0.5)	$-1.5 + 12u - 16\,u^2$
	[0.5, 0.75)	$4.5 - 12u + 8\,u^2$
$N_{1,2}(u)$	[0.25, 0.5)	$0.5 - 4u + 8u^2$
	[0.5, 0.75)	$-1.5 + 8u - 8u^2$
	[0.75, 1)	$8(1 - u)^2$

The following figure shows the two basis functions. The three vertical blue lines indicate the positions of knots. Note that each basis function is a composite curve of three degree 2 curve segments. For example, $N_{0,2}(u)$ is the green curve, which is the union of three parabolas defined on [0,0.25), [0.25, 0.5) and [0.5,0.75). These three curve segments join together forming a smooth bell shape. Please verify that $N_{0,2}(u,)$ (*resp.*, $N_{1,2}(u)$) is C^1 continuous at its knots 0.25 and 0.5 (*resp.*, 0.5 and 0.75). As mentioned on the previous page, at the knots, this composite curve is of C^1 continuity.

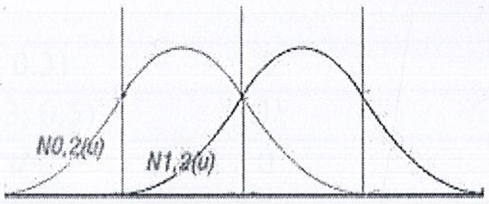


Fig.2.7.1.3

2.7.2 Knots with Positive Multiplicity

If a knot vector contains knots with positive multiplicity, we will encounter the case of 0/0 as will be seen later. Therefore, we shall define 0/0 to be 0. Fortunately, this is only for hand calculation. For computer implementation, there is an efficient algorithm free of this problem. Furthermore, if u_i is a knot of multiplicity k (*i.e.*, $u_i = u_{i+1} = ... = u_{i+k-1}$), then knot spans $[u_i,u_{i+1})$, $[u_{i+1},u_{i+2})$, ..., $[u_{i+k-2},u_{i+k-1})$ do not exist, and, as a result, $N_{i,0}(u)$, $N_{i+1,0}(u)$, ..., $N_{i+k-1,0}(u)$ are all zero functions.

Consider a knot vector $U = \{ 0, 0, 0, 0.3, 0.5, 0.5, 0.6, 1, 1, 1 \}$. Thus, 0 and 1 are of multiplicity 3 (*i.e.*, 0(3) and 1(3)) and 0.5 is of multiplicity 2 (*i.e.*, 0.5(2)). As a result, $m = 9$ and the knot assignments are

u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9

Let us compute $N_{i,0}(u)$'s. Note that since $m = 9$ and $p = 0$ (degree 0 basis functions), we have $n = m - p - 1 = 8$. As the table below shows, there are only four non-zero basis functions of degree 0: $N_{2,0}(u)$, $N_{3,0}(u)$, $N_{5,0}(u)$ and $N_{6,0}(u)$.

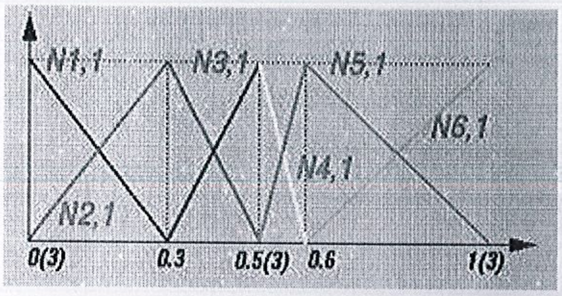
Basis Function	***Range***	***Equation***	***Comments***
$N_{0,0}(u)$	all u	0	since $[u_0, u_1)$ = [0,0) does not exist
$N_{1,0}(u)$	all u	0	since $[u_1, u_2)$ = [0,0) does not exist
$N_{2,0}(u)$	[0, 0.3)	1	
$N_{3,0}(u)$	[0.3, 0.5)	1	
$N_{4,0}(u)$	all u	0	since $[u_4, u_5)$ = [0.5,0.5) does not exist
$N_{5,0}(u)$	[0.5, 0.6)	1	
$N_{6,0}(u)$	[0.6, 1)	1	
$N_{7,0}(u)$	all u	0	since $[u_7, u_8)$ = [1,1) does not exist
$N_{8,0}(u)$	all u	0	since $[u_8, u_9)$ = [1,1) does not exist

Then, we proceed to basis functions of degree 1. Since p is 1, $n = m - p - 1 = 7$. The following table shows the result:

Basis Function	***Range***	***Equation***
$N_{0,1(u)}$	all u	0
$N_{1,1(u)}$	[0, 0.3)	1 - (10/3)u

$N_{2,1}(u)$	[0, 0.3)	$(10/3)u$
	[0.3, 0.5)	$2.5(1 - 2u)$
$N_{3,1}(u)$	[0.3, 0.5)	$5u - 1.5$
$N_{4,1}(u)$	[0.5, 0.6)	$6 - 10u$
$N_{5,1}(u)$	[0.5, 0.6)	$10u - 5$
	[0.6, 1)	$2.5(1 - u)$
$N_{6,1}(u)$	[0.6, 1)	$2.5u - 1.5$
$N_{7,1}(u)$	all u	0

The following figure shows the graphs of these basis functions:



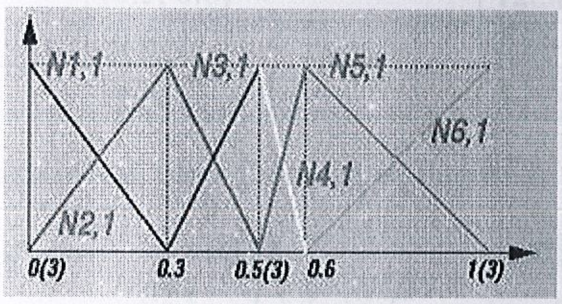


Fig 2.7.2.1

Let us take a look at a particular computation, say $N_{1,1}(u)$. It is computed with the following expression:

$$N_{1,1}(u) = \frac{u - u_1}{u_2 - u_1} N_{1,0}(u) + \frac{u_3 - u}{u_3 - u_2} N_{2,0}(u)$$

Plugging $u_1 = u_2 = 0$ and $u_3 = 0.3$ into this equation yields the following:

$$N_{1,1}(u) = \frac{u}{0} N_{1,0}(u) + \left(1 - \frac{10}{3}u\right) N_{2,0}(u)$$

Since $N_{1,0}(u)$ is zero everywhere, the first term becomes 0/0 and is defined to be zero. Therefore, only the second term has an impact on the result. Since $N_{2,0}(u)$ is 1 on [0,0.3), $N_{1,1}(u)$ is 1 - (10/3)u on [0,0.3).

Next, let us compute all $N_{i,2}(u)$'s. Since $p = 2$, we have $n = m - p - 1 = 6$. The following table contains all $N_{i,2}(u)$'s:

Basis Function	***Range***	***Equation***
$N_{0,2}(u)$	[0, 0.3)	$(1 - (10/3)u)^2$
$N_{1,2}(u)$	[0, 0.3)	$(20/3)(u - (8/3)u^2)$
	[0.3, 0.5)	$2.5(1 - 2u)^2$
$N_{2,2}(u)$	[0, 0.3)	$(20/3)u^2$
	[0.3, 0.5)	$-3.75 + 25u - 35u^2$
$N_{3,2}(u)$	[0.3, 0.5)	$(5u - 1.5)^2$
	[0.5, 0.6)	$(6 - 10u)^2$
$N_{4,2}(u)$	[0.5, 0.6)	$20(-2 + 7u - 6u^2)$
	[0.6, 1)	$5(1 - u)^2$
$N_{5,2}(u)$	[0.5, 0.6)	$12.5(2u - 1)^2$
	[0.6, 1)	$2.5(-4 + 11.5u - 7.5u^2)$
$N_{6,2}(u)$	[0.6, 1)	$2.5(9 - 30u + 25u^2)$

The following figure shows all basis functions of degree 2.

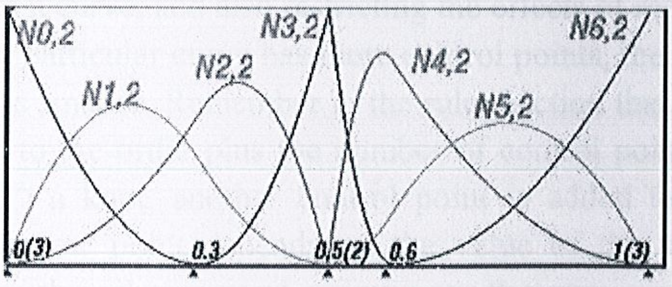


Fig 2.7.2.2

Let us pick a typical computation as an example, say $N_{3,2}(u)$. The expression for computing it is

$$N_{3,2}(u) = \frac{u - u_3}{u_5 - u_3} N_{3,1}(u) + \frac{u_6 - u}{u_6 - u_4} N_{4,1}(u)$$

Plugging in $u_3 = 0.3$, $u_4 = u_5 = 0.5$ and $u_6 = 0.6$ yields

$$N_{3,2}(u) = (5u - 1.5)N_{3,1}(u) + (6 - 10u)N_{4,1}(u)$$

Since $N_{3,1}(u)$ is non-zero on [0.3, 0.5) and is equal to $5u$ - 1.5, $(5u - 1.5)^2$ is the non-zero part of $N_{3,2}(u)$ on [0.3, 0.5). Since $N_{4,1}(u)$ is non-zero on [0.5, 0.6) and is equal to 6 - $10u$, $(6 - 10u)^2$ is the non-zero part of $N_{3,2}(u)$ on [0.5, 0.6).

Let us investigate the continuity issues at knot 0.5(2). Since its multiplicity is 2 and the degree of these basis functions is 2, basis function $N_{3,2}(u)$ is C^0 continuous at 0.5(2). This is why $N_{3,2}(u)$ has a sharp angle at 0.5(2). For knots not at the two ends, say 0.3, C^1 continuity is maintained since all of them are simple knots.

2.8 Adding Control points

It is possible to add control points to a curve without changing it's shape. This process is called *refinement* or *knot insertion.* Knot insertion is a way of adding more flexibility to a curve, and also restricting the effects of moving a particular control point. If a particular curve has more control points, the effect of moving any one of them is smaller. Remember in the rules section that the total number of knots is equal to the order plus the number of control points. So each time you add ("insert") a knot, another control point is added to the curve. The placement of the new point depends on the value of the new knot. Curve refinement moves the other control points near the new one to preserve the shape of the curve. As you add more knots, the control points get closer and closer to the actual curve.

In the example below, the original knot vector was [0 0 0 0 1 2 3 3 3 3]. In order to add some flexibility, two control points are added at 1.5 and 2.5, making the knot vector [0 0 0 0 1 1.5 2 2.5 3 3 3 3]. Note how moving a control point in the third picture effects a much smaller portion of the curve than in the second picture.

