**KATHOLIEKE UNIVERSITEIT LEUVEN**
FACULTEIT TOEGEPASTE WETENSCHAPPEN
DEPARTEMENT COMPUTERWETENSCHAPPEN
AFDELING NUMERIEKE ANALYSE EN
TOEGEPASTE WISKUNDE
Celestijnenlaan 200A – B-3001 Heverlee

# WAVELETS: SOFTWARE AND APPLICATIONS

Promotoren:
Prof. Dr. ir. D. Roose
Prof. Dr. A. Bultheel

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de toegepaste wetenschappen

door

**Geert UYTTERHOEVEN**

April 1999

**KATHOLIEKE UNIVERSITEIT LEUVEN**
FACULTEIT TOEGEPASTE WETENSCHAPPEN
DEPARTEMENT COMPUTERWETENSCHAPPEN
AFDELING NUMERIEKE ANALYSE EN
TOEGEPASTE WISKUNDE
Celestijnenlaan 200A – B-3001 Heverlee

# WAVELETS: SOFTWARE AND APPLICATIONS

Jury:
Prof. Dr. ir. R. Govaerts, voorzitter
Prof. Dr. ir. D. Roose, promotor
Prof. Dr. A. Bultheel, promotor
Prof. Dr. ir. B. De Moor
Prof. Dr. ir. R. Cools
Prof. Dr. J.-P. Antoine
(Université catholique de Louvain)

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de toegepaste wetenschappen

door

**Geert UYTTERHOEVEN**

U.D.C. 681.3∗I4

April 1999

D/1999/7515/17

ISBN 90-5685-189-X

Rev. 1

# Wavelets: Software and Applications

Geert Uytterhoeven
Department of Computer Science, K.U.Leuven
Celestijnenlaan 200A, B-3001 Heverlee, Belgium

**Abstract**

During the last decade, the wavelet transform has proven to be a valuable tool in many application fields. In this thesis we focus on two applications: image processing and compression to create low-dimensional models for dynamical systems.

We worked out the details of integer wavelet transforms, based on the lifting scheme, for a class of biorthogonal wavelets (Cohen-Daubechies-Feauveau). Based on this we designed and implemented a software library called *WAILI* (Wavelets with Integer Lifting) that provides wavelet transforms and wavelet-based image processing operations on two-dimensional images. Later we added support for very large images and block-based processing.

We created a new kind of second-generation wavelets on a rectangular grid, based on a red-black blocking scheme. These Red-Black wavelets are less anisotropic than tensor product wavelets.

We reduced the complexity of the Proper Orthogonal Decomposition (POD) by biorthogonal wavelet packet compression and evaluated the resulting Approximate POD by analyzing a large-scale dynamical system, described by partial differential equations.

# Preface

My first contacts with wavelets were in 1993, when I started working on my Master's Thesis about compression using multi-wavelets, under supervision of Prof. Adhemar Bultheel, with Wim Sweldens as daily advisor. After I finished my thesis, Wim Sweldens and Prof. Björn Jawerth gave me the opportunity to visit the Department of Mathematics at the University of South Carolina for one month, where I continued the work that I started with my thesis.

In September 1994 I joined the Scientific Computing Research Group of the Department of Computer Science at K.U.Leuven, under the direction of Prof. Dirk Roose, where wavelet research is done in collaboration with Prof. Adhemar Bultheel of the Numerical Approximation and Linear Algebra Group of the same department. I want to thank both for becoming the advisors for my Ph.D. Thesis.

My first research topic was wavelet-based compression to create low-dimensional models for dynamical systems. After about 1.5 years I went back to the area of image processing, thanks to the Flemish Information Technology Action Program.

Within the K.U.Leuven wavelet team, I enjoyed working together with Maurits Malfait and Maarten Jansen (wavelet-based noise reduction) and Filip Van Wulpen (development of WAILI). At the Department of Computer Science, there lives an open and pleasant atmosphere, and I had lots of interesting discussions with colleagues.

I kept in touch with Wim Sweldens and his wife Kirsten Schulz, who were generous in their hospitality when I visited the United States.

I would like to thank Prof. Bart De Moor for being a member of the reading committee, and Prof. René Govaerts, Prof. Ronald Cools and Prof. Jean-Pierre Antoine for being members of the jury.

My research was financially supported by the projects IT/IF/05 (Belgian programme on Informatics for the Future) and IUAP-17 (Belgian programme on Inter-university Poles of Attraction), initiated by the Belgian State — Prime Minister's Service — Federal Office for Scientific, Technical and Cultural Affairs, and by the project ITA/950244 (Flemish Information Technology Action Program), initiated by the Flemish Government, in collaboration with Eurosense and Barco. I would like to thank these institutes for their support.

Last but not least, I owe a lot to my parents, who gave me the opportunity to study, and to my sisters Inge and Katrien, who always supported me during the previous years. Also I want to thank my friends in and near (the world is getting smaller :-) the city of Leuven for the great times we had together.

Geert Uytterhoeven
April 1999

## Nederlandse Samenvatting

# Wavelets: Programmatuur en Toepassingen

## Inhoudsopgave


**1 Inleiding** **viii**

1.1 Een overzicht van wavelet-onderzoek . . . . . . . . . . . . . viii
1.2 Een overzicht van de thesis . . . . . . . . . . . . . . . . . . . . . x
1.3 Situering van het onderzoek . . . . . . . . . . . . . . . . . . xi

**2 Wavelets** **xii**

2.1 Wat zijn wavelets? . . . . . . . . . . . . . . . . . . . . . . . . xii
2.2 Het waarom van wavelets . . . . . . . . . . . . . . . . . . . . . xii
2.3 Wavelets en multiresolutie-analyse . . . . . . . . . . . . . . . xiii
2.4 De wavelet-transformatie en translatie-invariantie . . . . . . . xv
2.5 De redundante wavelet-transformatie . . . . . . . . . . . . . . xvi
2.6 Wavelet-packets . . . . . . . . . . . . . . . . . . . . . . . . . . xvi

**3 Het liftingschema** **xvi**

3.1 Voorspellen en bijsturen . . . . . . . . . . . . . . . . . . . . xvi
3.2 Eenmaking van de twee modellen . . . . . . . . . . . . . . . xviii
3.3 De gehele-getallen-wavelet-transformatie . . . . . . . . . . . xx
3.4 Behandeling van randen . . . . . . . . . . . . . . . . . . . . . xxi
3.5 Conclusies . . . . . . . . . . . . . . . . . . . . . . . . . . . . . xxii

**4 Wavelet-gebaseerde beeldverwerking** **xxii**

4.1 Digitale beelden . . . . . . . . . . . . . . . . . . . . . . . . . xxii
4.2 Ruisverwijdering met wavelets . . . . . . . . . . . . . . . . . xxii
4.3 Schalering van beelden . . . . . . . . . . . . . . . . . . . . . xxiii

4.4 Knip- en plakwerk . . . . . . . . . . . . . . . . . . . . . . . xxiii
4.5 Beeldcompressie . . . . . . . . . . . . . . . . . . . . . . . xxiii

**5 Ontwerp van de programmabibliotheek PIEFPAK xxiv**
5.1 Ontwerpcriteria . . . . . . . . . . . . . . . . . . . . . . . . . xxiv
5.2 Wavelet-gebaseerde bewerkingen . . . . . . . . . . . . . . . . . xxvi
5.3 Implementatieaspecten . . . . . . . . . . . . . . . . . . . . . xxvii
5.4 Samenvatting van de functionaliteit van PIEFPAK . . . . . . . xxvii
5.5 Evaluatie van het compressiepotentieel . . . . . . . . . . . . . xxviii

**6 Grootschalige beeldverwerking xxviii**
6.1 Zeer grote beelden en tegeling . . . . . . . . . . . . . . . . . . xxix
6.2 Beeldverwerking op getegelde beelden . . . . . . . . . . . . . xxx
6.3 Transformatieresultaten . . . . . . . . . . . . . . . . . . . . . xxxi

**7 De rood-zwart-wavelet-transformatie xxxi**
7.1 Rood-zwart en blauw-geel . . . . . . . . . . . . . . . . . . . xxxi
7.2 Eigenschappen van de rood-zwart-wavelet-transformatie . . . xxxii
7.3 Vergelijking met de CDF $(2,2)$ tensorproduct-wavelets . . . . xxxiii

**8 Experimenten met een wavelet-gebaseerde APOD xxxiii**
8.1 De orthogonale eigenontbinding . . . . . . . . . . . . . . . . xxxiii
8.2 De benaderende orthogonale eigenontbinding . . . . . . . . . xxxiv
8.3 Resultaten . . . . . . . . . . . . . . . . . . . . . . . . . . . . xxxv

**9 Conclusies en suggesties voor toekomstig onderzoek xxxvi**
9.1 Wavelet-transformaties . . . . . . . . . . . . . . . . . . . . . xxxvi
9.2 Een programmabibliotheek voor wavelet-gebaseerde beeldverwerking . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . xxxvi
9.3 Een benaderende POD m.b.v. wavelet-packets . . . . . . . . . xxxvii
9.4 Suggesties voor toekomstig onderzoek . . . . . . . . . . . . . xxxviii

# 1 Inleiding

Tijdens de voorbije jaren heeft de wavelet-transformatie haar belang bewezen als een hulpmiddel voor o.a. signaal- en beeldverwerking en numerieke analyse, en dit in allerlei toepassingsgebieden.

## 1.1 Een overzicht van wavelet-onderzoek

Wavelet-theorie is een synthese van ideeën uit vele verschillende onderzoeksdomeinen. Ze combineert aspecten zoals filterbanken, multiresolutie, (bi)orthogonaliteit, . . .

De eerste wavelet-transformaties waren continue wavelet-transformaties. Bij een continue wavelet-transformatie (CWT) wordt een signaal geanalyseerd door inwendige producten te berekenen met testfuncties. Alle testfuncties worden gegenereerd door één oscillerende functie met een compacte drager. Deze wavelet-functie moet aan slechts één voorwaarde voldoen: haar integraal moet nul zijn. Men genereert verschillende testfuncties als dilaties en translaties van de wavelet-functie: de analyse van het signaal gebeurt dus door een instelbaar tijds-frequentie-venster. Deze lokaliteit in zowel tijd als frequentie biedt een aantal voordelen t.o.v. Fourieranalyse [21].

De discrete wavelet-transformatie (DWT) is een aanpassing aan digitale signalen die discreet zijn [43]. Meestal beperkt men niet enkel de translaties tot veelvouden van de bemonsteringsafstand, maar neemt men alleen dyadische dilatiefactoren. Men onderscheidt dus meerdere resolutieniveaus, met een schaalfactor 2 tussen elke 2 opeenvolgende niveaus. Hierdoor bekomt men een multiresolutie-analyse [34] die verwant is met piramidale schema's zoals de Laplaciaanse piramide [1].

Als de wavelet-functies orthogonaal zijn is de DWT stabiel. De eerste orthogonale wavelets met compacte support werden geconstrueerd door Daubechies [12]. Deze lieten een efficiënte implementatie toe: de snelle wavelet-transformatie (FWT), ook wel gekend als het filterbankalgoritme. Hier heeft men dus een verband met filterbanken en subband-codering. Door de orthogonaliteitsvoorwaarden te verzwakken tot biorthogonaliteitsvoorwaarden kreeg men biorthogonale wavelets, die een combinatie van gunstige eigenschappen zoals symmetrie en een compacte drager toelieten. Een voorbeeld hiervan zijn de biorthogonale wavelets zoals geconstrueerd door Cohen, Daubechies en Feauveau [9].

Wavelet-packets vormen een uitbreiding van de wavelet-theorie: zij voorzien in een aanpasbare basis gebaseerd op het filterbankalgoritme, en vullen daarmee de ruimte tussen de standaard wavelet-ontbindingsboom en subband-codering.

Het liftingschema, ontwikkeld door Sweldens, ontstond als een methode om een gegeven discrete wavelet-transformatie te verbeteren met het oogmerk specifieke bijkomende eigenschappen te verkrijgen [48]. Later werd het ook een efficiënt algoritme om een willekeurige wavelet-transformatie te berekenen als een opeenvolging van eenvoudige liftingstappen [14]. Digitale signalen bestaan meestal uit een sequentie van gehele getallen, terwijl wavelet-transformaties als resultaat vlottende-kommagetallen geven. Om een efficiënte reversibele implementatie te verkrijgen is het van het grootste belang om over een algoritme te beschikken dat gehele getallen omzet in gehele getallen. Gelukkig kan een liftingstap op eenvoudige wijze worden aangepast om gehele getallen te transformeren, met behoud van de reversibiliteit [6]. Hierdoor werd het liftingschema een methode om reversibele gehele-getallen-wavelet-transformaties te implementeren.

Het liftingschema werd eveneens veralgemeend tot een methode om tweede-generatie-wavelets te construeren [49]. Dit zijn wavelets die niet langer gegenereerd worden als translaties en dilaties van één wavelet-functie. Het liftingschema laat toe om wavelets te construeren op hoger-dimensionale roosters, onregelmatige bemonsteringen, triangularisaties, bollen, ... [40, 31].

Wavelets worden gebruikt in tal van toepassingen, zoals signaal- en beeldverwerking, beeld- en videocompressie, wiskundige natuurkunde, numerieke analyse, communicatie, systeemtheorie, identificatie, geometrische modellering, computergrafiek, theoretisch fysica, benaderingstheorie, stochastische processen, .... Natuurlijk is deze lijst verre van compleet. We verwijzen naar de literatuur voor meer informatie, bvb. [3, 7, 8, 28, 29, 33, 36, 46, 58].

Wij concentreerden ons op twee toepassingsgebieden: beeldverwerking en compressie voor het creëren van laagdimensionale modellen voor dynamische systemen.

We werkten de details uit van gehele-getallen-wavelet-transformaties, op basis van het liftingschema, voor een klasse van biorthogonale wavelets (Cohen-Daubechies-Feauveau). Dit gebruikten we om de programmabibliotheek PIEF-PAK (WAILI) te ontwikkelen. Deze bibliotheek implementeert wavelet-transformaties en wavelet-gebaseerde beeldverwerkingsoperaties op tweedimensionale beelden [56]. Later voegden we ondersteuning toe voor zeer grote beelden en blokgebaseerde verwerking [55].

We construeerden een nieuw soort tweede-generatie-wavelets op een rechthoekig rooster, gebaseerd op een rood-zwart blokschema. Deze rood-zwart-wavelets zijn minder anisotroop dan tensorproduct-wavelets [52].

We reduceerden de complexiteit van de orthogonale eigenontbinding (POD) m.b.v. biorthogonale wavelet-packet-compressie en evalueerden de resulterende benaderende POD door de analyse van een grootschalig dynamisch systeem, beschreven door partiële differentiaalvergelijkingen [53].

## 1.2 Een overzicht van de thesis

Hoofdstuk 2 geeft een introductie tot de concepten multiresolutie-analyse en wavelets, bekeken vanuit het klassiek oogpunt. Het liftingschema, behandeld in Hoofdstuk 3, voorziet niet alleen in een kader voor de constructie van zogenaamde tweede-generatie-wavelets, maar het laat eveneens toe om de klassieke wavelet-transformaties te implementeren. Vergeleken met klassieke implementaties biedt het liftingschema verscheidene voordelen en extra eigenschappen, zoals bvb. een echte gehele-getallenvariante.

Het grootste deel van deze thesis is gewijd aan de toepassing van wavelets voor beeldverwerking. Vele beeldverwerkingsoperaties kunnen immers worden uitgevoerd in het wavelet-domein. Een aantal van deze bewerkingen komen aan bod in Hoofdstuk 4. Beeldcompressie is één van de meest belangrijke beeldverwerkingsoperaties. Het ligt voor de hand dat grote beelden best worden bewaard en behandeld in gecomprimeerde vorm. Wavelets vormen hier een oplossing aangezien wavelet-gebaseerde technieken betere compressieresultaten geven dan andere methoden.

Hoofdstuk 5 behandelt vervolgens het ontwerp en de kenmerken van *PIEFPAK*, een programmabibliotheek die de gehele-getallen-wavelet-transformatie implementeert, zoals besproken in het vorige hoofdstuk. De bibliotheek laat wavelettransformaties en wavelet-gebaseerde bewerkingen toe op tweedimensionale beelden en is geschreven in C++. In toepassingsgebieden zoals geografische informatiesystemen (GIS) worden beeldgroottes uitgedrukt in gigabytes en soms zelfs in terabytes. Hoofdstuk 6 beschrijft de uitbreiding van PIEFPAK tot een blokgebaseerde verwerking voor zulke grote beelden, wat een eventuele parallelle implementatie toelaat.

In Hoofdstuk 7 tonen we aan dat het liftingschema ook nuttig is voor de constructie van een nieuw soort tweede-generatie-wavelets op een rechthoekig rooster. Deze wavelets zijn gebaseerd op een rood-zwart blokschema en hebben een aantal interessante eigenschappen die niet mogelijk zijn bij tensorproducten van traditionele wavelets.

Wavelets hebben niet alleen toepassingen in beeld- en signaalverwerking, maar zijn ook van belang voor de constructie van laagdimensionale modellen voor dynamische systemen. De analyse van de originele partiële differentiaalvergelijkingen die het dynamisch systeem beschrijven kan worden vervangen door de analyse van een laagdimensionaal model dat de originele vergelijkingen benadert. De orthogonale eigenontbinding (Proper Orthogonal Decomposition — POD) laat toe om uitgaande van oplossingen van de partiële differentiaalvergelijkingen een "beste" basis te construeren voor zo'n model, maar dat is een dure operatie. In Hoofdstuk 8 stellen we een methode voor om de complexiteit van de POD te reduceren, gebruik makende van compressie m.b.v. biorthogonale wavelet-packets. We evalueren de hieruit voortvloeiende benaderende POD door een laagdimensionaal model te construeren voor een grootschalig dynamisch systeem.

## 1.3 Situering van het onderzoek

Het grootste deel van dit onderzoek werd uitgevoerd in het kader van het project "Wavelet-Based Interactive Video Communication and Image Database Consulting" (ITA/950244). Dit project betrok drie partners (Eurosense, Barco en K.U.Leuven) en had de volgende doelstellingen: de ontwikkeling van een programmabibliotheek voor wavelet-transformaties, met inbegrip van de ondersteuning van zeer grote beelden en beelden samengesteld uit deelbeelden, de implementatie van beeldcompressie en -verwerking, de integratie in een geografisch informatiesysteem (GIS), en een hardware-ondersteunde implementatie van de wavelet-transformatie voor reële-tijd-video. De ontwikkeling van de programmabibliotheek en de wavelet-gebaseerde beeldverwerkingsoperaties werden door ons uitgevoerd.

De onderzoeksresultaten uit Hoofdstuk 8, betreffende de benaderende POD, dateren van vóór het bovenvermelde project. Zij kaderen in het onderzoek naar de analyse van periodieke oplossingen van grootschalige dynamische systemen, beschreven door partiële differentiaalvergelijkingen, door andere leden van de onderzoeksgroep Technisch-Wetenschappelijk Rekenen van het departement Computerwetenschappen.

# 2 Wavelets

## 2.1 Wat zijn wavelets?

Het basisidee achter wavelet-analyse is de ontbinding van een signaal $f$ in een basis van functies $\Psi_i$:

$$f = \sum_i a_i \Psi_i.$$

Om een efficiënte voorstelling voor $f$ te bekomen met slechts een weinig coëfficiënten $a_i$ is het zeer belangrijk om een geschikte familie van functies $\Psi_i$ te gebruiken. Deze familie moet namelijk over eigenschappen beschikken die sterk overeenkomen met het signaal dat we wensen voor te stellen.

In realiteit zijn signalen meestal zowel beperkt in de tijd (of ruimte, in het geval van beelden) als in frequentie. Wavelets ("golfjes") vormen een compromis tussen functies die ofwel enkel beperkt zijn in de tijd, ofwel enkel in frequentie. Ze combineren dus het beste uit twee werelden.

## 2.2 Het waarom van wavelets

Eén van de belangrijkste eigenschappen van wavelets is dat ze goede decorrelatie-eigenschappen bezitten:

- Wavelets zijn gelokaliseerd in zowel ruimte/tijd als in schaal/frequentie. Zij slagen er dus in lokale eigenschappen van een signaal te detecteren.

- Wavelets zijn gebaseerd op een multiresolutie-analyse. M.b.v. een wavelet-ontbinding kan men een signaal analyseren op verschillende resolutieniveaus of schalen.

- Wavelets zijn voldoende zachtverlopend. Zachtverlopende signalen kunnen dus goed worden benaderd met een beperkt aantal basisfuncties.

Bovendien kunnen de wavelet-transformatie en haar inverse berekend worden m.b.v. snelle en stabiele algoritmen.

## 2.3 Wavelets en multiresolutie-analyse

Klassiek worden discrete wavelets geïntroduceerd m.b.v. het concept multiresolutie-analyse.

### Multiresolutie-analyse

In een multiresolutie-analyse [34] wordt de ruimte $\mathbf{L}^2$, de vectorruimte der kwadratisch integreerbare functies in $\mathbf{R}$, ontbonden in geneste deelruimten $V_j$ ($V_j \subset V_{j+1}$). De sluiting van de unie van deze deelruimten is $\mathbf{L}^2$ en hun doorsnede bevat alleen de nulfunctie. In het dyadische geval heeft een functie $f(x)$ die behoort tot $V_j$ de volgende eigenschappen:

$$\begin{aligned} f(x) \in V_j \quad &\Leftrightarrow \quad \text{dilatie } f(2x) \in V_{j+1}, \\ f(x) \in V_0 \quad &\Leftrightarrow \quad \text{translatie } f(x+1) \in V_0. \end{aligned}$$

Indien we een functie $\varphi(x) \in V_0$ kunnen vinden zodanig dat

$$\{\varphi(x-k)\}_{k\in\mathbf{Z}}$$

een basis vormt voor de ruimte $V_0$, dan noemen we dit een *schaalfunctie*. Voor de andere deelruimten $V_j$ (met $j \neq 0$) definiëren we:

$$\varphi_{j,k}(x) \equiv 2^{j/2}\varphi\left(2^j x - k\right).$$

### Wavelet-functies

Aangezien de deelruimten $V_j$ genest zijn kan men $V_{j+1}$ ontbinden in $V_j$ en $W_j$, met $W_j$ het orthogonale complement van $V_j$ in $V_{j+1}$:

$$\begin{aligned} V_j \oplus W_j \quad &= \quad V_{j+1}, \\ W_j \quad &\perp \quad V_j. \end{aligned}$$

De sluiting van de directe som van de deelruimten $W_j$ is opnieuw gelijk aan $\mathbf{L}^2$. Dit betekent dat $V_j$ een "grove resolutie"-voorstelling is van $V_{j+1}$, terwijl $W_j$ het "fijne resolutie"-verschil tussen $V_j$ en $V_{j+1}$ bevat.

Indien we een functie $\psi(x) \in W_0$ kunnen vinden die voldoet aan de translatievergelijking

$$\psi(x) \in W_0 \Leftrightarrow \text{translatie } \psi(x+1) \in W_0,$$

en waarbij

$$\{\psi(x-k)\}_{k\in\mathbf{Z}}$$

een basis vormt voor de ruimte $W_0$, dan noemen we dit een *wavelet-functie*. Voor de andere deelruimten $W_j$ (met $j \neq 0$) definiëren we:

$$\psi_{j,k}(x) \equiv 2^{j/2}\psi\left(2^j x - k\right).$$

### De snelle wavelet-transformatie

Aangezien zowel $V_0$ en $W_0$ deelruimten zijn van $V_1$ kunnen we $\varphi(x)$ en $\psi(x)$ uitdrukken in functie van de basisfuncties van $V_1$:

$$\begin{aligned}\varphi(x) &= 2\sum_k h_k\,\varphi(2x-k),\\ \psi(x) &= 2\sum_k g_k\,\varphi(2x-k).\end{aligned}$$

Dankzij de multiresolutie-analyse zijn deze verbanden ook geldig tussen $V_{j+1}$, $V_j$ en $W_j$ voor een willekeurige $j$. We noemen de coëfficiënten $h_k$ en $g_k$ de filtercoëfficiënten. Vermits $V_{j+1} = V_j \oplus W_j$ kunnen we een functie

$$f(x) = \sum_k \lambda_{j+1,k}\,\varphi_{j+1,k}(x)$$

ook ontbinden als

$$f(x) = \sum_l \lambda_{j,l}\,\varphi_{j,l}(x) + \sum_l \gamma_{j,l}\,\psi_{j,l}(x).$$

waarbij de transformatiecoëfficiënten $\lambda_{j,l}$ en $\gamma_{j,l}$ gedefinieerd zijn als

$$\begin{aligned}\lambda_{j,l} &= \sqrt{2}\sum_k h_{k-2l}\,\lambda_{j+1,k},\\ \gamma_{j,l} &= \sqrt{2}\sum_k g_{k-2l}\,\lambda_{j+1,k}.\end{aligned}$$

Deze operatie staat bekend als de *snelle wavelet-transformatie* (Fast Wavelet Transform — FWT) en heeft een complexiteit evenredig met de signaallengte ($O(n)$). De filters $h_k$ en $g_k$ zijn laag-doorlaat- respectievelijk hoog-doorlaatfilters. De inverse transformatie gebeurt op een analoge wijze. Dit algoritme wordt het filterbankalgoritme genoemd.