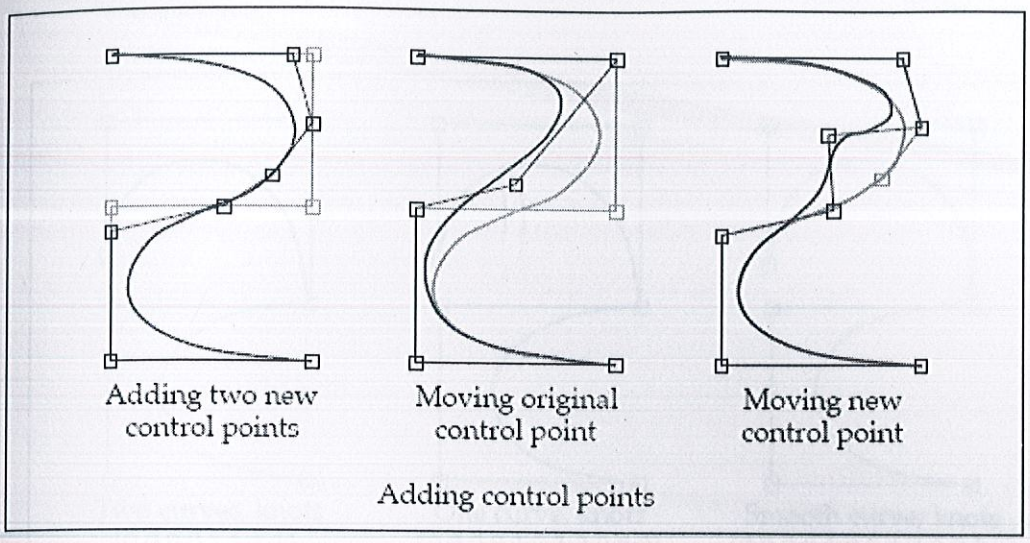


Figure 2.8

If you insert multiple knots with the same values, you'll introduce the discontinuities described in the previous section (of course, the kink or a break won't actually appear until you move the control points).

2.9 Joining Curves

As a final example of using knot vectors, it's sometimes desirable to join two adjoining curves and then smooth out the join between them. The knot vectors provide a straightforward mechanism for doing this. Suppose we have two pinned uniform cubic curves, with knot vectors starting at zero, that meet at a common point. Here's how to hook them up: 1. Add the last value in the first curve's knot vector to all the knots in the second curve.2. Remove the first control point and the *k* first knots from the second curve (*k* = order). Remove the last knot from the first

curve. Concatenate the second curve's control points and knot vector.3. To smooth the join, space apart the knots where the two knot vectors were joined.

This removes the discontinuity. As they're moved further apart, the join becomes less sharp.

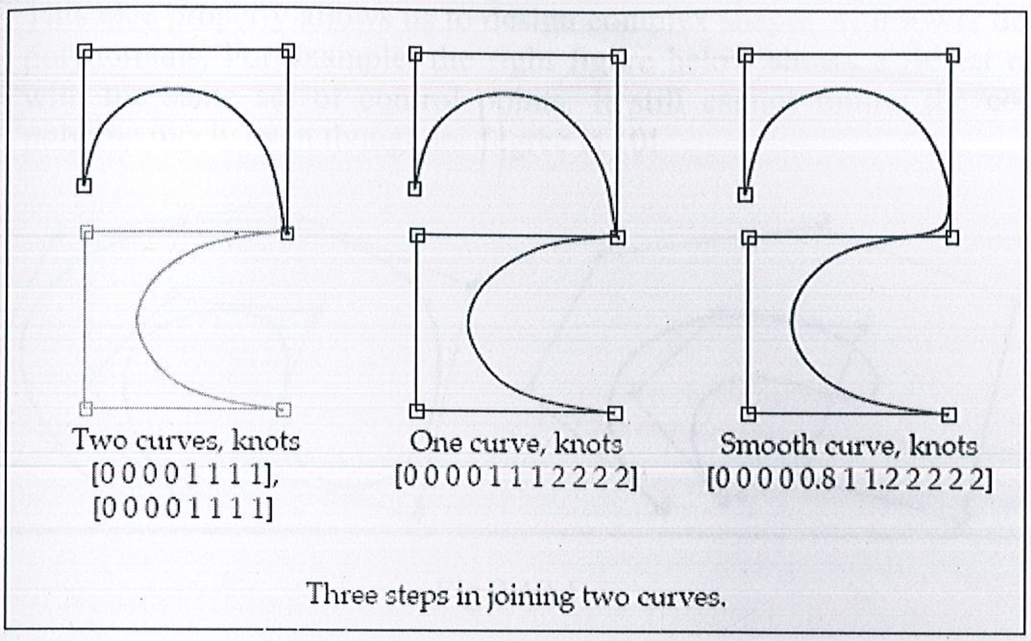


Fig 2.9

2.10 B-spline curves

Important properties

B-spline curves share many important properties with Bézier curves, because the former is a generalization of the later. Moreover, B-spline curves have more desired properties than Bézier curves. The list below shows some of the most important properties of B-spline curves.

In the following we shall assume a B-spline curve **C**(u) of degree p is defined by $n + 1$ control points and a knot vector $U = \{ u_0, u_1, ..., u_m \}$ with the first $p+1$ and last $p+1$ knots "clamped" (*i.e.*, $u_0 = u_1 = ... = u_p$ and $u_{m-p} = u_{m-p+1} = ... = u_m$).

1) **B-spline curve C(u) is a piecewise curve with each component a curve of degree p.**
As mentioned in previous page, **C**(u) can be viewed as the union of curve segments defined on each knot span. In the figure below, where $n = 10$, m

= 14 and p = 3, the first four knots and last four knots are clamped and the 7 internal knots are uniformly spaced. There are eight knot spans, each of which corresponds to a curve segment. In the left figure below, these knot points are shown as triangles.

This nice property allows us to design complex shapes with lower degree polynomials. For example, the right figure below shows a Bézier curve with the same set of control points. It still cannot follow the control polyline nicely even though its degree is 10!

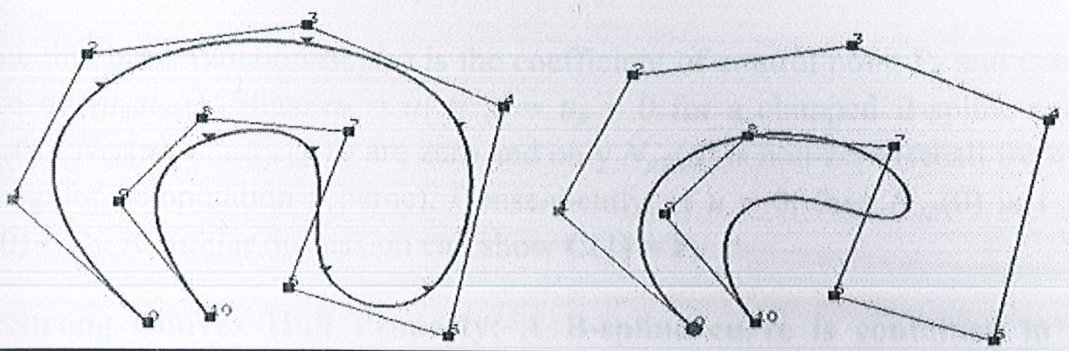


Fig 2.10.1

In general, the lower the degree, the closer a B-spline curve follows its control polyline. The following figures all use the same control polyline and knots are clamped and uniformly spaced. The first figure has degree 7, the middle one has degree 5 and the right figure has degree 3. Therefore, as the degree decreases, the generated B-spline curve moves closer to its control polyline

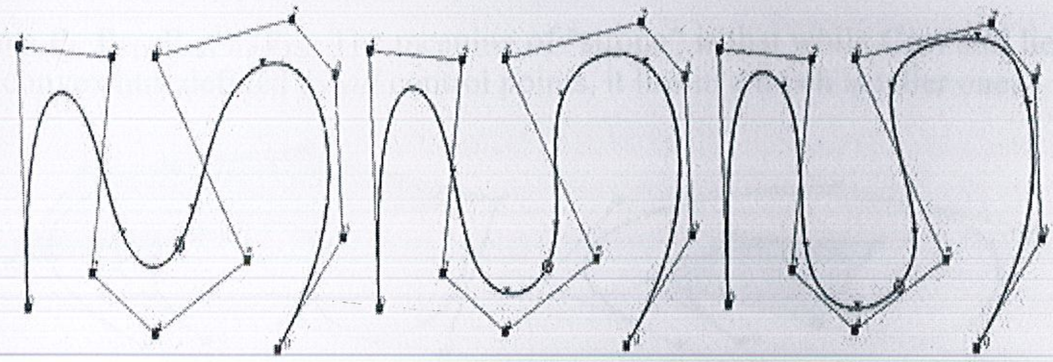


Fig 2.10.2

2) **Equality $m = n + p + 1$ must be satisfied.**

Since each control point needs a basis function and the number of basis functions satisfies $m = n + p + 1$.

3) **Clamped B-spline curve $\mathbf{C}(u)$ passes through the two end control points $\mathbf{P}_0$ and $\mathbf{P}_n$.**

Note that basis function $N_{0,p}(u)$ is the coefficient of control point $\mathbf{P}_0$ and is non-zero on $[u_0,u_{p+1})$. Since $u_0 = u_1 = \dots = u_p = 0$ for a clamped B-spline curve, $N_{0,0}(u)$, $N_{1,0}(u)$, ..., $N_{p-1,0}(u)$ are zero and only $N_{p,0}(u)$ is non-zero (recall from the triangular computation scheme). Consequently, if $u = 0$, then $N_{0,p}(0)$ is 1 and $\mathbf{C}(0) = \mathbf{P}_0$. A similar discussion can show $\mathbf{C}(1) = \mathbf{P}_n$

4) **Strong Convex Hull Property: A B-spline curve is contained in the convex hull of its control polyline. More specifically, if u is in knot span $[u_i,u_{i+1})$, then $\mathbf{C}(u)$ is in the convex hull of control points $\mathbf{P}_{i-p}$, $\mathbf{P}_{i-p+1}$, ..., $\mathbf{P}_i$.**

If u is in knot span $[u_i, u_{i+1})$, there are only p+1 basis functions (*i.e.*, $N_{i,p}(u)$, ... , $N_{i-p+1,p}(u)$, $N_{i-p,p}(u)$) non-zero on this knot span. Since $N_{k,p}(u)$ is the coefficient of control point $\mathbf{P}_k$, only p+1 control points $\mathbf{P}_i$, $\mathbf{P}_{i-1}$, $\mathbf{P}_{i-2}$, .., $\mathbf{P}_{i-p}$ have non-zero coefficients. Since on this knot span the basis functions are non-zero and sum to 1, their "weighted" average, $\mathbf{C}(u)$, must lie in the convex hull defined by control

points $\mathbf{P}_i$, $\mathbf{P}_{i-1}$, $\mathbf{P}_{i-2}$, .., $\mathbf{P}_{i-p}$. The meaning of "strong" is that while $\mathbf{C}(u)$ still lies in the convex hull defined by *all* control points, it lies in a much smaller one.

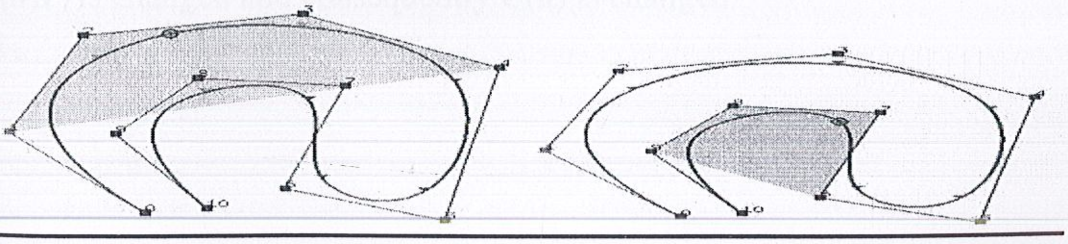


Fig 2.10.3

The above two B-spline curves have 11 control points (*i.e.*, $n = 10$), degree 3 (*i.e.*, $p=3$) and 15 knots ($m = 14$) with first four and last four knots clamped Therefore, the number of knot spans is equal to the number curve segments. The knot vector is

$u_0\ u_1\ u_2\ u_3\ u_4\ u_5\ u_6\ u_7\ u_8\ u_9\ u_{10}\ u_{11}\ u_{12}\ u_{13}\ u_{14}$ (0 0 0 0 0.12 0.25 0.37 0.5 0.62 0.75 0.87 1 1 1 1)

The left figure has u in knot span $[u_4, u_5) = [0.12,0.25)$ and the corresponding point (*i.e.* $\mathbf{C}(u)$) in the second curve segment. Therefore, there are $p+1 = 4$ basis functions non-zero on this knot span (*i.e.*, $N_{4,3}(u)$, $N_{3,3}(u)$, $N_{2,3}(u)$ and $N_{1,3}(u)$) and the corresponding control points are $\mathbf{P}_4$, $\mathbf{P}_3$, $\mathbf{P}_2$ and $\mathbf{P}_1$. The shaded area is the convex hull defined by these four points. It is clear that $\mathbf{C}(u)$ lies in this convex hull.