### Orthogonale wavelets

Als de functies $\varphi_{j,k}(x)$ en $\psi_{j,k}(x)$ orthonormaal staan t.o.v. elkaar:

$$\langle \varphi_{j,l}, \varphi_{j,l'} \rangle = \delta_{l-l'} \qquad \text{en} \qquad \langle \psi_{j,l}, \psi_{j',l'} \rangle = \delta_{j-j'}\,\delta_{l-l'},$$

is de ontbinding in de wavelet-basis stabiel en kunnen de wavelet-coëfficiënten berekend worden als inwendige producten met de basisfuncties:

$$\lambda_{j,l} = \langle f, \varphi_{j,l} \rangle \qquad \text{en} \qquad \gamma_{j,l} = \langle f, \psi_{j,l} \rangle.$$

Voor deze ontbinding geldt de gelijkheid van Parseval:

$$\|f\|^2 = \sum_k \lambda_{j+1,k}^2 = \sum_l \lambda_{j,l}^2 + \sum_l \gamma_{j,l}^2.$$

### Biorthogonale wavelets

Men kan wavelets creëren met bepaalde extra eigenschappen door de orthogonaleitsvoorwaarde te laten vallen. In het biorthogonale geval hebben we twee multiresolutie-analyses, een primale ($V_j$, $W_j$, $\varphi_{j,k}$, $\psi_{j,k}$) en een duale ($\widetilde{V}_j$, $\widetilde{W}_j$, $\widetilde{\varphi}_{j,k}$, $\widetilde{\psi}_{j,k}$), met

$$\langle \widetilde{\varphi}_{j,l}, \varphi_{j,l'} \rangle = \delta_{l-l'} \qquad \text{en} \qquad \langle \widetilde{\psi}_{j,l}, \psi_{j',l'} \rangle = \delta_{j-j'} \delta_{l-l'}.$$

De ontbinding in een biorthogonale wavelet-basis is nog steeds stabiel omdat biorthogonale wavelets een Riesz-basis vormen:

$$A\|f\|^2 \leq \sum_l \lambda_{j,l}^2 + \sum_l \gamma_{j,l}^2 \leq B\|f\|^2,$$

met $A$ en $B$ positieve constanten. De wavelet-coëfficiënten worden nu berekend als inwendige producten met de duale basisfuncties:

$$\lambda_{j,l} = \langle f, \widetilde{\varphi}_{j,l} \rangle \qquad \text{en} \qquad \gamma_{j,l} = \langle f, \widetilde{\psi}_{j,l} \rangle.$$

Het filterbankalgoritme is nog steeds bruikbaar, op voorwaarde dat we het duale filterpaar $(\widetilde{h}, \widetilde{g})$ gebruiken voor de ontbinding en het primale filterpaar $(h, g)$ voor de reconstructie.

### De wavelet-ontbindingsboom

Aangezien $V_j$ telkens wordt opgesplitst in $V_{j-1}$ en $W_{j-1}$ krijgen we een ontbindingsboom. De totale kost van deze ontbinding is nog steeds $O(n)$ omdat de hoeveelheid rekenwerk wordt gehalveerd op elk volgend resolutieniveau.

### Hogere dimensies

Bovenstaande wavelets werden gedefinieerd op de reële as $\mathbf{R}$; dit is een ééndimensionaal domein. Men kan hoger-dimensionale wavelets creëren als tensorproducten van ééndimensionale wavelets. Echte multidimensionale wavelets, bvb. rood-zwart-wavelets, vormen een valabel alternatief.

## 2.4 De wavelet-transformatie en translatie-invariantie

Een wavelet-transformatie is niet translatie-invariant omwille van de decimatiestap in het filterbankalgoritme. De transformatie is slechts translatie-invariant indien het (discreet) signaal vertraagd of versneld wordt met een veelvoud van $2^n$, met $n$ het aantal transformatieniveaus.

### 2.5 De redundante wavelet-transformatie

De redundante wavelet-transformatie is een variante van de wavelet-transformatie zonder decimatiestap. Ze heeft als voordeel dat ze wel translatie-invariant is, ten koste van meer rekenwerk, hogere geheugenvereisten $O(n \log n)$ en een meer complexe inverse transformatie.

### 2.6 Wavelet-packets

In elke stap van een wavelet-transformatie wordt het laagfrequente deel van het signaal opgesplitst in een laag- en een hoogfrequent deel. Bij een wavelet-packet-transformatie wordt ook het hoogfrequente deel verder opgesplitst, wat leidt tot een redundante boom van mogelijke basisfuncties [10]. Een wavelet-packet-transformatie heeft een complexiteit van $O(n \log n)$.

Het doel van wavelet-packets is het vinden van een "beste basis", d.w.z. het vinden van de verzameling basisfuncties die de gegevens het beste decorreleert [61]. De selectie van de beste basis gebeurt aan de hand van een kostfunctie die berekent hoeveel bits nodig zijn om de gegevens voor te stellen in een specifieke basis.

## 3 Het liftingschema

Het liftingschema is een efficiënt algoritme om wavelet-transformaties te berekenen. Oorspronkelijk werd het ontwikkeld om een gegeven wavelet-transformatie te verbeteren tot ze aan bepaalde eigenschappen voldeed. Later werd het liftingschema uitgebreid tot een algemene methode om tweede-generatie-wavelets te creëren. Deze zijn meer flexibel en laten toe om wavelets te definiëren op intervallen, onregelmatige roosters, ...

Men kan alle traditionele wavelet-transformaties implementeren m.b.v. het liftingschema. De ontbinding van een klassieke wavelet-filter in liftingstappen kan eenvoudig bekomen worden via een Euclidisch algoritme.

De theorie achter de constructie van wavelets in het vorige hoofdstuk steunt zeer sterk op Fourieranalyse. Het liftingschema daarentegen introduceert wavelets zonder gebruik te maken van het concept Fourier-transformatie. Hierdoor wordt het heel wat eenvoudiger om de basisprincipes achter wavelets uit te leggen [48, 49, 40, 47].

### 3.1 Voorspellen en bijsturen

De wavelet-transformatie van een ééndimensionaal discreet signaal is een multiresolutie-voorstelling, met als basisfuncties wavelets. Op elk resolutieniveau wordt (het laagfrequente deel van) het signaal opgesplitst in een hoog- en een laagfrequent deel, gebruik makende van corresponderende wavelet-filters.

Het liftingschema is een efficiënte implementatie van deze filteroperaties. Veronderstel dat de verzameling $\lambda_{j+1}$ het laagfrequent deel van een signaal op resolutieniveau $j+1$ vormt. Deze verzameling wordt opgesplitst in twee andere verzamelingen op niveau $j$: een laagfrequent deel $\lambda_j$ en een hoogfrequent deel $\gamma_j$. Traditioneel bestaat deze opsplitsing uit een splitsing in even en oneven monsters: de luie wavelet-transformatie.

Liftingstappen worden meestal in paren uitgevoerd. Een duale liftingstap is een predictie: de gegevens $\gamma_j$ worden voorspeld uitgaande van de $\lambda_j$. Zulke voorspelling is een goede benadering als er een hoge correlatie is tussen de twee signalen, zodat $\gamma_j$ kan vervangen worden door de predictiefout $\gamma_j - \mathcal{P}(\lambda_j)$, waarbij $\mathcal{P}$ de predictieoperator voorstelt. Dit is de decorrelerende stap.

Het originele signaal had echter een aantal eigenschappen die verloren gegaan zijn bij de overgang naar de nieuwe voorstelling, zoals bvb. het behoud van de gemiddelde waarde. Om deze eigenschappen te herstellen heeft men een primale liftingstap nodig, waarbij de verzameling $\lambda_j$ wordt bijgestuurd m.b.v. gegevens berekend uit $\gamma_j$. Dus $\lambda_j$ wordt vervangen door $\lambda_j + \mathcal{U}(\gamma_j)$, met $\mathcal{U}$ een bijsturingsoperator.

In het algemene geval kunnen meerdere liftingstappen worden uitgevoerd bij de overgang van resolutieniveau $j+1$ naar niveau $j$. Via een iteratie op $\lambda_j$ worden deze stappen herhaald uitgevoerd en verkrijgt men een multiresolutie-ontbindingsboom.

### De inverse transformatie

De inverse transformatie van het liftingschema is zeer eenvoudig te vinden. Een wavelet-transformatie wordt immers ontbonden in een aantal elementaire stappen die elk op zich eenvoudig te inverteren zijn. Dus de inverse transformatie luidt als volgt: keer de volgorde van de stappen om, inverteer de tekens in de liftingstappen en vervang de opsplitsingen door samenvoegingen.

### Voorbeeld: lineaire voorspelling

De transformatie van een discreet ééndimensionaal signaal $x = \{x_k\}$ gaat als volgt:

**Splitsing** Splits het signaal $x$ (d.w.z. $\lambda_{j+1}$) in even (d.w.z. $\lambda_j$) en oneven monsters (d.w.z. $\gamma_j$). Dit noemt men de luie wavelet-transformatie:

$$\begin{aligned} s_i &\leftarrow x_{2i}, \\ d_i &\leftarrow x_{2i+1} \end{aligned}$$

**Predictie** Voorspel de oneven monsters m.b.v. lineaire interpolatie:

$$d_i \leftarrow d_i - \frac{1}{2}(s_i + s_{i+1}).$$

**Bijsturing** Pas de even monsters aan om de gemiddelde waarde van het signaal te behouden:

$$s_i \leftarrow s_i + \frac{1}{4}(d_{i-1} + d_i).$$

Hierna vormt het signaal $s = \{s_k\}$ een grove voorstelling van het originele signaal $x$, terwijl $d = \{d_k\}$ de hoogfrequente informatie bevat die verloren ging bij de overgang van niveau $j+1$ naar $j$. Meer geavanceerde interpolatieschema's zijn mogelijk en leiden tot andere wavelets [50].

Deze transformatie werkt enkel op ééndimensionale gegevens. Voor twee dimensies kan men ze toepassen op de rijen en kolommen van een matrix, met een tensorproduct-wavelet-transformatie tot gevolg.

Later zullen we zien dat het bovenstaande voorbeeld het lifting-equivalent is van de populaire Cohen-Daubechies-Feauveau biorthogonale wavelets met twee nulmomenten voor zowel de primale als de duale wavelet (CDF $(2,2)$ wavelet) [9].

#### Flexibiliteit

Het liftingschema is veel flexibeler dan klassieke methodes voor de constructie van wavelets. Dankzij lifting kan men wavelets construeren voor hoge dimensies, op onregelmatige bemonsteringen, intervallen, triangularisaties, ... [40].

In Hoofdstuk 7.1 presenteren we een eenvoudige methode om tweedimensionale wavelets te construeren op een rechthoekig rooster m.b.v. het liftingschema.

## 3.2 Eenmaking van de twee modellen

Alle klassieke wavelet-transformaties kunnen worden geïmplementeerd m.b.v. het liftingschema.

#### De polyfasevoorstelling

De $z$-transformatie van een signaal $x = \{x_k\}$ wordt gedefinieerd als $x(z) = \sum_k x_k z^{-k}$. De $z$-transformatie van een eindig filter $h = \{h_{k_b}, \ldots, h_{k_e}\}$ is een Laurent-veelterm $h(z) = \sum_{k=k_b}^{k_e} h_k z^{-k}$.

Als we de opsplitsing van $x(z)$ in haar even en oneven componenten $x_e(z^2)$ en $x_o(z^2)$ definiëren als

$$x_e(z^2) = \frac{x(z) + x(-z)}{2} \qquad \text{en} \qquad x_o(z^2) = \frac{z}{2}\left[x(z) - x(-z)\right],$$

kunnen we het filterbankalgoritme in het $z$-domein schrijven als

$$\begin{bmatrix} \lambda(z) \\ \gamma(z) \end{bmatrix} = P(z) \begin{bmatrix} x_e(z) \\ z^{-1}x_o(z) \end{bmatrix} = \begin{bmatrix} h_e(z) & h_o(z) \\ g_e(z) & g_o(z) \end{bmatrix} \begin{bmatrix} x_e(z) \\ z^{-1}x_o(z) \end{bmatrix},$$

waarbij $x(z)$ het invoersignaal voorstelt en $\lambda(z)$ en $\gamma(z)$ de resulterende laag- respectievelijk hoogfrequente uitvoersignalen zijn. $P(z)$ wordt de polyfasematrix genoemd en bepaalt welke wavelet-transformatie wordt uitgevoerd. De polyfasematrix horende bij de luie wavelet-transformatie is de $2 \times 2$ eenheidsmatrix.

### Primale en duale lifting

Het liftingschema [48, 49, 47] vindt zijn oorsprong in een methode om een gegeven wavelet-transformatie te verbeteren, met het oogmerk wavelets met specifieke eigenschappen te bekomen.

Bij primale (duale) lifting creëert men een nieuwe filter $h^{\text{nieuw}}$ ($g^{\text{nieuw}}$), uitgaande van een filterpaar $(h,g)$:

$$\begin{aligned} h^{\text{nieuw}}(z) &= h(z) + s(z^2)g(z), \\ g^{\text{nieuw}}(z) &= g(z) + t(z^2)h(z). \end{aligned}$$

M.b.v. de polyfasevoorstelling wordt dit:

$$P^{\text{nieuw}}(z) = \begin{bmatrix} h_e^{\text{nieuw}}(z) & h_o^{\text{nieuw}}(z) \\ g_e(z) & g_o(z) \end{bmatrix} = \begin{bmatrix} 1 & s(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} h_e(z) & h_o(z) \\ g_e(z) & g_o(z) \end{bmatrix},$$

respectievelijk

$$P^{\text{nieuw}}(z) = \begin{bmatrix} h_e(z) & h_o(z) \\ g_e^{\text{nieuw}}(z) & g_o^{\text{nieuw}}(z) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ t(z) & 1 \end{bmatrix} \begin{bmatrix} h_e(z) & h_o(z) \\ g_e(z) & g_o(z) \end{bmatrix}.$$

Dit proces kan men herhalen om een aaneensluiting van alternerende primale en duale liftingstappen te bekomen. Indien de initiële matrix $P(z)$ de eenheidsmatrix is — d.w.z. de eerste stap is een luie wavelet-transformatie — dan krijgen we een product van $2 \times 2$ boven- en benedendriehoeksmatrices, bestaande uit Laurent-veeltermen.

### Ontbinding in liftingstappen

Daubechies en Sweldens [14] bewezen dat elke polyfasematrix die een wavelet-transformatie met eindige filters voorstelt kan ontbonden worden in een product van $2 \times 2$ boven- en benedendriehoeksmatrices en een diagonaalmatrix:

$$P(z) = \underbrace{\begin{bmatrix} K_1 & 0 \\ 0 & K_2 \end{bmatrix}}_{\text{normalisatie}} \prod_{i=m}^{1} \left\{ \overbrace{\begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix}}^{\text{primale lifting}} \underbrace{\begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix}}_{\text{duale lifting}} \right\}, \tag{1}$$

met $s_i(z)$ en $t_i(z)$ Laurent-veeltermen en $K_1$ en $K_2$ normalisatiefactoren. Men kan bewijzen dat alle wavelet-transformaties kunnen worden uitgevoerd als een luie wavelet-transformatie, gevolgd door een afwisseling van primale en duale liftingstappen en een normalisatie. Hieruit volgt onmiddellijk de inverse transformatie:

$$P^{-1}(z) = \prod_{i=1}^{m} \left\{ \begin{bmatrix} 1 & 0 \\ -t_i(z) & 1 \end{bmatrix} \begin{bmatrix} 1 & -s_i(z) \\ 0 & 1 \end{bmatrix} \right\} \begin{bmatrix} 1/K_1 & 0 \\ 0 & 1/K_2 \end{bmatrix}.$$

De gezochte $s_i(z)$ en $t_i(z)$ kan men vinden door een Euclidisch algoritme uit te voeren op de even en oneven delen van de wavelet-filters [14]. We geven de ontbindingen voor een aantal wavelet-transformaties uit de biorthogonale familie van Cohen-Daubechies-Feauveau.

**Voordelen van het liftingschema**

Wavelet-transformaties uitgevoerd m.b.v. het liftingschema bieden een aantal voordelen t.o.v. het klassieke filterbankalgoritme:

- Het liftingschema is sneller (maar natuurlijk nog steeds $O(n)$).
- De inverse transformatie is triviaal.
- Het liftingschema is eenvoudiger te begrijpen en te implementeren.
- Het liftingschema is meer algemeen en laat de constructie van tweede-generatie-wavelets toe.

Bovendien kunnen alle klassieke wavelet-transformaties worden geïmplementeerd m.b.v. het liftingschema, via het algoritme van Euclides. Dit algoritme vormt dus de brug tussen de klassieke wavelet-transformatie en het liftingschema.

## 3.3 De gehele-getallen-wavelet-transformatie

In vele toepassingen (bvb. beeldcompressie en -verwerking, video, audio, ...) zijn de invoergegevens gehele getallen. Bovendien zijn gehele getallen eenvoudiger te bewaren en encoderen dan vlottende-kommagetallen. Alle bovenstaande transformaties veronderstellen echter dat het invoersignaal vlottende-kommagetallen bevat. Maar zelfs als de filteroperaties worden uitgevoerd op gehele getallen verkrijgt men hierdoor rationale of reële getallen omdat de filtercoëfficiënten niet noodzakelijk geheel zijn. Om een efficiënte implementatie van de discrete wavelet-transformatie te bekomen is het dus van het grootste belang dat ook de wavelet-coëfficiënten geheel zijn (eventueel op een schaalfactor na). Dit kan niet gebeuren door de coëfficiënten eenvoudigweg af te ronden, aangezien het hierna niet meer mogelijk is het oorspronkelijke signaal te reconstrueren.

**Gehele-getallentransformaties**

Gelukkig kan het liftingschema eenvoudig gewijzigd worden tot een transformatie die gehele getallen omzet in gehele getallen en die bovendien perfect omkeerbaar is [6, 57]. Hiervoor worden een aantal afrondingsoperaties toegevoegd, aangeduid door accolades. In de predictie- en bijsturingsstappen berekenen we nu afgeronde getallen:

$$\gamma_j \leftarrow \gamma_j - \{\mathcal{P}(\lambda_j)\} \qquad \text{en} \qquad \lambda_j \leftarrow \lambda_j + \{\mathcal{U}(\gamma_j)\}\,.$$

Aangezien een operatie van de vorm

$$a^{\text{nieuw}} \leftarrow a^{\text{oud}} + \{b\}$$

$b$ niet verandert, onafhankelijk van de gebruikte afrondingsmethode, kan $a^{\text{oud}}$ altijd worden gereconstrueerd zonder enige fout als

$$a^{\text{oud}} \leftarrow a^{\text{nieuw}} - \{b\}\,.$$

De inverse operatie wordt nu:

$$\lambda_j \leftarrow \lambda_j - \{\mathcal{U}(\gamma_j)\} \qquad \text{en} \qquad \gamma_j \leftarrow \gamma_j + \{\mathcal{P}(\lambda_j)\}\,.$$

**Voorbeeld:** CDF $(2,2)$.

| Voorwaartse transformatie | | Inverse transformatie | |
|---|---|---|---|
| Splitsing: | $s_i \leftarrow x_{2i}$<br>$d_i \leftarrow x_{2i+1}$ | Inverse primale lifting: | $s_i \leftarrow s_i - \left\{\frac{1}{4}(d_{i-1}+d_i)\right\}$ |
| Duale lifting: | $d_i \leftarrow d_i - \left\{\frac{1}{2}(s_i+s_{i+1})\right\}$ | Inverse duale lifting: | $d_i \leftarrow d_i + \left\{\frac{1}{2}(s_i+s_{i+1})\right\}$ |
| Primale lifting: | $s_i \leftarrow s_i + \left\{\frac{1}{4}(d_{i-1}+d_i)\right\}$ | Samenvoeging: | $x_{2i} \leftarrow s_i$<br>$x_{2i+1} \leftarrow d_i$ |

**Normalisatie**

In het bovenstaande voorbeeld hebben we de normalisatiefactoren $K_1$ en $K_2$ verwaarloosd. In toepassingen zoals beeldcompressie moet men echter wel degelijk rekening houden met de werkelijke genormaliseerde grootte van de waveletcoëfficiënten.

## 3.4 Behandeling van randen

In de praktijk strekken signalen zich niet oneindig ver uit in ruimte of tijd, maar zijn ze beperkt tot een eindig interval. Aangezien het filterbankalgoritme een oneindige signaallengte veronderstelt moet het eindige signaal worden uitgebreid. Klassiek gebruikt men hiervoor periodieke of symmetrisch extensie, waarbij symmetrische extensie betere resultaten geeft omdat ze discontinuïteiten vermijdt [4].

Het liftingschema biedt de mogelijkheid om wavelets te construeren die gedefinieerd zijn op een interval. Nochtans hebben wij niet gekozen voor deze optie, maar voor symmetrische extensie omdat dit eenvoudiger te implementeren is en ruimschoots voldoende is voor onze toepassingen. Bovendien laat het liftingschema symmetrische uitbreiding toe met alle soorten wavelets, en niet alleen met symmetrische wavelets.

### 3.5 Conclusies

Het liftingschema laat een efficiënte implementatie van een reversibele gehele-getallen-wavelet-transformatie toe. Alle klassieke wavelet-transformaties kunnen worden uitgevoerd m.b.v. het liftingschema. We hebben de details uitgewerkt van een gehele-getallen-wavelet-transformatie voor een bepaalde klasse van biorthogonale wavelets (Cohen-Daubechies-Feauveau).

## 4 Wavelet-gebaseerde beeldverwerking

### 4.1 Digitale beelden

In een computersysteem worden grijswaardenbeelden voorgesteld door een matrix. Elk matrixelement bepaalt de helderheid van een beeldpunt in het beeld. Kleurenbeelden bestaan uit meerdere matrices, één voor elke kleurencomponent (bvb. rood, groen en blauw): de kleur van een beeldpunt wordt nu bepaald door de corresponderende elementen in elke matrix.

### 4.2 Ruisverwijdering met wavelets

Ruisverwijdering is een belangrijke toepassing van wavelets.

Een wavelet-gebaseerde ruisverwijderingsmethode werkt als volgt: eerst wordt het signaal of beeld getransformeerd m.b.v. een discrete wavelet-transformatie, daarna worden de wavelet-coëfficiënten gemanipuleerd, gevolgd door een inverse wavelet-transformatie. Of een coëfficiënt gemanipuleerd wordt of niet hangt ervan af of hij beïnvloed is door ruis. Eerst moet men dus bepalen welke coëfficiënten ruis bevatten.

De meest voor de hand liggende methode gebruikt de absolute waarde van een wavelet-coëfficiënt als regulariteitsmaat: een kleine coëfficiënt bevat weinig informatie en dus hoofdzakelijk ruis, een grote coëfficiënt bevat veel informatie en weinig ruis. M.b.v. drempelmaskering kan men dus de kleine ruizige coëfficiënten verwijderen. Hiervan bestaan twee varianten: bij harde drempelmaskering worden enkel de kleine coëfficiënten vervangen door nul, bij zachte drempelmaskering wordt de grootte van de grote coëfficiënten verminderd met de drempelwaarde.

Meestal is zachte drempelmaskering te verkiezen omdat het een meer continue operatie is.

Voor de selectie van de drempelwaarde nemen we het minimum van de "veralgemeende kruisvalidatie"-functie, gebaseerd op het werk van Jansen [60, 24, 23, 25]. Deze methode is asymptotisch optimaal: voor voldoende grote beelden geeft ze dus zeer goede resultaten.

## 4.3 Schalering van beelden

Na één stap van de discrete wavelet-transformatie van een beeld is de laagfrequente subband een verkleinde versie (met factor 2) van het originele beeld. Men kan een wavelet-transformatie dus ook gebruiken om beelden te verkleinen met een factor $2^n$. Omwille van het filteren met een laag-doorlaatfilter treedt er geen vouwvervorming op.

Bijgevolg kan men beelden ook vergroten met een factor $2^n$ m.b.v. een inverse wavelet-transformatie, waarbij de wavelet-coëfficiënten van de toegevoegde hoogfrequente subbanden op nul gezet worden. Dit geeft betere resultaten dan eenvoudige beeldpuntduplicatie omdat er geen blokeffect optreedt.

Indien de schaalfactor geen macht van twee is voegen we een extra interpolatiestap toe [62].

## 4.4 Knip- en plakwerk

In het ruimtedomein is het zeer eenvoudig om een rechthoekig deel van een beeld uit te knippen. Ondanks de lokaliteit van wavelets is dit echter veel moeilijker in het wavelet-domein: de wavelet-transformatie is immers niet translatie-invariant. Vermits knip- en plakoperaties belangrijk zijn hebben we dit probleem omzeild, zoals aangegeven in Hoofdstuk 5.2.

## 4.5 Beeldcompressie

We gebruiken steeds meer digitale beelden in multimediatoepassingen. Bovendien worden digitale beelden steeds groter. Omwille van de grote opslag- en transmissiekost van digitale beelden is er een sterke behoefte aan beeldcompressie.

Omwille van hun goede decorrelatie-eigenschappen zijn wavelets een sleuteltechnologie voor beeldcompressie: ze laten goede compressieverhoudingen toe met behoud van een goede beeldkwaliteit [15, 5]. Vooral voor hoge compressieverhoudingen geven wavelet-gebaseerde methoden veel betere resultaten dan concurrerende technieken. Vergeleken met JPEG [59] vertonen ze geen storende blokeffecten.

De multiresolutie-eigenschappen van wavelets bieden een aantal extra mogelijkheden, zoals progressieve transmissie en inzoomen.

Beeldcompressietechnieken op basis van wavelets maken meestal gebruik van nulboomcodering [42, 39].

# 5 Ontwerp van de programmabibliotheek PIEFPAK

PIEFPAK (WAILI — Wavelets with Integer Lifting) is een programmabibliotheek voor wavelet-transformaties en wavelet-gebaseerde operaties op tweedimensionale beelden [57]. We ontwikkelden PIEFPAK in het kader van het project "Wavelet-Based Interactive Video Communication and Image Database Consulting".

De broncode van PIEFPAK is beschikbaar voor onderzoeksdoeleinden [56]. Momenteel is PIEFPAK in gebruik bij ca. 250 onderzoeksinstellingen.