The B-spline curve in the right figure is defined the same way. However, u is in $[u_9, u_{10}) = [0.75,0.87)$ and the non-zero basis functions are $N_{9,3}(u)$, $N_{8,3}(u)$, $N_{7,3}(u)$ and $N_{6,3}(u)$. The corresponding control points are $\mathbf{P}_9$, $\mathbf{P}_8$, $\mathbf{P}_7$ and $\mathbf{P}_6$.

Consequently, as u moves from 0 to 1 and crosses a knot, a basis functions becomes zero and a new non-zero basis function becomes effective. As a result, one control point whose coefficient becomes zero will leave the the definition of the current convex hull and is replaced with a new control point whose coefficient becomes non-zero.

5) Local Modification Scheme: changing the position of control point $\mathbf{P}_i$ only affects the curve $\mathbf{C}(u)$ on interval $[u_i, u_{i+p+1})$.

This follows from another important property of B-spline basis functions. Recall that $N_{i,p}(u)$ is non-zero on interval $[u_i, u_{i+p+1})$. If u is not in this interval, $N_{i,p}(u)\mathbf{P}_i$ has no effect in computing $\mathbf{C}(u)$ since $N_{i,p}(u)$ is zero. On the other hand, if u is in the indicated interval, $N_{i,p}(u)$ is non-zero. If $\mathbf{P}_i$ changes its position, $N_{i,p}(u)\mathbf{P}_i$ is changed and consequently $\mathbf{C}(u)$ is changed.

.

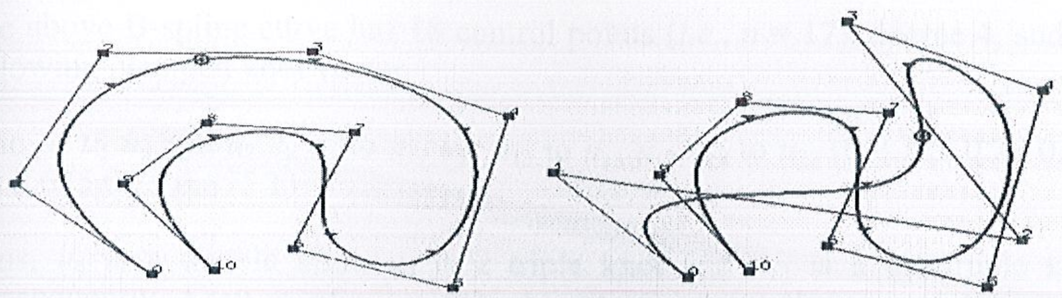


Fig 2.10.4

The above B-spline curves are defined with the same parameters as in the previous convex hull example. We intent to move control point $\mathbf{P}_2$. The coefficient of this control point is $N_{2,3}(u)$ and the interval on which this coefficient is non-zero is $[u_2, u_{2+3+1}) = [u_2, u_6) = [0,0.37)$. Since $u_2 = u_3 = 0$, only three segments that correspond to $[u_3, u_4)$ (the domain of the first curve segment), $[u_4, u_5)$ (the domain of the second curve segment) and $[u_5, u_6)$ (the domain of the third curve segment) will be affected. The right figure shows the result of moving $\mathbf{P}_2$ to the lower right corner. As you can see, only the first, second and third curve segments change their shapes and all remaining curve segments stay in their original,place without any change.

This local modification scheme is very important to curve design, because we can modify a curve locally without changing the shape in a global way. This will be elaborated on the moeing control point page. Moreover, if fine-tuning curve shape is required, one can insert more knots (and therefore more control points) so that the affected area could be restricted to a very narrow region. We shall talk about knot insertion later

6) $\mathbf{C}(u)$ is C^{p-k} continuous at a knot of multiplicity k.

If u is not a knot, $\mathbf{C}(u)$ is in the middle of a curve segment of degree p and is therefore infinitely differentiable. If u is a knot in the non-zero domain of $N_{i,p}(u)$, since the latter is only C^{p-k} continuous, so does $\mathbf{C}(u)$.

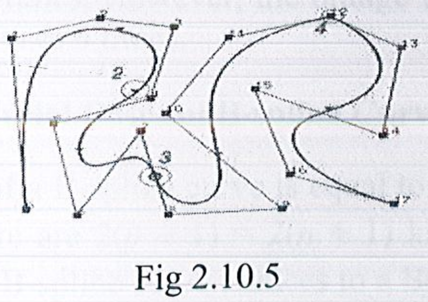


Fig 2.10.5

The above B-spline curve has 18 control points (*i.e.*, $n = 17$), degree 4, and the following clamped knot vector

u_0 to u_4 u_5 u_6 and u_7 u_8 u_9 to u_{11} u_{12} u_{13} to u_{16} u_{17} u_{18} to u_{22} (0 0.125 0.25 0.375 0.5 0.625 0.75 0.875 1)

Thus, u_6 is a double knot, u_9 is a triple knot and u_{13} is a quadruple knot. Consequently, $\mathbf{C}(u)$ is of C^4 continuous at any point that is not a knot, C^3 continuous at all simple knots, C^2 continuous at u_6, C^1 continuous at u_9, C^0 continuous at u_{13}.

All points on the curve that correspond to knots are marked with little triangles. Those corresponding to multiple knots are further marked with circles and their multiplicities. It is very difficult to visualize the difference between C^4, C^3 and even C^2 continuity. For the C^1 case, the corresponding point lies on a leg, while the C^0 case forces the curve to pass through a control point. We shall return to this issue later when discussing modifying knots.

7) **Variation Diminishing Property**.

The variation diminishing property also holds for B-spline curves. If the curve is in a plane (*resp.*, space), this means *no straight line (resp., plane) intersects a B-spline curve more times than it intersects the curve's control polyline*.

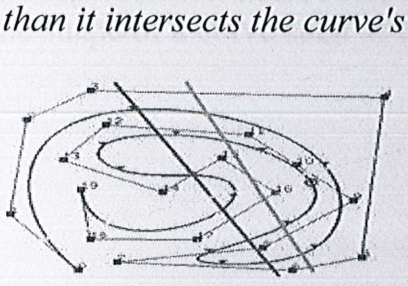


Fig 2.10.6

In the above figure, the blue line intersects both the control polyline and the B-spline curve 6 times, while the yellow line also intersects the control polyline and the B-spline curve 5 times. However, the orange line intersects the control polyline 6 times and the curve 4 times.

8) **Bézier Curves Are Special Cases of B-spline Curves.**

If $n = p$ (*i.e.*, the degree of a B-spline curve is equal to n, the number of control points minus 1), and there are $2(p + 1) = 2(n + 1)$ knots with $p + 1$ of them clamped at each end, this B-spline curve reduces to a Bézier curve.

9) **Affine Invariance.**

The affine invariance property also holds for B-spline curves. If an affine transformationis applied to a B-spline curve, the result can be constructed from the affine images of its control points. This is a nice property. When we want to apply a geometric or even affine transformation to a B-spline curve, this property states that we can apply the transformation to control points, which is quite easy, and once the transformed control points are obtained the transformed B-spline curve is the one defined by these new points. Therefore, we do not have to transform the curve.

2.11 The Advantage of Using B-spline Curves

B-spline curves require more information (*i.e.*, the degree of the curve and a knot vector) and a more complex theory than Bézier curves. But, it has more advantages to offset this shortcoming. First, a B-spline curve can be a Bézier curve Third, B-spline curves provide more control flexibility than Bézier curves can do. For example, the degree of a B-spline curve is separated from the number of control points. More precisely, we can use lower degree curves and still maintain a large number of control points. We can change the position of a control point without globally changing the shape of the whole curve (local modification property). Since B-spline curves satisfy the strong convex hull property, they have a finer shape control. Moreover, there are other techniques for designing and editing the shape of a curve such as changing knots.

Chapter III

B-SPLINE WAVELET MATRICES

This appendix presents the matrices required to make use of endpoint-interpolating B-spline wavelets of low degree. The Matlab code used to generate these matrices appears in Appendix C. These concrete examples should serve to elucidate the ideas presented in Section 7.3.2. In order to emphasize the sparse structure of the matrices, zeros have been omitted. Diagonal dots indicate that the previous column is to be repeated the appropriate number of times,shifted down by two rows for each column. The *P* matrices have entries relating unnormalized B-spline scaling functions, while the *Q* matrices have entries defining normalized, minimally supported wavelets. Columns of the *Q* matrices that are not represented exactly with integers are given to six decimal places.

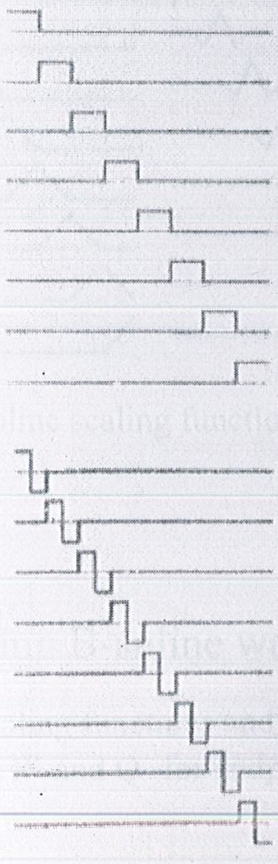


Fig.3.1 The B-spline constant B spline scaling functions and wavelet for j=3

3.1 Haar Wavelets

The B-spline wavelet basis of degree 0 is simply the haar basis described in chapter 2.Some examples of the box scaling functions and Haar wavelets are described in figure 3.1.The synthesis matrices P' and Q' are given below

$$P^j = \begin{bmatrix} 1 & & & \\ 1 & & & \\ & 1 & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \\ & & & 1 \end{bmatrix} \qquad Q^j = \sqrt{\frac{2^j}{2}} \begin{bmatrix} 1 & & & \\ -1 & & & \\ & 1 & & \\ & -1 & & \\ & & \ddots & \\ & & & 1 \\ & & & -1 \end{bmatrix}$$

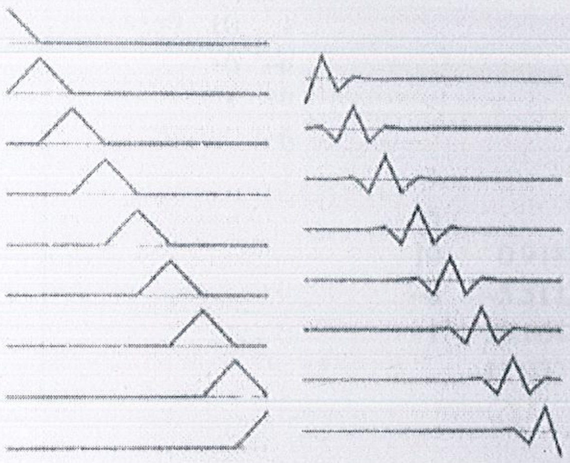


Fig.3.2 The linesr B-spline scaling functions and wavelets for j=3

3.2 Endpoint-interpolating B-spline wavelets

Fig.3.2 shows a few typical scaling functions and wavelets for linear B-splines.The synthesis matrices P' and Q' for endpoint- interpolating linear B-spline wavelets are given below.

$$P^1 = \frac{1}{2}\begin{bmatrix} 2 & \\ 1 & 1 \\ & 2 \end{bmatrix} \quad Q^1 = \sqrt{3}\begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$$

$$P^2 = \frac{1}{2}\begin{bmatrix} 2 & & \\ 1 & 1 & \\ & 2 & \\ & 1 & 1 \\ & & 2 \end{bmatrix} \quad Q^2 = \sqrt{\frac{3}{64}}\begin{bmatrix} -12 & \\ 11 & 1 \\ -6 & -6 \\ 1 & 11 \\ & -12 \end{bmatrix}$$

$$= \sqrt{\frac{2^j}{72}}\begin{bmatrix} -11.022704 & & & & \\ 10.104145 & 1 & & & \\ -5.511352 & -6 & & & \\ 0.918559 & 10 & 1 & & \\ & -6 & -6 & & \\ & 1 & 10 & & \\ & & -6 & & \\ & & 1 & \ddots & \\ & & & 1 & \\ & & & -6 & \\ & & & 10 & 0.918559 \\ & & & -6 & -5.511352 \\ & & & 1 & 10.104145 \\ & & & & -11.022704 \end{bmatrix}$$

3.3 Endpoint-interpolating quadratic B-spline wavelets

Figure 3.3 shows some of the quadratic B-spline scaling functions and wavelets. The synthesis matrices P^j and Q^j in the quadratic case are given below.

$$P^1 = \frac{1}{2}\begin{bmatrix} 2 & \\ 1 & 1 \\ & 1 & 1 \\ & & 2 \end{bmatrix} \quad Q^1 = \sqrt{\frac{5}{4}}\begin{bmatrix} -2 \\ 3 \\ -3 \\ 2 \end{bmatrix}$$

$$P^2 = \frac{1}{4}\begin{bmatrix} 4 & & & \\ 2 & 2 & & \\ & 3 & 1 & \\ & 1 & 3 & \\ & & 2 & 2 \\ & & & 4 \end{bmatrix} \qquad Q^2 = \sqrt{\frac{3}{4936}}\begin{bmatrix} -144 & \\ 177 & 21 \\ -109 & -53 \\ 53 & 109 \\ -21 & -177 \\ & 144 \end{bmatrix}$$

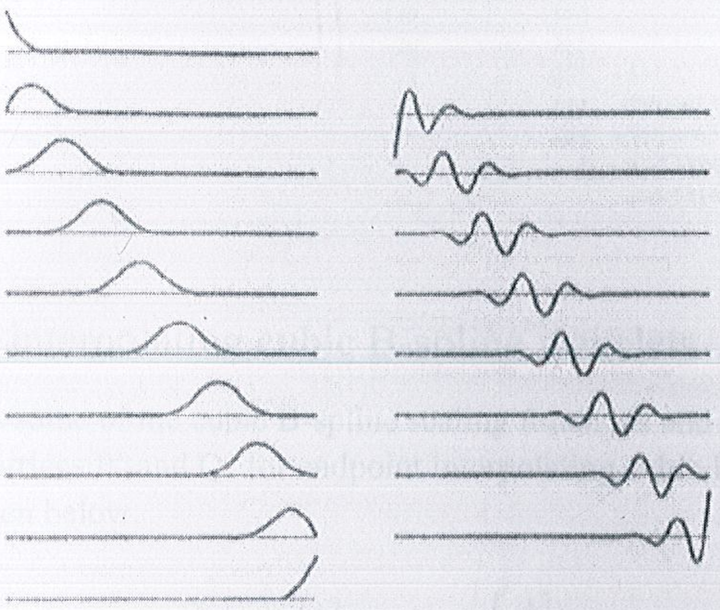


Fig.3.3

The quadratic B-spline scaling functions and wavelets for j=3

$$P^{j\geq 3} = \frac{1}{4}\begin{bmatrix} 4 & & & & & & \\ 2 & 2 & & & & & \\ & 3 & 1 & & & & \\ & 1 & 3 & & & & \\ & & 3 & 1 & & & \\ & & 1 & 3 & & & \\ & & & 3 & & & \\ & & & 1 & 1 & & \\ & & & & 3 & & \\ & & & & 3 & 1 & \\ & & & & 1 & 3 & \\ & & & & & 2 & 2 \\ & & & & & & 4 \end{bmatrix} \qquad Q^3 = \sqrt{\frac{1}{713568}}\begin{bmatrix} -4283.828550 & & & \\ 5208.746077 & 780 & & \\ -3099,909150 & -1949 & -11 & \\ 1300,002166 & 3481 & 319 & \\ -253.384964 & -3362 & -1618 & -8.737413 \\ 8.737413 & 1618 & 3362 & 253.384964 \\ & -319 & -3481 & -1300,002166 \\ & 11 & 1949 & 3099.909150 \\ & & -780 & -5208.746077 \\ & & & 4283.828550 \end{bmatrix}$$

$$Q^{j\geq 4}=\sqrt{\frac{3\cdot 2^j}{136088}}\begin{bmatrix} -381.872771 & & & & & & \\ 464.322574 & 69.531439 & & & & & \\ -276.334798 & -173.739454 & -1 & & & & \\ 115.885924 & 310.306330 & 29 & & & & \\ -22.587463 & -299.698329 & -147 & -1 & & & \\ 0.778878 & 144.233164 & 303 & 29 & & & \\ & -28.436576 & -303 & -147 & & & \\ & 0.980572 & 147 & 303 & -1 & & \\ & & -29 & -303 & 29 & & \\ & & 1 & 147 & -147 & -0.980572 & \\ & & & -29 & 303 & 28.436576 & \\ & & & 1 & -303 & -144.233164 & -0.77878 \\ & & & & 147 & 299.698329 & 22.587463 \\ & & & & -29 & -310.306330 & -115.885924 \\ & & & & 1 & 173.739454 & 276.334798 \\ & & & & & -69.531439 & -464.322574 \\ & & & & & & 381.872771 \end{bmatrix}$$

3.4 Endpoint interpolating cubic B-spline wavelets

Figure 3.4 shows some of the cubic B-spline scaling functions and wavelets. The synthesis matrices P' and Q' for endpoint interpolating cubic B-spline wavelets are given below.

$$P^1=\frac{1}{2}\begin{bmatrix} 2 & & & \\ 1 & 1 & & \\ & 1 & 1 & \\ & & 1 & 1 \\ & & & 2 \end{bmatrix} \qquad Q^1=\sqrt{7}\begin{bmatrix} 1 \\ -2 \\ 3 \\ -2 \\ 1 \end{bmatrix}$$

$$P^2=\frac{1}{16}\begin{bmatrix} 16 & & & & \\ 8 & 8 & & & \\ & 12 & 4 & & \\ & 3 & 10 & 3 & \\ & & 4 & 12 & \\ & & & 8 & 8 \\ & & & & 16 \end{bmatrix} \qquad Q^2=\sqrt{\frac{315}{31196288}}\begin{bmatrix} 1368 & \\ -2064 & -240 \\ 1793 & 691 \\ -1053 & -1053 \\ 691 & 1793 \\ -240 & -2064 \\ & 1368 \end{bmatrix}$$

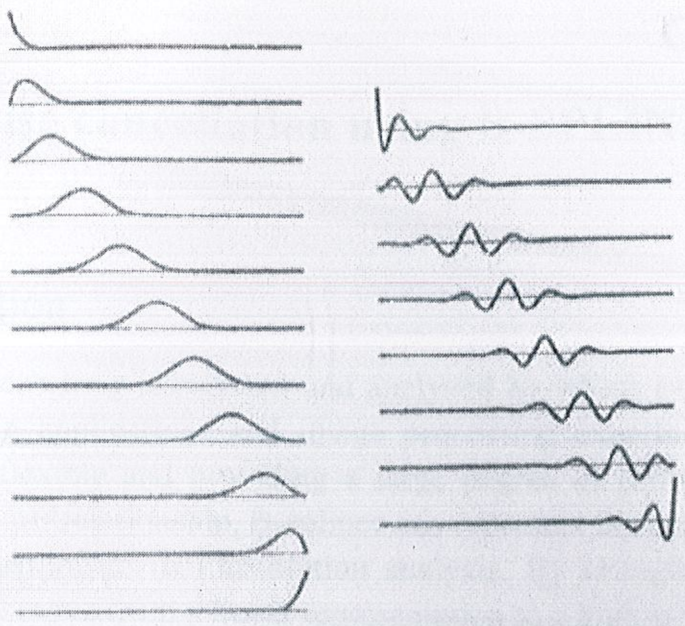


Fig.3.4

The cubic B-spline scaling functions and wavelets for j=3

$$P^{(3)} = \frac{1}{16}\begin{bmatrix}
16 & & & & & & & & \\
8 & 8 & & & & & & & \\
 & 12 & 4 & & & & & & \\
 & 3 & 11 & 2 & & & & & \\
 & & 8 & 8 & & & & & \\
 & & 2 & 12 & 2 & & & & \\
 & & & 8 & 8 & & & & \\
 & & & 2 & 12 & & & & \\
 & & & & 8 & & & & \\
 & & & & 2 & 2 & & & \\
 & & & & & 8 & & & \\
 & & & & & 12 & 2 & & \\
 & & & & & 8 & 8 & & \\
 & & & & & 2 & 11 & 3 & \\
 & & & & & & 4 & 12 & \\
 & & & & & & & 8 & 8 \\
 & & & & & & & & 16
\end{bmatrix}$$

$$Q^{3} = \begin{bmatrix}
6.311454 & & & \\
-9.189342 & -1.543996 & & \\
7.334627 & 4.226722 & 0.087556 & \\
-3.514553 & -5.585477 & -0.473604 & -0.000155 \\
1.271268 & 6.059557 & 1.903267 & 0.019190 \\
-0.259914 & -4.367454 & -4.367454 & -0.259914 \\
0.019190 & 1.903267 & 6.059557 & 1.271268 \\
-0.000155 & -0.473604 & -5.585477 & -3.514553 \\
 & 0.087556 & 4.226722 & 7.334627 \\
 & & -1.543996 & -9.189342 \\
 & & & 6.311454
\end{bmatrix}$$

Chapter IV

Image Noise cancellation using B-SPLINE wavelets

4.1) Introduction

B-splines have been long introduced and analyzed by which caught interest of many engineering applications and image processing experts . Due to their merits of being flexible and providing a large degree of differentiability and cost/quality trade off relationship, B-splines can represent the next generation of wavelets for signal/image multiresolution analysis. By changing the B-spline function order we move from a linear representation to a high-order bandlimited representation. The wavelet theory for image noise cancellation introduced was our main reason for using B-splines for non-band limited signals. B-splines were utilized for image noise cancellation using polynomial equations.A new set of fractional B-splines were introduced for different degrees. We note here that in spite of the long history of B-splines in engineering applications, they have not been fully analyzed from a image noise cancellation (analysis/synthesis) or perfect reconstruction (PR) point of view. B-splines have been utilized to build wavelets from any two sequences as long as they are non perpendicular.Another set of wavelets has been constructed using shifted B-splines.B-splines were used for signal/image zooming,interpolation and image noise cancellation .

The primary objective of our work is to design and develop B-spline based multiscale representations of signals/images that are based on a PR frame work, like the wavelet analysis introduced. Our proposed technique for calculating the B-spline basis is a straightforward approach that is based on matrix multiplications (Toeplitz); it avoids inverse filtering and unstable filter calculations that are first proposed. It also calculates the B-spline basis for a batch of input image samples in a concurrent process without any extra basis or overlaps.

We note here that in spite of the long history of the use of B-splines in signal representation, classification, and interpolation applications ,they have not been fully investigated in a image noise cancellation applications.We propose a method for improved noise cancellation method through some B-spline basis. Due to the selective nature of the B-spline semi-orthogonal decomposition, a distinctive type of data correlation (that is usually around edges) is captured and removed from this pre-processing process.

This would further reduce the data correlation and would achieve better cancellation of noise in the image as evident by our adopted correlation measure with and without the proposed technique and by the enhanced PSNR/bpp curves.We tested our generated B-spline wavelets in image noise cancellation and compared our approach with the well-known 9/7 Antonini et al. filter using the SURE technique.