## 5.1 Ontwerpcriteria

### Het liftingschema

PIEFPAK implementeert de tweedimensionale tensorproduct-wavelet-transformatie, gebaseerd op de gehele-getallenversie van het liftingschema. Beelden kunnen meerdere kanalen bevatten (bvb. RGB of YUV), eventueel elk met een andere (willekeurige) grootte. Alle berekeningen gebeuren met 16-bit invoer- en uitvoergegevens. Dit is voldoende voor toepassingen met 8-bit invoerbeelden.

### Cohen-Daubechies-Feauveau biorthogonale wavelets

We kozen voor een aantal wavelets uit de biorthogonale familie van Cohen-Daubechies-Feauveau omwille van hun eigenschappen [9]:

- Ze hebben een compacte drager zodat lokale beeldeigenschappen behouden blijven.
- De schaalfunctie is altijd symmetrisch, terwijl de wavelet-functie ofwel symmetrisch, ofwel antisymmetrisch is. Symmetrie is belangrijk voor beeldverwerking omdat de detectie van beeldeigenschappen dan niet afhankelijk is van de oriëntatie.
- Ze hebben dyadische filtercoëfficiënten ($z/2^n$, met $z \in \mathbf{Z}$ en $n \in \mathbf{N}$). Dit vergemakkelijkt de implementatie.

Natuurlijk kan de bibliotheek eenvoudig uitgebreid worden met andere wavelets, op voorwaarde dat men de beschikking heeft over een ontbinding in liftingstappen.

### Restricties voor de beeldgrootte

In het ideale geval bevatten digitale signalen $2^n$ monsters. Dit is het enige geval waarbij men $n$ opeenvolgende niveaus van een wavelet-transformatie kan uitvoeren, waarbij de laagfrequente en hoogfrequente subbanden op elk resolutieniveau hetzelfde aantal monsters tellen. Uitgebreid naar twee dimensies zijn ideale beelden vierkant, met zijde $2^n$ beeldpunten. In de praktijk zijn beelden rechthoekig, met willekeurige grootte, wat de implementatie heel wat complexer maakt. PIEFPAK ondersteunt willekeurige beeldgroottes.

### Beelden en hun wavelet-transformatie

Het uitvoeren van wavelet-transformaties op beelden vergt interactie tussen een aantal componenten (C++-klassen):

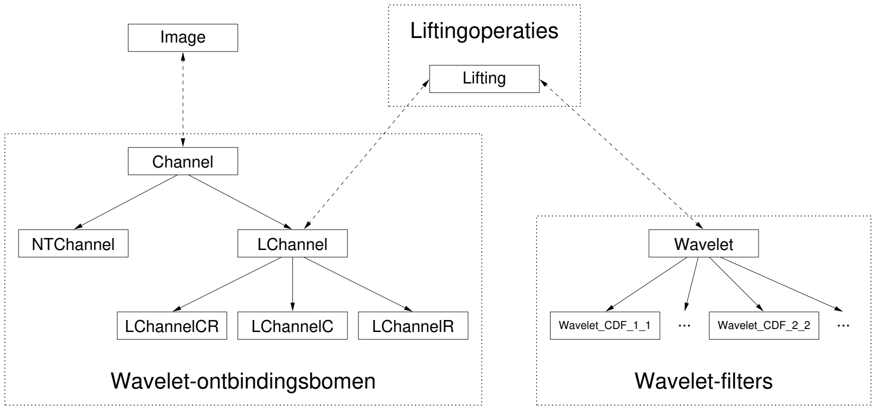
**Image** Een *Image* is een beeld, bestaande uit een aantal onafhankelijke kanalen.

**Channel** Een *Channel* is een kanaal van een beeld en vormt het basisblok van PIEFPAK. Een kanaal kan niet-getransformeerd zijn (een *NTChannel*) of wavelet-getransformeerd (een *LChannel*). Omwille van de multiresolutie bevat een LChannel een aantal subbanden, welke opnieuw Channels zijn. Het aantal subbanden hangt af van het type transformatie (kolommen en rijen (*LChannelCR*), alleen kolommen (*LChannelC*) of alleen rijen (*LChannelR*)).

**Wavelet** Een *Wavelet* stelt de wavelet-filters en liftingstappen voor horende bij een specifieke wavelettransformatie. Verschillende soorten wavelet-transformaties worden geïmplementeerd door overerving. Een aantal transformaties uit de familie van Cohen-Daubechies-Feauveau zijn beschikbaar, maar men kan gemakkelijk andere wavelet-transformaties toevoegen.

**Lifting** Een *Lifting*-object voert liftingstappen uit. Dankzij de object-georiënteerde aard van C++ kan men eventueel geoptimaliseerde versies voor specifieke wavelet-transformaties toevoegen.

De relaties tussen de belangrijkste componenten van WAILI worden getoond in onderstaande figuur:




## 5.2 Wavelet-gebaseerde bewerkingen

PIEFPAK bevat een aantal beeldverwerkingsoperaties gebaseerd op wavelets.

### Knip- en plakoperaties op wavelet-getransformeerde beelden

De wavelet- en schaalfuncties zijn gelokaliseerd in zowel het ruimtelijk als het frequentiedomein. Als men een rechthoekig gedeelte $B$ wil uitknippen uit een beeld $A$ kan men gebruik maken van de ruimtelijk lokaliteit van de basisfuncties. Op elk resolutieniveau zijn er dus slechts een beperkt aantal extra wavelet-coëfficiënten nodig omwille van de overlappende basisfuncties aan de randen van $B$. M.b.v. deze techniek implementeerden we op efficiënte wijze knip- en plakoperaties op wavelet-getransformeerde beelden.

De wavelet-transformatie is echter niet translatie-invariant. Om dit probleem te omzeilen introduceerden we globale coördinaten voor beelden. Op elk resolutieniveau bepaalt de pariteit van de coördinaten of een beeldpunt tijdens de splitsingsstap van het liftingschema zal terechtkomen in de “even” of in de “oneven” partitie. Hierdoor is het mogelijk knip- en plakoperaties uit te voeren zonder het deelbeeld te hertransformeren, op voorwaarde dat de coördinaten van de (deel)beelden op voorhand bekend zijn. Dit is het geval in een aantal GIS-toepassingen.

### Het schaleren van beelden

Een wavelet-transformatie kan gebruikt worden om beelden te verkleinen met een factor $2^n$; haar inverse om ze te vergroten met een factor $2^n$. Voor andere schaalfactoren maken we gebruik van een extra interpolatiestap.

### Ruisverwijdering

Zachte drempelmaskering met een drempelwaarde bepaald door veralgemeende kruisvalidatie is een efficiënte en performante ruisverwijderingstechniek.

## 5.3 Implementatieaspecten

### "In place"-ordening versus Mallat-ordening

Het liftingschema laat toe om een wavelet-transformatie "in place" te berekenen. Dit heeft als voordeel dat minder geheugen nodig is, echter het resulteert in een minder gunstige ordening van de waveletcoëfficiënten: coëfficiënten uit dezelfde subband zitten verspreid in het geheugen.

De tweedimensionale tensorproduct-wavelet-transformatie van een matrix houdt in dat zowel de kolommen als de rijen van de matrix worden getransformeerd m.b.v. de ééndimensionale transformatie. Als de waveletcoëfficiënten georganiseerd worden in het geheugen per subband krijgt men de "Mallat"-ordening [35]. Deze ordening is interessant indien men gebruik wenst te maken van de multiresolutie-eigenschappen van een wavelet-transformatie. Voor PIEFPAK verkozen we dus de Mallat-ordening boven de "in place"-ordening.

### Normalisatie van de wavelet-coëfficiënten

In het ideale geval is de normalisatiefactor $K_1$ (zie Vergelijking 1) voor de laagfrequente gegevens altijd $\sqrt{2}$. In dat geval blijft het dynamisch bereik van de waveletcoëfficiënten in de laagfrequente subbanden immers steeds gelijk.

Natuurlijk moet men rekening houden met de accumulatie van de normalisatiefactoren voor opeenvolgende resolutieniveaus. De werkelijke waarde $x$ van een waveletcoëfficiënt kunnen we schrijven als $x = m \cdot (\sqrt{2})^e$, met $m$ de coëfficiënt na transformatie en $e$ een "shift".

## 5.4 Samenvatting van de functionaliteit van PIEFPAK

Samengevat biedt PIEFPAK de volgende functionaliteit:

**Wavelet-transformaties** Wavelet-transformaties gebaseerd op de gehele-getallenvariant van het liftingschema. Men kan zowel de kolommen als de rijen van een beeld transformeren, of enkel de kolommen of rijen.

**Beeldverwerkingsoperaties** Knip- en plakoperaties, ruisverwijdering op basis van veralgemeende kruisvalidatie, schalering, zachte en harde drempelmaskering, randverbetering.

**Basisbewerkingen** Optellen en aftrekken van beelden.

**Hulpoperaties** Berekening van de piek signaal-ruisverhouding (PSNR), eerste orde Shannon-entropie, correlatiematrices, minima en maxima in een beeld, histogrammen, veralgemeende kruisvalidatiewaarden.

**Beeldinvoer en -uitvoer** Invoer van beelden in PNM- en TIFF-formaat en uitvoer van beelden in PNM-formaat.

PIEFPAK bevat tevens een interactief voorbeeldprogramma dat het gebruik van een aantal van deze operaties (bvb. wavelet-transformaties, drempelmaskering en ruisverwijdering) demonstreert. Broncode in C++ voor een eenvoudig compressievoorbeeld is beschikbaar in Appendix B.

De C++-broncode van PIEFPAK, met inbegrip van de voorbeelden, telt ca. 12500 programmalijnen.

### 5.5 Evaluatie van het compressiepotentieel

We evalueerden het compressiepotentieel van een aantal wavelet-transformaties m.b.v. een eenvoudige compressietechniek: harde drempelwaardemaskering. Als testbeeld namen we een 8-bit grijswaardenbeeld van $768 \times 768$ beeldpunten (een luchtfoto van onze campus). Het beeld werd getransformeerd tot op 6 ontbindingsniveaus m.b.v. de gehele-getallen-wavelet-transformatieroutines in WAILI.

De compressieverhouding werd geschat op basis van de eerste orde Shannon-entropie [41] van de wavelet-coëfficiënten na drempelwaardemaskering. De entropie is een maat voor de hoeveelheid bits die nodig zijn om deze coëfficiënten te encoderen. De beeldkwaliteit na reconstructie werd gemeten m.b.v. de piek signaal-ruisverhouding. We vergeleken zowel met het originele beeld als met een door WAILI ontruisde versie van het beeld. Dit heeft zin aangezien het originele beeld ruis bevat en omdat ruisverwijdering ook gebruik maakt van drempelwaardemaskering: bij lage compressieverhoudingen ziet het beeld na reconstructie er beter uit dan het origineel omdat het minder ruis bevat. De CDF $(2,2)$-wavelets blijken de beste resultaten te geven.

## 6 Grootschalige beeldverwerking

In toepassingen zoals Geografische Informatiesystemen (GIS) worden beeldgroottes uitgedrukt in gigabytes (GB) en soms zelfs in terabytes (TB). Een kleurenbeeld van België waarbij elk beeldpunt overeenkomt met één vierkante meter verbruikt 90 GB opslagcapaciteit. Het is duidelijk dat zulke beelden moeten worden opgeslagen en verwerkt in gecomprimeerde vorm. Dit is mogelijk m.b.v. wavelet-gebaseerde technieken.

We beschrijven een uitbreiding van de programmabibliotheek PIEFPAK naar blokgebaseerde verwerkingen [55]. Dit laat eventueel een parallelle implementatie toe.

## 6.1 Zeer grote beelden en tegeling

We definiëren een *zeer groot beeld* als een beeld dat niet past in het geheugen van de computer waarmee het wordt verwerkt.

### Tegeling

Zulke grote beelden moeten worden opgedeeld in blokken (tegeling). Aangezien wavelets lokaal zijn in het ruimtedomein heeft men slechts een beperkt deel van een invoerbeeld nodig om een deel van de wavelet-coëfficiënten te berekenen. Vermits niet alle blokken tegelijkertijd hoeven aanwezig te zijn in het hoofdgeheugen kan men dus op efficiënte wijze grote beelden verwerken. Bovendien kan de verwerking van de afzonderlijke blokken worden verdeeld over meerdere processoren.

In PIEFPAK verdelen we elke subband apart in blokken. Dit heeft de volgende gevolgen voor wavelet-transformaties:

**Splitsing (luie wavelet-transformatie)** Om een blok op te splitsen in "even" en "oneven" coëfficiënten zijn geen andere blokken nodig. In deze stap zijn er dus geen problemen.

**Liftingstappen** Om een liftingstap te berekenen van een randcoëfficiënt zijn een paar coëfficiënten — een aantal afhankelijk van de gekozen wavelet-transformatie — van een naburig blok nodig. In een parallelle implementatie vereist dit dus communicatie tussen de processoren.