4.2) Background

4.2.1) Mathematical background

The *m*th order B-spline function *Bm(t),* satisfies the following basic properties:

1. *Bm(t)* is of finite support and equals zeros at $t \leq 0$ and $t \geq m$. Between the knots $t = 1, 2, \ldots, m-1$, it is represented by a polynomials of order $(m-1)$ in t. It satisfies the recurrence relation:

$$B_m(t) = \frac{t}{m-1} B_{m-1}(t) + \frac{m-t}{m-1} B_{m-1}(t-1)$$
$$\frac{\partial B_m(t)}{\partial t} = B_{m-1}(t) - B_{m-1}(t-1) \tag{1}$$

2. The Fourier transform *Bm(ω)* is given by

$$B_m(\omega) = e^{-j\frac{m\omega}{2}} \left(\frac{\sin\left(\frac{\omega}{2}\right)}{\left(\frac{\omega}{2}\right)} \right)^m \Rightarrow B_m(t)$$
$$= B_1(t)^* B_1(t)^* \ldots^* B_1(t)$$
$$\longleftarrow - - - \quad m \text{ times} \quad - \longrightarrow \tag{2}$$

3. *Bm(t)* is symmetric around *m/2* ;

$$\text{i.e. } B_m\left(\frac{m}{2}+t\right) = B_m\left(\frac{m}{2}-t\right)$$

The discrete B-spline basis *Bm(n)* is obtained by sampling *Bm(t)* at its knots *t* = 0, 1, . . . ,*m.* The *L*th interpolated discrete B-spline basis *BLm(n)* is obtained by sub-sampling the sampling

interval into *L* equal intervals i.e

$$B^L_m(t) = B^1_m\left(\frac{t}{L}\right).$$

4.2.2) Signal interpolation using B-spline polynomials.

For a discrete signal *g(k)* of length *N,*that

is interpolated using an *m*th order discrete B-spline *Bm(n),*we would have

$$g(k) = \sum_{l=-\infty}^{\infty} c(l)B_m(k-l) = \sum_{l=-m+1}^{N-2} c(l)B_m(k-l) \quad (3)$$

where *c(l)_s* are the B-spline decomposition coefficients for the *g(k)* signal. The limits in are due to the nonzero values of *Bm(n),* This means that a batch of length *N* can be expanded by B-spline polynomial having an utmost *N* +*m*−2 coefficients.To compute the *c*_ swehave toevaluate the*C(z),*as follows:

$$G(z) = C(z)B_m(z) \Rightarrow C(z) = \frac{G(z)}{\sum_{k=1}^{m-1} B_m(k)z^{-k}} \quad (4)$$

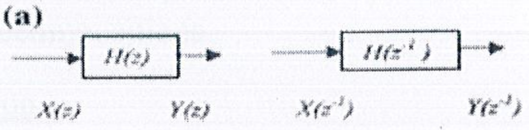
Hence, the interpolating coefficients *c(l)* can be viewed as the output of the IIR filter1 *Bm(z),* when driven by the sequence *g(k).* However, online computations of the *c*_s are not possible,

as *Bm(z)* has roots outside the unit circle u.c., as well as inside the u.c. (as a result of mirror symmetric around *m*2*).*Solution of this problem can be achieved in non-real time as follows: consider the digital system of Fig.1a, as *Y (z*−1*)* = *H(z*−1*)X(z*−1*),* and if *H(z)* has all its poles outside the u.c,then *Y (z*−1*)* is recursively computable. To evaluate *y(n)*, we apply the following flipping technique in computing *y(n)* :

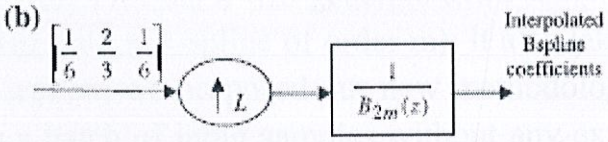
1. Flip the batch of the sequence *x(n)*, to get *x f (n)*.

2. Filter *x f (n)* by the now-stable system *H(z–1)* to yield the output *y f (n)*.

3. Flip *y f (n)* back to get *y(n)*.



flipping technique of non-real time computation



[..refer research paper (21)]

Principle of computing $\dot{B}_m(t), m = 4$

The idea is now exploited int the computations of c(l) as follows

1. Factorize $B_m(z)$ as $B_m(z) = z^{-1} \prod_{k=1}^{\frac{m-2}{2}} (1 - z_k z^{-1}) \left(1 - \frac{z^{-1}}{z_k}\right) = H_s(z)\, H_{us}(z), |z_k| < 1$. H_s, H_{us} are the stable and unstable parts of $B^m(z)$, respectively

[..refer research paper (21)]

2.) Filter the sequence *g(k)* by all-pole filter *Hs(z)*, to get *y(k)*, *k* = 0, 1, . . . , *N* – 1

3.) As *C(z) =Y (z)/Hus(z)* , apply the proposed flipping technique,to get *c(l)*. Select *(N + m –* 2*)* consecutive coefficients of this output. Figure 1b, shows the B-spline coefficients for a cubic B-spline with an interpolation factor L. B-splines are known for their energy compaction/

concentration feature: as will be shown later, this feature was the main reason for its compression enhancement performance. Figure 2 compares the mean Eigen values of DCT coefficients (which is best known for energy concentration) with mean B-spline coefficients of different orders for the 8 × 8 block of the 256 × 256 Cameraman image. Order 7 is the highest Eigen value on the horizontal axis. As shown in the figure for the first few Eigen values, the energy concentration for B-splines is higher than DCT, but after a certain order, the B-spline energy concentration feature will be less than DCT. This justifies the compression performance improvement for B-splines for low bit rates over regular orthonormal decompositions.

4.3) B-spline calculation

As shown in the previous sections, a discrete signal g(k) of length N, that is interpolated using an mth order discrete B-spline Bm(n),we would have N+m−2 resulting samples, which means it will generate extra samples (m samples for N samples interpolated with a B-spline of order m). It also takes one input sample at a time. In this section, we proposed our new methodology of B-spline basis calculations using a batch of input samples without any extra samples. The c's can be calculated if the limits of Eq. 3, are changed to..

$$g(k) = \sum_{l=-\frac{m}{2}}^{N-\frac{m}{2}-1} c(l)\, B_m(k-l) \tag{5a}$$

$$\mathrm{B} = \begin{bmatrix} B_m\left(\frac{m}{2}\right) & B_m\left(\frac{m}{2}-1\right) & \ldots\ldots & B_m(1) & 0 & \ldots & 0 \\ B_m\left(\frac{m}{2}+1\right) & B_m\left(\frac{m}{2}\right) & B_m\left(\frac{m}{2}-1\right)\ldots\ldots & & B_m(1) & 0 & \\ & B_m\left(\frac{m}{2}+1\right) & B_m\left(\frac{m}{2}\right) & & & & \\ & & & \ldots & & & \\ 0 & & & & \ldots & & \\ 0 & & & & & \ldots & \ldots\ldots \\ 0 & 0 & & & & B_m\left(\frac{m}{2}\right) & B_m\left(\frac{m}{2}-1\right) \\ 0 & 0 & & & \ldots\ldots & B_m\left(\frac{m}{2}+1\right) & B_m\left(\frac{m}{2}\right) \end{bmatrix},$$

$$C = \begin{bmatrix} c_{-\frac{m}{2}} \\ c_{-\frac{m}{2}+1} \\ \ldots \\ c_{N-\frac{m}{2}-1} \end{bmatrix} \tag{5b}$$

[..refer research paper (21)]

Then only N coefficients are needed. The c's are solution to the linear system.

$$BC = g, \qquad g = [\,g(0)\ g(1)\ \dots\ g(N-1)\,]'$$

The solution of this system is much simpler. In case of cubic B-spline, the matrix B is reduced to Tri-diagonal (Toeplitz) matrix, whose solution is straightforward and can be implemented online. At this point we can show that, the interpolation of L −1 new points between every two knots is given by,

$$g\left(\frac{k}{L}\right) = \sum_{r=-\frac{m}{2}}^{N-\frac{m}{2}-1} C_r\, B_m\left(\frac{k}{L} - r\right) \tag{6}$$

$$\hat{g}(j) = \sum_{r=-\frac{m}{2}}^{N-\frac{m}{2}-1} C_r\, B_m^{(L)}(j - rL),$$

where *j* is the new interpolation of the input *k* samples.As far as signal compression is concerned, if a batch of *N* samples of *g(n)* is decimated by *L*, where *N* is multiple of *L*, then the optimum B-spline coefficients ˆ*c*, must be chosen
to minimize the norm of the error signal *e(n)* = *g(n)*-where *j* is the new interpolation of the input *k* samples.As far as signal compression is concerned, if a batch of *N* samples of *g(n)* is decimated by *L*, where *N* is multiple of *L*,then the optimum B-spline coefficients ˆ *c*, must be chosen to minimize the norm of the error signal *e(n)* = *g(n)* −

$$\hat{g}(n);\ \hat{g}(n) = \sum_{r=-\frac{m}{2}}^{\frac{N}{L}-\frac{m}{2}-1} \hat{c}(r) B_m^{(L)}(n - rL) n = 0, 1, \dots,$$
$$N - 1.$$

[..refer research paper (21)]

This means that the optimum c's are the solution to the following linear equation:

$$\sum_{k=-\frac{m}{2}}^{\frac{N}{L}-\frac{m}{2}-1} \left(\sum_{n=0}^{N-1} B_m^{(L)}(n-kL)\, B_m^{(L)}(n-rL) \right) \hat{c}(k) = \sum_{n=0}^{N-1} g(n)\, B_m^{(L)}(n-rL)$$

[..refer research paper (21)]

where *r = -m/2,-m/2 + 1,……..N/L - m/2 – 1.* As B-spline interpolated basis is known $B_m^{(l)}$(n),the solution can be achieved. Both of the interpolation and decimation techniques can be applied on images by working on rows and then columns, or vice versa.

4.3.1) Derivation of B-spline wavelet bases

4.3.1) B-spline 2-scale relation, dual and wavelet functions

The 2-scale property of B_m(t),is defined as:

$$B_m(t) = \sum_k p_k B_m(2t-k) \tag{7}$$

Taking the Fourier transform FT, one can show that,

$$\begin{aligned} B_m(\omega) &= \frac{1}{2} \sum_k p_k\, e^{-j\frac{k\omega}{2}} B_m\left(\frac{\omega}{2}\right) \\ P(z) &\equiv \frac{1}{2} \sum_k p_k\, e^{-j\frac{k\omega}{2}} = \frac{B_m(\omega)}{B_m\left(\frac{\omega}{2}\right)}. \\ &= \frac{1}{2^m} \sum_{k=0}^{m} \binom{m}{k} z^{-k} = \left(\frac{1+z^{-1}}{2}\right)^m, \quad z = e^{j\frac{\omega}{2}} \end{aligned} \tag{8}$$

[..refer research paper (21)]

where p_k's are the scaling function (low pass filter) of the B-spline wavelet subband coding system.

The dual B-spline function $\hat{B}_m(t)$, is defined to satisfy,

$$\int B_m(t)\, \hat{B}_m(t-k)\, \mathrm{d}t = \delta(k) \tag{9}$$

This condition, together with the properties of the B-spline polynomials would yield

$$B_m(t) = \sum_{k=-m+1}^{m-1} \alpha_k \hat{B}_m(t-k), \tag{10}$$

$$\alpha_k = \int B_m(t)\, B_m(t-k)\, \mathrm{d}t = B_{2m}(m+k)$$

So, the Fourier transform of $\hat{B}_m(t)$, is given by

$$\hat{B}_m(\omega) = \frac{B_m(\omega)}{\sum_{k=-m+1}^{m-1} \alpha_k \mathrm{e}^{-jk\omega}} = \frac{B_m(\omega)}{N(z^2)} \tag{11}$$

[..refer research paper (21)]

Where N(z²) is the summation of squared absolute shifted B-spline basis. Figure 4b shows how to compute cubic dual B-spline $\hat{B}_m(t)$.for a resolution of L. Thus, Bm(t) belongs to the family of semi-orthogonal wavelets. This means that a function f (t) can be decomposed by using either Bm(t), or its dual $\hat{B}_m(t)$. ; i.e.,

$$f(t) = \sum_k c_k B_m(t-k) = \sum_k \hat{c}_k \hat{B}_m(t-k)$$

$$\Rightarrow \hat{c}_k - \sum_{r=-m+1}^{m-1} \alpha_r c_{k-r} = \alpha^* c. \tag{12}$$

[..refer research paper (21)]

where α's are factors.

The *m*th order B-spline wavelet ψ*m*(t), is constructed to satisfy the relation.

$$\int \psi_m(t-k) B_m(t) \mathrm{d}t = 0 \tag{13}$$

for all integer shifts of k. Thus, one can express ψ (t) as,

$$\psi_m(t) = \sum_k q_k\, B_m(2t - k)$$

$$Q(z) = \frac{\Psi_m(\omega)}{B_m\left(\frac{\omega}{2}\right)}, \quad Q(z) = \frac{1}{2}\sum_k q_k\, z^{-k} \tag{14}$$

[..refer research paper (21)]

where qk 's are the wavelet function basis (high pass filter) of the B-spline wavelet subband coding system. Moreover, due to the orthogonality between the scaling andwavelet B-spline functions, we have:

$$\int \Psi_m(\omega)\ \overline{B_m(\omega)}\ e^{-jk\omega}d\omega = 0,$$

where $\overline{B_m(\omega)}$ is the complement of $B_m(\omega)$

$$\int Q(z)P(z^{-1})\left|B_m(\frac{\omega}{2})\right|^2 e^{-jk\omega}d\omega = 0,$$

$$Q(z) = \frac{1}{2}\sum_k q_k z^{-k}, \quad z = e^{j\frac{\omega}{2}}.$$

$$\therefore \sum_n \int_0^{4\pi} Q(z)P(z^{-1})\left|B_m(\frac{\omega}{2}+2\pi\, n)\right|^2 e^{-jk\omega}d\omega = 0 \tag{15}$$

[..refer research paper (21)]

However, as in the classical wavelet theory [11], Parseval's energy theory yields :

$$\int B_m(t)B_m(t-k)\mathrm{d}t = \frac{1}{2\pi}\int_{-\infty}^{\infty} |B_m(\omega)|^2 \mathrm{e}^{-jk\omega}\mathrm{d}\omega$$
$$= \frac{1}{2\pi}\int_{0}^{2\pi} \sum_n |B_m(\omega+2n\pi)|^2 \mathrm{e}^{-jk\omega}\mathrm{d}\omega$$
$$\equiv B_{2m}(m-k) \quad (16)$$

Thus, we have the Fourier series expansion

$$\sum_n |B_m(\omega+2n\pi)|^2 \sum_{k=-m+1}^{m-1} B_{2m}(m-k)\,\mathrm{e}^{-jk\omega} \equiv N(z^2), \quad (17)$$

However, since

$$\begin{aligned} N(z^2) = {} & \cdots + |B_m(\omega+8\pi)|^2 + |B_m(\omega+6\pi)|^2 \\ & + |B_m(\omega+4\pi)|^2 + |B_m(\omega+2\pi)|^2 + |B_m(\omega)|^2 \\ & + |B_m(\omega-2\pi)|^2 \\ & + |B_m(\omega-4\pi)|^2 + |B_m(\omega-6\pi)|^2 \\ & + |B_m(\omega-8\pi)|^2 + \cdots \\ = {} & \cdots |B_m(\omega+8\pi)|^2 + |B_m(\omega+4\pi)|^2 + |B_m(\omega)|^2 \\ & + |B_m(\omega-4\pi)|^2 + |B_m(\omega-8\pi)|^2 \\ & + \cdots |B_m((\omega+2\pi)+4\pi)|^2 + |B_m(\omega+2\pi)|^2 \\ & + |B_m(\omega-2\pi)|^2 + |B_m(\omega-2\pi-4\pi)|^2 + \cdots . \end{aligned} \quad (18a)$$

$$\text{i.e.}\quad N(z^2) = P(z)P(z^{-1})\left\{\sum_n \left|N_m(\frac{\omega}{2}+2n\pi)\right|^2\right\} + P(-z)P(-z^{-1})\left\{\sum_n \left|N_m(\frac{\omega+2\pi}{2}+2n\pi)\right|^2\right\}$$

$$\text{i.e.}\quad N(z^2) = P(z)P(z^{-1})N(z) + P(-z)P(-z^{-1})N(-z) \quad (18b)$$

[..refer research paper (21)]

Substituting Eq. (18) back in Eq. (15), and in order for $Q(z)$ to be a causal FIR, we have

$$
\begin{aligned}
& Q(z)\,P(z^{-1})N(z) + Q(-z)\,P(-z^{-1})N(-z) = 0 \\
\text{i.e. } & Q(z) = z^{-(2m-1)}\,P(-z^{-1})N(-z) \\
& = (-z)^{-(m-1)}\,\tilde{P}(-z)\,N(-z), \quad \tilde{P}(z) = z^{-m}P(z^{-1}) \\
& = -E(-z)\tilde{P}(-z); \quad E(z) \\
& = z^{-(m-1)}N(z) = \sum_{k=1}^{2m-1} B_{2m}(k)z^{-(k-1)} \qquad (19)
\end{aligned}
$$

[..refer research paper (21)]

Figure 4 shows the generated *P(z),Q(z)*, and *E(z)*, for a bilinear, cubic, quadratic, and order eight B-spline based representations. As shown in the figure, as the B-spline order increases, the overlap between *P(z)* and *Q(z)* is reduced,which makes this B-spline based decomposition more orthogonaland energy concentrated:

4.4) Two-scale relations of dual scaling and wavelet

B-splines

As the 2-scale B-spline scaling relation of Eq. (7), the 2-scale dual scaling relation is defined as

$$\hat{B}_m(t) = \sum_k a_k \hat{B}_m(2t-k) \qquad (20)$$

Taking the FT of Eq. (15), and using Eq. (11), one can show that

$$\hat{B}_m(\omega) = A(z)\,\frac{B_m(\frac{\omega}{2})}{N(z)} \equiv \frac{B_m(\omega)}{N(z^2)}; \quad A(z) = \frac{1}{2}\sum_k a_k z^{-k} \tag{21}$$

$$\text{i.e. } A(z) = \frac{N(z)}{N(z^2)}\,P(z) = z\,\frac{E(z)}{E(z^2)}\,\tilde{P}(z)$$

Similarly, the dual wavelet $\hat{\psi}_m(t)$ is orthogonal to the dual B-spline wavelet $\hat{B}_m(t)$, and is related by

$$\int \hat{\psi}_m(t)\,\hat{B}_m(t-k)\,\mathrm{d}t = 0 \Rightarrow \hat{\Psi}_m(t)$$

$$= \sum_k b_k\,\hat{B}_m(2t-k) \quad \text{for all integer } k \tag{22}$$

Taking the FT of Eq. (22), and using Eqs. (11, 17), yields

$$\hat{\Psi}_m(\omega) = \frac{1}{2}\sum_k b_k\,\mathrm{e}^{-j\frac{k\omega}{2}}\,\hat{B}_m\left(\frac{\omega}{2}\right)$$

$$\therefore \quad \frac{\Psi_m(\omega)}{\sum_n |\Psi_m(\omega+2n\pi)|^2} = B(z)\,\frac{B_m(\frac{\omega}{2})}{N(z)} \tag{23}$$

As in Eq. (13), one can show that

$$\begin{aligned}\sum_n |\Psi_m(\omega+2n\pi)|^2 &= Q(z)Q(z^{-1})N(z)\\ &\quad + Q(-z)Q(-z^{-1})N(-z)\\ &= P(-z)P(-z^{-1})N^2(-z)N(z)\\ &\quad + P(z)P(z^{-1})N^2(z)N(-z)\\ &= N(z)\,N(-z)\,N(z^2)\end{aligned} \tag{24}$$

$$\begin{aligned}\therefore\quad B(z) &= \frac{Q(z)}{N(z^2)N(-z)}\\ &= -(-z)^{-(m-1)}\frac{\tilde{P}(-z)}{N(z^2)}\\ &= \text{i.e.}\quad B(z^{-1}) = z\frac{P(-z)}{E(z^2)}\end{aligned}$$

Note that unlike $P(z)$ and $Q(z)$, each of $A(z)$ and $B(z)$ is an IIR non-causal symmetric function.

3.4 The perfect reconstruction, decomposition, and reconstruction relations

[..refer research paper (21)]

These equations describe PR system. To find its system simulation, multiplying both sides of the first equation by B_m(w/2) $e^{-j\,kw/2}$,yields:

$$\begin{aligned}B_m\left(\frac{\omega}{2}\right)e^{-j\frac{k\omega}{2}} &= A(z^{-1})\,P(z)B_m\left(\frac{\omega}{2}\right)e^{-j\frac{k\omega}{2}}\\ &\quad + B(z^{-1})Q(z)B_m\left(\frac{\omega}{2}\right)e^{-j\frac{k\omega}{2}}\\ &= \frac{1}{2}\sum_r a_r B_m(\omega)\,e^{-j\frac{(k-r)\omega}{2}}\end{aligned}$$

$$+\frac{1}{2}\sum_r b_r\,\psi_m(\omega)\,e^{-j\frac{(k-r)\omega}{2}} \text{ i.e.,} \tag{26}$$

$$B_m(2t-k)=\frac{1}{2}\sum_r a_r B_m\left(t-\frac{k-r}{2}\right)$$

$$+\frac{1}{2}\sum_r b_r\,\psi_m\left(t-\frac{k-r}{2}\right)$$

$$\text{i.e.,}\quad =\frac{1}{2}\sum_l a_{k-2l} B_m(t-l)$$

$$+\frac{1}{2}\sum_l b_{k-2l}\psi_m(t-l)$$

Similarly, multiplying the 2nd equation by $\hat{B}_m\left(\frac{\omega}{2}\right)e^{-j\frac{k\omega}{2}}$ yields

$$\hat{B}_m(2t-k)=\frac{1}{2}\sum_l p_{k-2l}\,\hat{B}_m(t-l)$$

$$+\frac{1}{2}\sum_l q_{k-2l}\,\hat{\psi}_m(t-l) \tag{27}$$

[..refer research paper (21)]

To relate the preceding relations to data compression, we proceed as in classical wavelet theory , as follows: Given a function *f (t),* its decomposition using *m*th B-spline functions in various scales. The relation between the *n*th and*(n* − 1*)*th scales, yields to:

$$f_n(t)=\sum_k c_{n,k}B_m(2^n t-k)$$

$$=\sum_k c_{n-1,k}B_m(2^{n-1}t-k)$$

$$+\sum_k d_{n-1,k}\psi_m(2^{n-1}t-k) \tag{28}$$

To relate $c_{n,k}$ to $c_{n-1,k}$ & $d_{n-1,k}$; substitute Eq. (25), back in Eq. (27), to yield

$$
\begin{aligned}
f_n(t) &= \sum_k c_{n,k}\left(\frac{1}{2}\sum_r a_{k-2r} N_m(2^{n-1}t - r)\right. \\
&\quad \left. + \frac{1}{2}\sum_r b_{k-2r}\,\psi_m(2^{n-1}t - r)\right) \\
&= \sum_r \left(\frac{1}{2}\sum_k c_{n,k} a_{k-2r}\right) B_m(2^{n-1}t - r) \\
&\quad + \frac{1}{2}\sum_r \left(\sum_k c_{n,k} b_{k-2r}\right) \Psi_{N_m}(2^{n-1}t - r) \\
&= \sum_r c_{n-1,r} B_m(2^{n-1}t - r) \\
&\quad + \sum_r d_{n-1r} \Psi m(2^{n-1}t - r) \\
\Rightarrow \quad c_{n-1,r} &= \frac{1}{2}\sum_k c_{n,k}\, a_{k-2r} \\
d_{n-1,r} &= \frac{1}{2}\sum_k c_{n,k} c_{n,k}\, b_{k-2r} \qquad (29)
\end{aligned}
$$