### Blokbeheer

We voegden ondersteuning voor beelden opgedeeld in blokken toe aan de bibliotheek PIEFPAK, die we tevoren hadden ontwikkeld. Dit gebeurde op een manier die zo goed als transparant is voor de gebruiker.

De verantwoordelijkheid voor het beheer van de blokken ligt bij een blokbeheerder: als een blok nodig is moet men eerst de blokbeheerder vragen om dit blok op te halen (bvb. van een schijf of via een netwerk). Als een blok wordt gewijzigd moet het nadien worden geactualiseerd op het opslagapparaat.

### Implementatie in PIEFPAK

We implementeerden blokgebaseerde wavelet-transformaties door twee subklassen af te leiden van de NTChannel-klasse: *NTChannelR* voor normale subbanden, bestaande uit slechts één blok, en *NTChannelB* voor subbanden bestaande uit meerdere blokken. Bij elk beeld hoort een *BlockMgr* object dat zorgt voor het beheer van de individuele blokken.

Deze uitbreiding deed de broncode van PIEFPAK groeien van ca. 12500 tot ca. 21000 lijnen C++-code.

## 6.2 Beeldverwerking op getegelde beelden

Een aantal wavelet-gebaseerde beeldverwerkingsoperaties kan men nog steeds uitvoeren op beelden die in blokken zijn verdeeld.

#### Blokgebaseerde ruisverwijdering

PIEFPAK gebruikt zachte drempelmaskering om additieve stationaire ruis te verwijderen. De drempelwaarde wordt per subband gekozen m.b.v. veralgemeende kruisvalidatie [57]. Voor getegelde beelden passen we deze techniek toe op de aparte blokken: de drempelwaarde kan dus verschillend zijn per blok. Dit is triviaal te parallelliseren.

Omdat enkel de hoogfrequente subbanden worden aangepast kan men de randen van de blokken visueel niet onderscheiden. Deze methode blijkt zelfs betere resultaten te geven dan met een drempelwaarde per subband, wat verklaard kan worden door de aanpassing aan de lokale eigenschappen van het beeld.

#### Beeldcompressie

Wavelet-gebaseerde compressietechnieken maken dikwijls gebruik van nulboomcodering. Deze techniek is echter niet bruikbaar voor getegelde beelden omdat de wavelet-coëfficiënten in een nulboom verspreid zijn over een veelvoud aan blokken. Een mogelijke oplossing is een strategie bestaande uit twee niveaus: nulboomcodering voor de grofste resolutieniveaus, waar elke subband slechts één blok bevat, en een blokgebaseerde encodering voor de fijne niveaus. Indien de blokgrootte voldoende groot is bevatten de blokken behorende tot de fijne niveaus hopelijk voldoende kleine coëfficiënten om efficiënt te worden gecomprimeerd. Ook dit is eenvoudig te parallelliseren.

#### Knip- en plakwerk

PIEFPAK implementeert knip- en plakoperaties op getegelde wavelet-transformaties op een transparante en efficiënte wijze. Omwille van de lokaliteit van wavelets zijn slechts een beperkt aantal blokken nodig om deze operaties uit te voeren.

#### Een GIS-toepassing

In GIS-toepassingen wordt een groot beeld meestal samengesteld uit een aantal kleinere luchtfoto's. Een wavelet-getransformeerde getegelde versie van zo'n beeld kunnen we op volgende wijze creëren:

1. Creëer een leeg zeer groot beeld (d.w.z. een egaal grijs beeld).

2. Voor alle luchtfoto's $i$

(a) Laad luchtfoto $i$ in het computergeheugen.

(b) Transformeer luchtfoto $i$, rekening houdende met de absolute coördinaten en de resolutie van de foto.

(c) Plak de wavelet-getransformeerde foto $i$ in het zeer grote beeld.

Hierna heeft men een wavelet-getransformeerd beeld dat identiek is aan de wavelet-transformatie van de samenvoeging van de kleine luchtfoto's.

Daarna kan men dankzij de multiresolutie-eigenschappen van wavelets een willekeurig rechthoekig deelbeeld selecteren op een willekeurig dyadisch resolutieniveau.

### 6.3 Transformatieresultaten

We analyseren de (sequentiële) performantie van onze wavelet-transformatiecode en maken een aantal bedenkingen voor een parallelle implementatie. Een parallelle implementatie vereist communicatie van de randcoëfficiënten van de blokken. Deze communicatie is echter verwaarloosbaar t.o.v. de benodigde rekentijd. We concluderen dat een parallelle implementatie efficiënt kan zijn indien het aantal wavelet-transformatieniveaus klein is en het aantal processoren niet te groot in verhouding tot de beeldgrootte.

## 7 De rood-zwart-wavelet-transformatie

Klassieke ééndimensionale wavelet-transformaties kunnen worden uitgebreid naar meerdere dimensies m.b.v. tensorproducten, met een separabele transformatie als gevolg. Dit heeft als nadeel dat men een anisotropie introduceert in de wavelet-ontbinding: in het tweedimensionale geval worden de horizontale en verticale eigenschappen van de invoergegevens bevoordeeld; anders-georiënteerde eigenschappen worden niet zo makkelijk gedetecteerd. Niet-separabele wavelets zijn een mogelijkheid, tweede-generatie-wavelets zijn een andere optie.

We presenteren een nieuw soort tweede-generatie-wavelets op een rechthoekig rooster (een quincunx rooster [30]), geconstrueerd m.b.v. het liftingschema.

### 7.1 Rood-zwart en blauw-geel

In Hoofdstuk 3.1 legden we de basisprincipes van het liftingschema uit op een abstracte manier. Deze principes kan men ook toepassen voor hoger-dimensionale gegevensverzamelingen. Wij passen ze toe op een rechthoekig rooster, gebruik makende van een tweedimensionaal analoog van de CDF $(2,2)$-wavelets.

We lieten ons inspireren door de welbekende rood-zwart Gauss-Seidel-techniek voor de iteratieve oplossing van lineaire systemen en splitsen het rooster $\lambda_{j+1}$ op volgens een schaakbordpatroon, in "rode" en "zwarte" vierkantjes. Vervolgens voorspellen we de zwarte verzameling $\gamma_j$ aan de hand van het gemiddelde van de vier meest nabije buren uit de rode verzameling $\lambda_j$. Daarna passen we de rode verzameling $\lambda_j$ aan m.b.v. de zwarte verzameling $\gamma_j$ om de gemiddelde waarde te behouden. Hierna is de rode verzameling een laagfrequente benadering van de originele gegevens, terwijl de zwarte verzameling de detailinformatie bevat.

Voor het daaropvolgende resolutieniveau $j-1$ gaan we verder met de rode verzameling uit het vorige niveau. Omdat de gegevens in deze verzameling georganiseerd zijn volgens de diagonalen kunnen we niet meer exact dezelfde operatie uitvoeren als op het vorige niveau. In de plaats daarvan roteren we het rooster over 45° en splitsen de rode verzameling $\lambda_j$ in een blauwe verzameling $\lambda_{j-1}$ en een gele verzameling $\gamma_{j-1}$. Dezelfde predictie- en bijsturingsoperatoren worden nu uitgevoerd op de blauwe en gele vierkantjes. De resulterende blauwe verzameling vormt terug een horizontaal-verticaal patroon.

De hele multiresolutie-boom bestaat dus uit een opeenvolging van afwisselend een rood-zwart- en een blauw-geel-opsplitsing. Een omkeerbare gehelegetallenversie is zeer eenvoudig af te leiden door het toevoegen van een afrondingsoperatie in elke liftingstap.

We maken gebruik van eenvoudige predictie- en bijsturingsoperatoren gebaseerd op de ééndimensionale CDF $(2,2)$-wavelets. Dit geeft wavelets met twee nulmomenten. Meer complexe predictie- en bijsturingsoperatoren genereren wavelets met andere eigenschappen.

Het principe van rood-zwart-wavelets is niet beperkt tot planaire quincunxroosters: het kan bvb. worden uitgebreid tot driehoekige of hexagonale regelmatige roosters, of tot roosters op een boloppervlak.

## 7.2 Eigenschappen van de rood-zwart-wavelet-transformatie

De rood-zwart-wavelet-transformatie heeft de volgende eigenschappen:

**2D** Het is een echte tweedimensionale transformatie die niet-separabele wavelets genereert.

**Isotropie** Ze is minder anisotroop dan tensorproduct-wavelets op hetzelfde rooster.

**Symmetrie** Alle basisfuncties hebben vier symmetrie-assen.

**Biorthogonaliteit** De rood-zwart-wavelets zijn biorthogonaal.

**Zachtverlopendheid** De rood-zwart-wavelets hebben twee nulmomenten.

**Implementatie** De complexiteit van de transformatie is $O(n)$, zoals geldt voor elke wavelet-transformatie.

### 7.3 Vergelijking met de CDF $(2,2)$ tensorproduct-wavelets

We vergeleken de rood-zwart-wavelets met de CDF $(2,2)$ tensorproduct-wavelets in een ruisverwijderingstoepassing. In beide gevallen ging het om de gehele-getallenvariant van de transformatie. De tensorproduct-wavelets gaven marginaal betere resultaten, behalve voor fijne randen die niet horizontaal, verticaal of diagonaal georiënteerd zijn. Zulke randen werden beter behouden door de rood-zwart-wavelets.

## 8 Experimenten met een wavelet-gebaseerde APOD

In vele deeldomeinen van toegepaste wetenschappen onderzoekt men het gedrag van grootschalige dynamische systemen, beschreven door partiële differentiaalvergelijkingen. Omwille van de hoogdimensionaliteit van deze systemen wenst men eerst de complexiteit terug te brengen.

De orthogonale eigenontbinding (POD — Proper Orthogonal Decomposition) of Karhunen-Loève-ontbinding (KLT) is een krachtig hulpmiddel voor modelreductie. Uitgaande van een verzameling oplossingsprofielen (verkregen via experimenten of berekeningen), momentopnames genoemd, berekent men een "optimale" basis van eigenmodes voor het systeem. Deze basis kan men later gebruiken om een laagdimensionaal model op te stellen. De constructie van een orthogonale eigenontbinding is echter een kostelijke operatie, evenals de projectie van het oorspronkelijke probleem op de basis gevormd door de eigenmodes.

### 8.1 De orthogonale eigenontbinding

De orthogonale eigenontbinding berekent een "ideale" orthonormale basis voor een gegeven set van $S$ lineair onafhankelijke momentopnames $X_i$ (vectoren met lengte $N$, waarbij $S \ll N$). Met "ideaal" bedoelen we de basis die de beste (kleinste kwadraten) benadering van de momentopnames geeft met zo weinig mogelijk basisvectoren. De basisvectoren zijn zo geordend dat de beste benadering met $i$ vectoren bekomen wordt door de eerste $i$ basisvectoren te nemen.

Veronderstel dat het gemiddelde van de momentopnames nul is, dan wordt deze basis gegeven door de eigenvectoren $E_k$ van de $N \times N$ covariantiematrix

$$C = \frac{1}{S} \sum_{i=0}^{S-1} X_i X_i^T .$$

De eigenwaarden $\lambda_k$ zijn een indicatie voor het relatief belang van de bijhorende eigenvector $E_k$: we sorteren de eigenvectoren dus in dalende volgorde van de grootte van de eigenwaarden. De eigenvectoren worden ook wel eigenmodes genoemd.

Als we de momentopnames eerst normaliseren:

$$X_i \leftarrow \frac{X_i}{||X_i||\sqrt{S}}$$

is de som van de eigenwaarden gelijk aan 1. In dat geval bevat eigenmode $E_k$ $100 \times \lambda_k\%$ van de totale energie.

De constructie van de covariantiematrix $C$ vereist $O(SN^2)$ operaties. De complexiteit kan echter gevoelig gereduceerd worden m.b.v. de "methode van de momentopnames" [45]. Als we gebruik maken van de eigenschap dat de eigenvectoren kunnen geschreven worden als een unieke lineaire combinatie van de momentopnames, dan kunnen we het oorspronkelijk eigenwaardenprobleem met matrix $C$ vervangen door een ander eigenwaardenprobleem met matrix

$$C^* = \left[c^*_{i,j}\right] = \left[X_i^T X_j\right], \qquad i,j = 0,\ldots,S-1.$$

De constructie van de matrix $C^*$ vergt slechts $O(NS^2)$ operaties.

## 8.2 De benaderende orthogonale eigenontbinding

Zelfs m.b.v. de methode van de momentopnames blijft de berekening van de POD een dure operatie. De basisidee achter de benaderende POD (APOD — Approximate Proper Orthogonal Decomposition) bestaat erin de momentopnames vooraf te decorreleren m.b.v. wavelets en de klassieke POD te berekenen voor de gecomprimeerde momentopnames. We hopen hiermee nog steeds een voldoende goede benadering van de echte eigenmodes te bekomen met gevoelig minder rekenwerk.