[..refer research paper (21)]

Thus, low-resolution coefficients can be recursively computed using high-resolution coefficients with IIR filters A(z−1)&B(z−1), as shown in Fig. 5a. To reconstruct the signal using low-resolution scales, we have

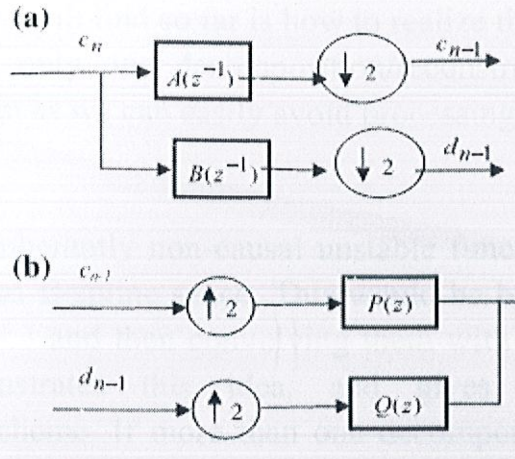


Fig.4.1 a)decomposition b)reconstruction

$$\because\ B_m(t) = \sum_r p_r B_m(2t - r)$$

$$\psi_m(t) = \sum_r q_r B_m(2t - r)$$

$$\therefore\quad B_m(2^{n-1}t - k) = \sum_r p_r B_m(2^n t - 2k - r);$$

$$\psi_m(2^{n-1}t - k) = \sum_r q_r N_m(2^n t - 2k - r) \qquad (30a)$$

$$= \sum_l p_{l-2k} B_m(2^n t - l);$$

$$= \sum_l q_{l-2k} B_m(2^n t - l)$$

$$f_n(t) = \sum_k c_{n-1,k} N_m(2^{n-1}t - k) + d_{n-1,k}\psi_m(2^{n-1}t - k)$$

$$= \sum_l \left[\sum_k (c_{n-1,k}\, p_{l-2k} + d_{n-1,k} q_{l-2k}) \right] N_m(2^n t - l) \qquad (30b)$$

$$\text{i.e.}\quad c_{n,l} = \sum_k (c_{n-1,k}\, p_{l-2k} + d_{n-1,k} q_{l-2k})$$

[..refer research paper (21)]

Figure 4.1b) shows the computations of the high-resolution coefficients from low-resolution ones using partial fractions. Figure 4.2 shows the complete decomposition/reconstruction system. Note that, due to Eq. (24), this system constitutes a PR system.

The only problem we will find so far is how to realize the unstable IIR functions A(z−1), B(z−1). If only one decomposition/reconstruction is needed, there should be no problem as we can easily avoid processing with

$\frac{1}{E(z^2)}$, which is inherently non-causal unstable function due to Eq. (11), by projection to the dual B-spline space. This would be by computing the dual of the c's, which are $\hat{c}$, and then convolving them with $\hat{N}_m(z)$ to reconstruct f (t). Figure 6 illustrates this idea, and gives the complete 1-level analysis/synthesis scheme. If more than one decomposition levels is required, the following scheme is proposed for implementing A(z−1)

and B(z−1), as summarized below:

1. Expand $\frac{1}{E(z^2)}$ using partial fractions expansion, i.e.

$$\begin{aligned}\frac{1}{E(z^2)} &= \sum_{k=1}^{m-1}\left(\frac{A_k}{1-z_k z^{-2}} + \frac{B_k}{1-\frac{z^{-2}}{z_k}}\right), \quad |z_k| < 1 \\ &= \sum_{k=1}^{m-1}\left[A_k \sum_{n=0}\left(z_k z^{-2}\right)^n - z_k B_k z^2 \sum_{n=0}\left(z_k z^2\right)^n\right] \\ &= \sum_{k=-\infty}^{\infty} h_k z^{2k} \qquad (31)\end{aligned}$$

[..refer research paper (21)]

Notice that the expansion is symmetric around some h_k,not necessary h_0.Moreover, it is stopped when expansion coefficients fall below a specified threshold.

2. Form $F_0 = \left(z\, E(z)\, \tilde{P}(z)\right) \otimes h(z), \quad F_1 = zP(-z) \otimes h(z)$
3. While keeping both of F_0, F_1 Symmetric around their centers, truncate both of them to yield almost perfect reconstruction and zero aliasing. This is achieved when $\text{length}(P(z)) + \text{length}(F_0(z)) = \text{length}(Q(z)) + \text{length}(F_1(z)))$.

[..refer research paper (21)]

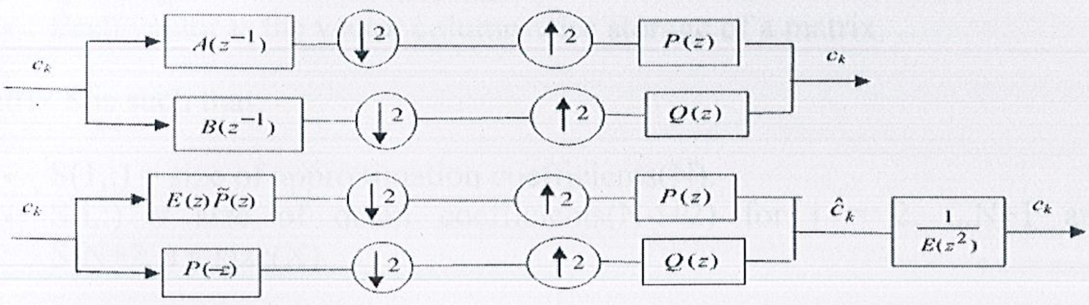


Fig.4.2 Complete input-to-output 1-level decomposition/reconstruction section

4.5 Image noise cancellation procedure:

The proposed B-spline wavelets are used in Image noise cancellation, as follows:

1) Decompose the received noisy image using the proposed B-spline wavelet to an arbitrary n decomposition level.
Denote the resulting detail and approximation coefficients by *d*1, *d*2, . . ., *dn*,*an*, respectively.

2) As the next step, The Matlab software/function *wavedec2* under 2-D Discrete Wavelets Toolbox category mainly implements the wavelet decomposition of the obtained matrix X representing the noised image at level N, using the wavelet named in string '*wname*' as described below,

[C,S] = wavedec2(X,N,'*wname*')

Outputs are the decomposition vector C and the corresponding bookkeeping matrix S.

Vector C is organized as

- C = [A(N) | H(N) | V(N) | D(N) | ...

- H(N-1) | V(N-1) | D(N-1) | ... | H(1) | V(1) | D(1)].

where A, H, V, D, are row vectors such that

- A = approximation coefficients
- H = horizontal detail coefficients
- V = vertical detail coefficients
- D = diagonal detail coefficients
- Each vector is the vector column-wise storage of a matrix.

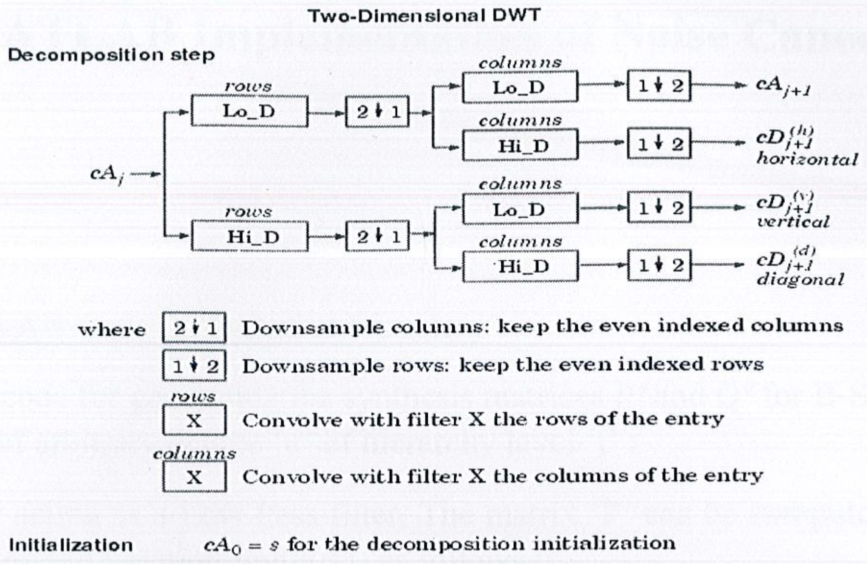
Matrix S is such that

- S(1,:) = size of approximation coefficients(N).
- S(i,:) = size of detail coefficients(N-i+2) for i = 2, ...N+1 and S(N+2,:) = size(X).

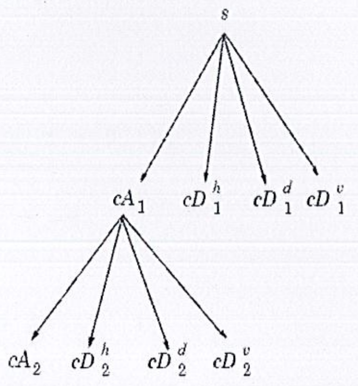
Decomposition Steps :

For images, an algorithm similar to the one-dimensional case is possible for two-dimensional wavelets and scaling functions obtained from one-dimensional ones by tensor product.

This kind of two-dimensional DWT leads to a decomposition of approximation coefficients at level *j* in four components: the approximation at level *j*+1, and the details in three orientations (horizontal, vertical, and diagonal).The following chart describes the basic decomposition step for images:


Two-Dimensional DWT
Decomposition step
rows
Lo_D
2↓1
columns
Lo_D
1↓2
cA_{j+1}
columns
Hi_D
1↓2
$cD^{(h)}_{j+1}$ horizontal
cA_j
rows
Hi_D
2↓1
columns
Lo_D
1↓2
$cD^{(v)}_{j+1}$ vertical
columns
Hi_D
1↓2
$cD^{(d)}_{j+1}$ diagonal
where
2↓1 Downsample columns: keep the even indexed columns
1↓2 Downsample rows: keep the even indexed rows
rows X Convolve with filter X the rows of the entry
columns X Convolve with filter X the columns of the entry
Initialization $cA_0 = s$ for the decomposition initialization

So, for *J*=2, the two-dimensional wavelet tree has the form,



3) Finally, **Image noise cancellation** is achieved using the denoised detail coefficients obtained during the second step as given above, together with the synthesis coefficient matrix obtained at the first step using the proposed B-spline reconstruction system.