De momentopnames $X_i$ worden dus getransformeerd m.b.v. een decorrelerende transformatie (wij gebruiken een wavelet-packet-transformatie) en gecomprimeerd door enkel de belangrijkste coëfficiënten in de wavelet-packet-basis te behouden. Vervolgens berekenen we de POD voor de gecomprimeerde momentopnames, wat ons gecomprimeerde eigenmodes en eigenwaarden geeft. Na decompressie en een inverse wavelet-packet-transformatie krijgen we dus een (hopelijk goede) benadering van de echte eigenmodes. Indien de decorrelerende transformatie niet orthogonaal is maar biorthogonaal, zijn er niet alleen primale maar ook duale benaderende eigenmodes.

De APOD kan gebruikt worden als alternatief voor de echte POD om de hoeveelheid rekenwerk te verminderen en/of om een laagdimensionaal model te construeren voor het originele probleem, gebaseerd op de benaderende eigenmodes.

We kozen voor een wavelet-packet-transformatie als decorrelerende transformatie aangezien deze dichter bij de echte POD staat dan een gewone wavelet-transformatie omwille van de extra vrijheidsgraden bij de keuze van de basisfuncties.

**Ontbinding in een basis van benaderende eigenmodes**

Een willekeurige momentopname kan benaderd worden door een lineaire combinatie van de genormaliseerde benaderende eigenmodes. De coëfficiënten van deze ontbinding worden berekend m.b.v. inwendige producten met de (duale) benaderende eigenmodes. Bovendien kunnen deze coëfficiënten ook bekomen worden in het wavelet-domein, als inwendige producten met de gecomprimeerde (duale) eigenmodes.

Welke aanpak men volgt hangt af van een aantal factoren. Indien de momentopnames vierkant zijn met grootte $\sqrt{N} \times \sqrt{N}$, de gecomprimeerde momentopnames $n$ elementen tellen en we $s$ eigenmodes behouden, dan is de complexiteit:

- in het originele domein: $s$ inwendige producten in $\mathbf{R}^N$ $(O(sN))$,
- in het wavelet-domein: $S$ wavelet-packet-transformaties $(O(SN \log \sqrt{N}))$ en $s$ inwendige producten in $\mathbf{R}^n$ $(O(sn))$.

## 8.3 Resultaten

We hebben het gebruik van de APOD geëvalueerd aan de hand van de analyse van het dynamisch gedrag van de discretisatie van een stelsel partiële differentiaalvergelijkingen, horende bij een reactie-diffusie-systeem (de ééndimensionale Brusselator [22]).

We kozen voor symmetrische biorthogonale wavelets uit de familie van Cohen-Daubechies-Feauveau met 2 nulmomenten [9]. Als kostcriterium voor de selectie van de beste wavelet-packet-basis namen we het logaritme van de energie-inhoud $(\sum_i \log x_i^2)$.

Uitgaande van een aantal verzamelingen van momentopnames voor een aantal verschillende parameterwaarden, bekomen m.b.v. tijdsintegratie, construeerden we een laagdimensionaal model voor het reactie-diffusie-systeem. Deze momentopnames bevatten een aantal intrinsieke eigenschappen van het model, bvb. periodieke oplossingen. De magnitude van een eigenwaarde geeft aan hoe belangrijk de bijhorende eigenmode is. We krijgen dus een goede benadering als we de eigenmodes horende bij de grootste eigenwaarden behouden. De verdeling van de eigenwaarden geeft aan hoe goed de momentopnames gedecorreleerd waren. De som van de 4 grootste eigenwaarden is steeds groter dan 0.99, zowel voor de POD als voor de APOD. Ook als we de momentopnames ontbinden in de basis van de 4 belangrijkste eigenmodes zien we dat deze ontbinding bijna altijd 99% van de totale energie in de momentopnames opvangt, zowel voor de POD als voor de APOD.

Het gedrag van het dynamisch systeem wordt duidelijk bij analyse van de coëfficiënten in de basis van de eigenmodes. De periodiciteit in de momentopnames is duidelijk zichtbaar in het verloop van de coëfficiënt horende bij de belangrijkste eigenmode. We testten het laagdimensionaal model aan de hand van een verzameling “vreemde” momentopnames die niet gebruikt werden voor de constructie van het model. Deze momentopnames tonen een periodieke oplossing inclusief overgangsverschijnselen. Ook dit gedrag is duidelijk zichtbaar in het verloop van de coëfficiënten.

De complexiteit van de APOD is $O(SN\log N) + O(S^2N/C) + O(S^3)$, met $S$ het aantal momentopnames, $N$ de grootte van de momentopnames en $C$ de compressieverhouding, vergeleken met $O(NS^2) + O(S^3)$ voor de POD zelf. Indien $S$ van dezelfde grootte-orde is als $\log N$ geeft de APOD dus geen rekenvoordeel. Merk op dat de factor $\log N$ enkel geldig is voor het ééndimensionale geval, voor het $d$-dimensionale geval wordt dit $\log \sqrt[d]{N}$. Deze factor daalt sterk voor stijgende dimensie, dus voor hogere dimensies kunnen we wel winst maken.

# 9 Conclusies en suggesties voor toekomstig onderzoek

## 9.1 Wavelet-transformaties

De wavelet-transformatie laat toe om gegevens te decorreleren omdat wavelets gelokaliseerd zijn in zowel ruimte/tijd als schaal/frequentie en omdat ze zijn gebaseerd op een multiresolutie-analyse. Het liftingschema is een efficiënte implementatie van een wavelet-transformatie-algoritme. Verder kan het eenvoudig worden omgezet om gehele-getallengegevens te verwerken. De inverse transformatie is triviaal te vinden en voorziet altijd in een perfecte reconstructie van de invoergegevens. We hebben de details van gehele-getallentransformaties uitgewerkt voor een bepaalde klasse van biorthogonale wavelets (Cohen-Daubechies-Feauveau).

Het liftingschema kan niet alleen worden gebruikt voor de berekening van klassieke biorthogonale wavelet-transformaties; het is eveneens een modelschema voor het ontwerp van zogenaamde tweede-generatie-wavelets. Dit zijn wavelets die niet eenvoudigweg afgeleid zijn van de translaties en dilaties van een ééndimensionale functie. Als voorbeeld hiervan creëerden we wavelets op een rechthoekig rooster, gebaseerd op een rood-zwart-blokschema. De resulterende rood-zwart-wavelets zijn niet-scheidbare tweedimensionale wavelets, in tegenstelling tot klassieke tensorproduct-wavelets. In een ruisverwijderingstoepassing toonden we aan dat ze minder anisotroop zijn en de horizontale en verticale dimensies niet zo sterk bevoordelen. De idee achter rood-zwart-wavelets kan eenvoudig worden toegepast op hogere dimensies en meer complexe geometrieën.

## 9.2 Een programmabibliotheek voor wavelet-gebaseerde beeldverwerking

We ontwikkelden de C++-programmabibliotheek PIEFPAK (WAILI — Wavelets with Integer Lifting), dewelke wavelet-transformaties en wavelet-gebaseerde beeldverwerkingsoperaties implementeert. De gehele-getallenversie van het liftingschema vormt het fundament van PIEFPAK.

De bibliotheek implementeert verscheidene wavelet-transformaties uit de biorthogonale familie van Cohen-Daubechies-Feauveau op beelden met willekeurige afmetingen. Men kan eenvoudig andere wavelet-transformaties toevoegen op voorwaarde dat men beschikt over een ontbinding in liftingstappen.

PIEFPAK voorziet in een aantal beeldverwerkingsoperaties op wavelet-getransformeerde beelden. Zachte drempelmaskering m.b.v. gegeneraliseerde kruisvalidatie is een efficiënte en goed presterende methode voor ruisverwijdering. Andere belangrijke bewerkingen zijn knip- en plakoperaties: dankzij de ruimtelijke lokaliteit van wavelets kan men deelbeelden uitknippen uit of plakken in een groter wavelet-getransformeerd beeld zonder ze te hertransformeren, waarbij slechts een zeer beperkte deelverzameling van wavelet-coëfficiënten wordt beïnvloed.

PIEFPAK werd ook uitgebreid voor de behandeling van zeer grote beelden. Door de individuele wavelet-subbanden in blokken te verdelen op een voor de gebruiker transparante manier, kunnen grote beelden worden getransformeerd en verwerkt. De blokken worden verwerkt onder supervisie van een blokbeheerder. De opdeling in blokken heeft eveneens een gunstige invloed op de prestaties van de ruisverwijdering: vermits ruisverwijdering nu gebeurt op onafhankelijke blokken past ze zich beter aan aan de lokale eigenschappen van het beeld.

Dit ontwerp biedt goede mogelijkheden voor parallellisatie aangezien de meeste bewerkingen slechts communicatie van een smalle strook wavelet-coëfficiënten vereisen (bvb. liftingstappen), of zelfs helemaal geen communicatie (bvb. ruisverwijdering).

## 9.3 Een benaderende POD m.b.v. wavelet-packets

Dankzij haar uitstekende decorrelatie-eigenschappen vormt een wavelet-packet-transformatie een goede benadering van de theoretisch ideale decorrelatiemethode, de orthogonale eigenontbinding (Proper Orthogonal Decomposition — POD). De complexiteit van een systeem ter analyse kan dramatisch worden gereduceerd door de gegevens te comprimeren in het wavelet-packet-domein.

Om de voorafberekende oplossingsprofielen van een complex dynamisch systeem te analyseren gebruikten we de benaderende POD (Approximate POD — APOD). We deden experimenten met een reactie-diffusie-systeem dat wordt beschreven door partiële differentiaalvergelijkingen. De APOD liet een goede reconstructie van de originele gegevens toe, waarbij slechts weinig eigenmodes nodig waren. Zodoende werd de laagdimensionale natuur van ons dynamisch systeem onthuld. De APOD kan worden gebruikt om een laagdimensionaal model te construeren voor het originele probleem, wat een snelle en eenvoudige analyse van het dynamisch systeem toelaat.

## 9.4 Suggesties voor toekomstig onderzoek

We suggereren voorlopig een tweetal richtingen voor toekomstig onderzoek:

1. Moderne microprocessoren bevatten extra instructies voor de efficiënte parallelle uitvoering van een aantal operaties die zeer nuttig zijn voor multimedia toepassingen. Het moet mogelijk zijn om m.b.v. deze instructies een zeer goed geoptimaliseerde versie van de gehele-getallen-wavelet-transformatie te implementeren en deze in PIEFPAK te integreren.

2. De uitbreiding van PIEFPAK voor de behandeling van zeer grote beelden opgedeeld in blokken werd nog niet geparallelliseerd. Een aanpassing van de bibliotheek voor een parallelle uitvoering zou een interessant onderwerp voor verder onderzoek zijn.

# Contents

**Preface** v

**Nederlandse Samenvatting** vii

**Contents** xxxix

**Notations and Abbreviations** xliii

**List of Figures** xlv

**List of Tables** xlix

**1 Introduction** 1
1.1 Overview of Wavelet Research . . . 1
1.2 Overview of the Thesis . . . 2
1.3 Situation of the Research . . . 3

**2 Wavelets** 5
2.1 What are Wavelets? . . . 5
2.2 Why wavelets? . . . 6
2.3 Wavelets and Multi-Resolution Analysis . . . 6
2.3.1 Multi-Resolution Analysis . . . 6
2.3.2 Wavelet Functions . . . 7
2.3.3 The Fast Wavelet Transform (FWT) . . . 8
2.3.4 Orthogonal Wavelets . . . 9
2.3.5 Biorthogonal Wavelets . . . 10
2.3.6 The Wavelet Decomposition Tree . . . 11
2.3.7 Higher Dimensions . . . 12
2.4 The Wavelet Transform and Translation Invariance . . . 13
2.5 The Redundant Wavelet Transform . . . 13
2.6 Wavelet Packets . . . 14

**3 The Lifting Scheme** **19**
3.1 Introduction . . . 19
3.2 Predict and Update . . . 20
3.2.1 The Inverse Transform . . . 21
3.2.2 Example: Linear Prediction . . . 21
3.2.3 Flexibility . . . 22
3.3 Unifying the Two Paradigms . . . 22
3.3.1 The Filter Bank Algorithm . . . 22
3.3.2 Polyphase Representations . . . 23
3.3.3 Primal and Dual Lifting . . . 25
3.3.4 Decomposition into Lifting Steps . . . 26
3.3.5 Advantages of the Lifting Scheme . . . 28
3.4 The Integer Wavelet Transform . . . 29
3.4.1 Integer Transforms . . . 30
3.4.2 Rational Lifting Coefficients . . . 30
3.4.3 Normalization . . . 33
3.5 Boundary Treatment . . . 33
3.5.1 Classical Extension Methods . . . 33
3.5.2 Signal Extension with Lifting . . . 36
3.6 Conclusions . . . 37

**4 Wavelet-Based Image Processing** **39**
4.1 Digital Images . . . 39
4.1.1 Discretization . . . 39
4.1.2 Quantization . . . 40
4.1.3 Color images . . . 40
4.1.4 Generalization . . . 41
4.1.5 $L^*a^*b^*$ . . . 41
4.2 Wavelet Denoising . . . 41
4.2.1 Thresholding . . . 41
4.2.2 Threshold selection . . . 42
4.2.3 Integer Minimization Problem . . . 43
4.3 Image Scaling . . . 44
4.4 Crop and Merge . . . 47
4.5 Image Compression . . . 47
4.5.1 Why Compression? . . . 47
4.5.2 Compression Techniques . . . 48
4.5.3 Features of Image Compression using Wavelets . . . 50
4.5.4 Zerotree Encoding . . . 50
4.6 Conclusions . . . 51