Chapter V

MATLAB Implementation of Noise Cancellation

5.1 MATLAB Code for B-Spline Wavelets

Here, the code for generating the synthesis matrices P^J and Q^J for B-Spline wavelets of arbitrary degree 'd' of hierarchy level 'j' :

(i) For acting as a Low Pass filter, The matrix 'P' can be computed with the following function findP(d,j) as follows :

```
function p = findP(d,j)

%d=fix(d);

if d < 0
error('must have d >=0.');
end;

%j=fix(j);

if j < 1
error('mus have j >=1.');
end;

if d == 0
    p=[1; 1];
    for i = 2:j
        p=[p zeros(size(p)); zeros(size(p)) p];
    end;
else
    u = knots(d,j-1);
    g = greville(d,u);
    p=eye(2^(j-1) + d);
    for k = 0:2^(j-1)-1
        [u, g, p] = insertknot(d,u,g,p,(2*k+1)/2^j);
    end;
end;

return;
```

(ii) For acting as a High Pass filter, The matrix 'Q' can be computed with the following function findQ(d,j) as follows :

```
function Q = FindQ(d, j, normalization)
if margin < 3
  normalization = 'min';
elseif ~strcmp(normalization, 'min') & ~strcmp(normalization, 'max') &
~strcmp(normalization, '12')
  error('FindQ: normaization must be ''min'', ''max'', or ''12''.');
end;
p = findP(d, j);
i = inner(d, j);
m = p'*i;
[m1, m2] = size(m);
n = m2 - rank(m);
q = zeros(m2, n);
found = 0;
start_col = 0;

while (found < n/2) & (start_col < m2)
    start_col = start_col + 1 + (found > d);
    width = 0;
    rank_def = 0;
    while ~rank_def & (width < m2 - start_col +1)
        width = width +1;
        submatrix = m(:,start_col:start_col+width-1);
        rank_def = width - rank(submatrix);
    end;

    if rank_def
        q_col = null(submatrix);
        if strcmp(normalization, 'min')
            q_col = q_col/min(abs(q_col + 1e38*(abs(q_col) < 1e-10)));
        elseif strcmp(normalization, 'max')
            q_col = q_col/max(abs(q_col));
        end;

        q_col = q_col*(-1)^(start_col + floor((d+1)/2) + (q_col(1,1) > 0));

        found = found + 1;

        q(start_col:start_col + width-1,found) = q_col;

        q(:,n-found+1) = flipud(q(:,found))*(-1)^(d+1);
    end;
end;

if strcmp(normalization,'12')
    ip = q'*i*q;
    q = q*diag(1./sqrt(diag(ip)));
end;

return;
```

The MATLAB code for all the functions called in between the above two functions can be referred from the appendix at the end of the report.

5.2 MATLAB Code for **wavedec2** Function

```
[C,S] = wavedec2(X,N,Lo_D,Hi_D)
```

Here,

Outputs are the decomposition vector C and the corresponding bookkeeping matrix S.

N must be a strictly positive integer.

Instead of giving the wavelet name, you can give the filters.

For [C,S] = wavedec2(X,N,Lo_D,Hi_D),

Lo_D is the decomposition low-pass filter and Hi_D is the decomposition high-pass filter.

Vector C is organized as

- C = [A(N) | H(N) | V(N) | D(N) | ...
- H(N-1) | V(N-1) | D(N-1) | ... | H(1) | V(1) | D(1)].
-

where A, H, V, D, are row vectors such that

- A = approximation coefficients
- H = horizontal detail coefficients
- V = vertical detail coefficients
- D = diagonal detail coefficients
- Each vector is the vector column-wise storage of a matrix.

Matrix S is such that

- S(1,:) = size of approximation coefficients(N).
- S(i,:) = size of detail coefficients(N-i+2) for i = 2, ...N+1 and S(N+2,:) = size(X).

5.3 'Image Noise Cancellation' Main Function to be applied over the noised image combining above two approaches' detailed & approximation coefficients matrices :

```
close all
clear all
I = imread('lena512color.tifF');
```

```
figure,imshow(I)
J= rgb2gray(I);
figure,imshow(J)
K = im2double(J);

nsd = imnoise(J,'salt & pepper',0.3);
figure,imshow(nsd)

Lo_D = findP(1, 1);


Hi_D = FindQ(1,1);

[c,s] = WAVEDEC2(J,2,Lo_D,Hi_D)
 A = APPCOEF2(c,s,Lo_D,Hi_D,2)
 [chd2,cvd2,cdd2] = detcoef2('all',c,s,2);
 DH=chd2;
 DV=cvd2;
 DD=cdd2;

 for k = 1:level

     A = APPCOEF2(c,s,Lo_D,Hi_D,2)
     [chd2,cvd2,cdd2] = detcoef2('all',c,s,2);

     A{k} = wcodemat(A{k});
     H{k} = wcodemat(H{k});
     V{k} = wcodemat(V{k});
     D{k} = wcodemat(D{k});

 End

level=2;

 for k = 1:level
        subplot(level,4,aff+1); image(A{k});
        title(['Approximation A',num2str(k)]);
        subplot(level,4,aff+2); image(H{k});
        title(['Horizontal Detail ',num2str(k)]);
        subplot(level,4,aff+3); image(V{k});
        title(['Vertical Detail ',num2str(k)]);
        subplot(level,4,aff+4); image(D{k});
        title(['Diagonal Detail ',num2str(k)]);
        aff = aff + 4;
 end
```

The MATLAB code for all the functions called in between the above code can be referred from the appendix at the end of the report.

Chapter VI

Conclusion

Through our noise cancellation method using B-Splines wavelets, we are able to remove the noise from the image to a satisfactory level and thus increasing the information content in the image which then could be used for the various other applications. We see that our proposed 'Noise cancellation' method which combines both *wavdec2* toolbox method clubbed up with our designed *B-spline filters* is better and effective than the other available methods. Thus, B-Splines wavelets are much better options then the other types of wavelets for image noise cancellation.

BIBLIOGRAPHY

1. Rajesh Siddavatam, Pradeep Kumar "An Evolutionary Approach to Image Noise Cancellation Using Adaptive Particle Swarm Optimization (APSO)" (2009)

2. Keys, R.G.: Cubic convolution interpolation for digital image processing. IEEE Trans. Acoust. Speech Signal Process. 29(6), 1153–1160 (1981)

3. Hämmerlin,G.,Hoffmann,K.-H.: Numerical Mathematics. Undergraduate Texts in Mathematics. Springer Verlag, New York (1991)

4. Schumaker, L.L.: Spline Functions: Basic Theory, Pure and Applied Mathematics. Wiley, New York (1981)

5. Goswami, J.C., Chan, A.K.: Fundamentals of Wavelets, Theory, Algorithms and Applications. Wiley, New York (1999)

6. Unser, M.: Splines—a perfect fit for signal and image processing Signal Process. Mag. 16(6), 22–38 (1999)

7. Unser, M., Aldroubi, A., Eden, M.: B-Spline signal processing. Part I: theory. IEEE Trans. Signal Process. 41(2), 821–833 (1993)

8. Unser, M., Aldroubi, A., Eden, M.: B-Spline signal processing. Part II: efficiency design and applications. IEEE Trans. Signal Process. 41(2), 8834–8848 (1993)

9. Unser, M., Aldroubi, A., Eden, M.: Fast B-spline transforms for continuous image representation and interpolation. IEEE Trans. Pattern Anal. Mach. Intell. 13(3), 277–285 (1991)

10. Aldroubi, A., Abry, P., Unser, M.: Construction of biorthogonal wavelets starting from any two multiresolutions. IEEE Trans. Signal Process. 46(4), 1130–1133 (1998)

11. Unser, M., Blu, T.: Fractional splines and wavelets. SIAM Rev. 42(1), 43–67 (2000)

12. Fahmy, M.F., Elhameed, T.A., Fahmy, G.F.: A fast BSPLINE-based method for image zooming and compression. In: Proceedings of 24th URSI Conference, Ain Shams University, Cairo, Egypt,March 2007

13. Van De Ville, D., Sage, D., Bala′ c, K., Unser, M.: The Marr wavelet pyramid and multiscale directional image analysis. EUSIPCO, August 25–29, Switzerland (2008)

14. Betram, M., Duchaineau, M., Hamann, B., Joy, K.: Generalized B-spline sub-division surface wavelets for geometry compression. IEEE Trans. Vis. Comput. Graph. 10(3), 326–338 (2004)

15. Blu, T.: A new design algorithmfor two-band orthonormal rational filter banks and orthonormal rational wavelets. IEEE Trans. Signal Process. 46(6), 1494–1504 (1998)

16. Müller, F., Brigger, P., Illgner, K., Unser, M.: Multiresolution approximation using shifted splines. IEEE Trans. Signal Process. 46(9), 2555–2558 (1998)

17. Martinez-Garcia,M.: Quantifying filter bank decorrelating performance via matrix diagonality. Signal Process. 89, 116–120 (2009)

18. Said, A., Pearlman, W.A.: A new fast and efficient image codec based on set partitioning in hierarchical trees. IEEE Trans. Circuits Syst. Video Technol. 6, 243–250 (1996)

19. Antonini,M., Barlaud,M.,Mathieu, P., Daubechies, I.: Image coding using wavelet transform. IEEETrans. Image Process. 1(2), 205–220 (1992)

20. Blu, T., Luisier, F.: The SURE-LET approach to image denoising. IEEE Trans. Image Process. 16(11), 2778–2786 (2007)

21. Fahmy,M.F., Raheem,G.A.,Mohamed,U.S., Fahmy,O.F., Fahmy,G.: Denoising and image compression using B-spline wavelets. In: IEEE International Symposium on Signal Processing and Information Technology, pp. 1–6, 16–19, December 2008

22. Shapiro, J.M.: Embedded image coding using zerotrees of wavelet coefficients. IEEE Trans. Signal Process. 41(12), 3445–3462 (1993)

APPENDIX

Implementations of important functions to be called from the main codes of generations of B-Spline wavelets and noise cancellation procedure:

- Insertknot function :

```
function [uret,gret,pret] = insertknot(d,u,g,p,unew)

    uret = sort([u unew]);
    gret = greville(d,uret);
    pret = polyeval(g,p,gret);

return;
```

- BernsteinWeights function :

```
function w = BernsteinWeights(d, j)
w = eye(2^j + d);
if d == 0
  return;
end;

u = knots(d,j);
g = greville(d, u);
for i = 1:2^j - 1
  for r = 1:d
    [u,  g,  w] = insertknot(d, u, g, w, i/2^j);
  end;
end;
return;
```

- Berstinner function :

```
function i = berstinner(d)

k = ones(d+1, 1)*[0:d];
j = k';
i = Choose(d, k).*Choose(d, j)./(Choose(2*d, k+j)*(2*d + 1));

return;
```

- knots function :

```
function x = knots(d,j)
    x = [zeros(1, d-1)  [0:2^j-1]/2^j ones(1,d)];
return;
```

- polyeval function :

```
function pret = polyeval(g,p,gnew)

    [m, n] = size(p);
    if length(g) ~= m
          error('g & p must be same');
    end;

    for i = 1:length(gnew)
        row = max(find(g<=gnew(i)));

        if row == m
            pret(i,:) = p(m,:);
        else
            frac = (g(row+1) - gnew(i))/(g(row+1)-g(row));
            pret(i,:) = frac*p(row,:) + (1-frac)*p(row+1,:);
        end;
    end;

return;
```

- Factorial function :

```
function t = Factorial(m)
[r,c] = size(m);
t = zeros(r, c);
for i = 1:r
  for j = 1:c
    t(i,j) = prod(2:m(i,j));
  end;
end;
return;
```

- Choose function :

```
function c = Choose(n,r)
c=Factorial(n)./(Factorial(r).*Factorial(n-r));
return;
```

- inner function :

```
function i = inner(d,j)
i0 = berstinner(d);
n = 2^j + d;
i = zeros(n);
w=BernsteinWeights(d,j);
for k=1:n
  w1=reshape(w(:,k), d+1, 2^j);
  for l=k:n
    w2=reshape(w(:,l), d+1, 2^j);
    i(k,l) = trace(w1'*i0*w2);
    i(l,k) = i(k,l);
  end;
end;
i = i / 2^j;
return;
```

- polyeval function :

```
function pret = polyeval(g,p,gnew)

    [m, n] = size(p);
    if length(g) ~= m
        error('g & p must be same');
    end;

    for i = 1:length(gnew)
        row = max(find(g<=gnew(i)));

        if row == m
            pret(i,:) = p(m,:);
        else
            frac = (g(row+1) - gnew(i))/(g(row+1)-g(row));
            pret(i,:) = frac*p(row,:) + (1-frac)*p(row+1,:);
        end;
    end;

return;
```