**5 Design of the Software Library WAILI** **53**

5.1 Design Criteria . . . . . . . . . . . . . . . . . . . . . . . . . . 53
5.2 WAILI: Wavelets with Integer Lifting . . . . . . . . . . . . . . . 54
5.2.1 The Lifting Scheme . . . . . . . . . . . . . . . . . . . . . 54
5.2.2 Cohen-Daubechies-Feauveau Biorthogonal Wavelets . . . 54
5.2.3 Image Size Constraints . . . . . . . . . . . . . . . . . . . 55
5.2.4 Images and Wavelet Transforms . . . . . . . . . . . . . . 56
5.3 Wavelet-Based Operations . . . . . . . . . . . . . . . . . . . . . 58
5.3.1 Crop and Merge on Wavelet-Transformed Images . . . . . 58
5.3.2 Scaling of Images . . . . . . . . . . . . . . . . . . . . . . 59
5.3.3 Noise Reduction . . . . . . . . . . . . . . . . . . . . . . 59
5.4 Implementation Issues . . . . . . . . . . . . . . . . . . . . . . . 60
5.4.1 In-Place Transform . . . . . . . . . . . . . . . . . . . . . 60
5.4.2 The Two-Dimensional Wavelet Transform: Mallat Ordering 60
5.4.3 Transform Order . . . . . . . . . . . . . . . . . . . . . . 61
5.4.4 Normalization of the Wavelet Coefficients . . . . . . . . . 62
5.5 Summary of the Functionality of WAILI . . . . . . . . . . . . . . 64
5.6 Evaluation of the Compression Potential . . . . . . . . . . . . . . 64
5.6.1 Description of the Compression Method . . . . . . . . . . 65
5.6.2 Measure of the Compression Rate . . . . . . . . . . . . . 65
5.6.3 Measure of the Image Quality . . . . . . . . . . . . . . . 65
5.6.4 Results . . . . . . . . . . . . . . . . . . . . . . . . . . . 66
5.7 Comparison with Other Wavelet Software . . . . . . . . . . . . . 68

**6 Large-Scale Image Processing** **69**

6.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . 69
6.2 Very Large Images and Tiling . . . . . . . . . . . . . . . . . . . 69
6.2.1 Tiling . . . . . . . . . . . . . . . . . . . . . . . . . . . . 70
6.2.2 Block Management . . . . . . . . . . . . . . . . . . . . . 72
6.2.3 Implementation in WAILI . . . . . . . . . . . . . . . . . 72
6.3 Image Processing on Tiled Images . . . . . . . . . . . . . . . . . 73
6.3.1 Block-Based Denoising . . . . . . . . . . . . . . . . . . . 73
6.3.2 Image Compression . . . . . . . . . . . . . . . . . . . . 74
6.3.3 Crop and Merge . . . . . . . . . . . . . . . . . . . . . . 74
6.3.4 A GIS Application . . . . . . . . . . . . . . . . . . . . . 74
6.4 Transform Results . . . . . . . . . . . . . . . . . . . . . . . . . 75
6.5 Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . . 76

**7 The Red-Black Wavelet Transform** **79**

7.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . 79
7.2 The Red-Black Wavelet Transform . . . . . . . . . . . . . . . . . 80
7.2.1 A Two-Step Method . . . . . . . . . . . . . . . . . . . . 80
7.2.2 Borders . . . . . . . . . . . . . . . . . . . . . . . . . . . 83

7.2.3 Reordering . . . . . . . . . . . . . . . . . . . . . . . . 83
7.2.4 Integer Wavelet Transform . . . . . . . . . . . . . . . . . 83
7.2.5 Basis Functions . . . . . . . . . . . . . . . . . . . . . . 84
7.2.6 Other Members of the Family . . . . . . . . . . . . . . . 84
7.3 Properties of the Red-Black Wavelet Transform . . . . . . . . . . 86
7.4 The Redundant Wavelet Transform . . . . . . . . . . . . . . . . . 87
7.5 Comparison with the CDF $(2,2)$ Tensor Product Wavelets . . . . . 87
7.5.1 Test case 1 . . . . . . . . . . . . . . . . . . . . . . . . 89
7.5.2 Test case 2 . . . . . . . . . . . . . . . . . . . . . . . . 89
7.6 Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . 89

**8 Experiments with a Wavelet-Based APOD 93**
8.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . 93
8.2 The Proper Orthogonal Decomposition . . . . . . . . . . . . . . . 94
8.3 The Approximate Proper Orthogonal Decomposition . . . . . . . . 96
8.3.1 Theory of Operation . . . . . . . . . . . . . . . . . . . . 96
8.3.2 Decorrelating Transforms . . . . . . . . . . . . . . . . . 97
8.3.3 Decomposition into the Basis of Approximate Eigenmodes 100
8.4 Wavelet Transforms . . . . . . . . . . . . . . . . . . . . . . . . 101
8.5 Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 103
8.5.1 Wavelets . . . . . . . . . . . . . . . . . . . . . . . . . . 103
8.5.2 The Dynamical System . . . . . . . . . . . . . . . . . . . 104
8.5.3 Snapshots . . . . . . . . . . . . . . . . . . . . . . . . . 104
8.5.4 “Strange” Snapshots . . . . . . . . . . . . . . . . . . . . 107
8.5.5 Eigenvalue Distribution . . . . . . . . . . . . . . . . . . 107
8.5.6 Eigenmodes . . . . . . . . . . . . . . . . . . . . . . . . . 108
8.5.7 Reconstruction of the Snapshots . . . . . . . . . . . . . . 108
8.5.8 Coefficients of the Snapshots in the APOD Basis . . . . . 111
8.5.9 Timings . . . . . . . . . . . . . . . . . . . . . . . . . . . 111
8.6 Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . 117

**9 Conclusions and Suggestions for Future Research 119**
9.1 Wavelet Transforms . . . . . . . . . . . . . . . . . . . . . . . . 119
9.2 A Software Library for Wavelet-Based Image Processing . . . . . 120
9.3 Approximate POD using Wavelet Packets . . . . . . . . . . . . . 120
9.4 Suggestions for Future Research . . . . . . . . . . . . . . . . . . 121

**Bibliography 123**

**A Lifting Decompositions 129**

**B A Simple Image Compression Example using WAILI 137**

**Index 141**

# Notations and Abbreviations

## List of Symbols

| | | |
|---|---|---|
| $\langle \cdot, \cdot \rangle$ | : | inner product |
| $\lvert \cdot \rvert$ | : | degree of a Laurent polynomial |
| $\lVert \cdot \rVert$ | : | vector norm |
| $\oplus$ | : | direct sum |
| $\otimes$ | : | tensor product |
| $\triangleright$ | : | compression |
| $\triangleleft$ | : | decompression |
| $\delta_i$ | : | Kronecker delta |
| $\gamma_{j,l}$ | : | $j$th wavelet coefficient at resolution level $l$ |
| $\lambda_{j,l}$ | : | $j$th scaling function coefficient at resolution level $l$ |
| $\varphi(x)$ | : | (primal) scaling function |
| $\psi(x)$ | : | (primal) wavelet function |
| $\widetilde{\varphi}(x)$ | : | dual scaling function |
| $\widetilde{\psi}(x)$ | : | dual wavelet function |
| $d_i$ | : | detail signal |
| $g$ | : | (primal) high-pass filter |
| $h$ | : | (primal) low-pass filter |
| $\widetilde{g}$ | : | dual high-pass filter |
| $\widetilde{h}$ | : | dual low-pass filter |
| $\mathbf{L}^2$ | : | vector space of square integrable functions in $\mathbf{R}$ |
| $\mathbf{N}$ | : | vector space of natural numbers |
| $O$ | : | big "O" order symbol |
| $\mathcal{P}(\cdot)$ | : | prediction operator |
| $\mathbf{R}$ | : | vector space of real numbers |
| $s_i$ | : | (coarse representation of a) signal |
| $\mathcal{U}(\cdot)$ | : | update operator |
| $V_j$ | : | subspace spanned by the (primal) scaling function at resolution level $j$ |

| | | |
|---|---|---|
| $W_j$ | : | subspace spanned by the (primal) wavelet function at resolution level $j$ |
| $\widetilde{V}_j$ | : | subspace spanned by the dual scaling function at resolution level $j$ |
| $\widetilde{W}_j$ | : | subspace spanned by the dual scaling function at resolution level $j$ |
| $X^T$ | : | transposition of the matrix $X$ |
| $x(z)$ | : | $z$-transform of the signal or filter $x$ |
| $\mathbf{Z}$ | : | vector space of integer numbers |

# List of Abbreviations

| | | |
|---|---|---|
| 1D | : | one-dimensional |
| 2D | : | two-dimensional |
| APOD | : | Approximate Proper Orthogonal Decomposition |
| CDF | : | Cohen-Daubechies-Feauveau |
| CPU | : | Central Processing Unit |
| CWT | : | Continuous Wavelet Transform |
| DCT | : | Discrete Cosine Transform |
| DWT | : | Discrete Wavelet Transform |
| FFT | : | Fast Fourier Transform |
| FIR | : | Finite Impulse Response |
| FWT | : | Fast Wavelet Transform |
| GB | : | Gigabytes |
| GCV | : | Generalized Cross Validation |
| GIS | : | Geographical Information Systems |
| HP | : | high-pass signal |
| JPEG | : | Joint Photographic Experts Group |
| KLT | : | Karhunen-Loève Transform |
| LP | : | low-pass signal |
| MB | : | Megabytes |
| PDE | : | Partial Differential Equation |
| POD | : | Approximate Proper Orthogonal Decomposition |
| PSNR | : | Peak Signal to Noise Ratio |
| QMF | : | Quadrature Mirror Filter |
| RMSE | : | Root Mean Square Error |
| TB | : | Terabytes |
| WAILI | : | Wavelets with Integer Lifting |
| WPT | : | Wavelet Packet Transform |

# List of Figures


2.1 The filter bank algorithm for orthogonal wavelets. . . . . . . . . . 9
2.2 Orthogonal scaling function $\varphi(x)$ and wavelet $\psi(x)$ with 2 vanishing moments, as constructed by Daubechies. . . . . . . . . . . . . 10
2.3 The filter bank algorithm for biorthogonal wavelets. . . . . . . . . 11
2.4 The primal scaling function $\varphi(x)$ and wavelet $\psi(x)$ for the CDF $(2,2)$ biorthogonal wavelets. . . . . . . . . . . . . . . . . . . . . . . . 11
2.5 The wavelet decomposition tree. . . . . . . . . . . . . . . . . . . 12
2.6 The two-dimensional wavelet transform. . . . . . . . . . . . . . . 13
2.7 Translation invariance of the wavelet transform. . . . . . . . . . . 14
2.8 A wavelet packet decomposition tree. . . . . . . . . . . . . . . . 15
2.9 The two-dimensional wavelet packet decomposition tree. . . . . . 16

3.1 The lifting scheme. . . . . . . . . . . . . . . . . . . . . . . . . 25
3.2 The floating point wavelet transform in a compression application. 29
3.3 Graphical representation of the integer Haar wavelet transform. . . 32
3.4 A finite signal. . . . . . . . . . . . . . . . . . . . . . . . . . . 34
3.5 Periodic extension of a finite signal. . . . . . . . . . . . . . . . 34
3.6 Symmetric extension of a finite signal. . . . . . . . . . . . . . . 35
3.7 The four cases of symmetric signal extension of a discrete finite signal. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
3.8 The filter bank scheme for the symmetric wavelet transform. . . . 36
3.9 The four cases of symmetric signal extension using lifting. . . . . 37

4.1 Noise reduction by wavelet shrinking. . . . . . . . . . . . . . . . 42
4.2 *GCV* and mean square error of the result in function of the threshold $\delta$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 44
4.3 Downscaling of an image using simple subsampling and using wavelet transforms. . . . . . . . . . . . . . . . . . . . . . . . . 45
4.4 Upscaling of an image using pixel replication and using wavelet transforms. . . . . . . . . . . . . . . . . . . . . . . . . . . . 46
4.5 Upscaling of an image with a factor 5 using pixel replication and using a wavelet transform. . . . . . . . . . . . . . . . . . . . . 47

4.6 General scheme of a lossless image compression algorithm using wavelet transforms. . . . 49
4.7 Zerotree encoding . . . . 51

5.1 Example of a channel after two transform levels. . . . 57
5.2 Relations among the major components of WAILI. . . . 58
5.3 The two-dimensional wavelet transform. . . . 61
5.4 Two steps of the two-dimensional wavelet transform. . . . 64
5.5 Comparison of compression rate vs. quality of reconstruction for different types of wavelets. . . . 66
5.6 Comparison of compression rate vs. quality of reconstruction for the original input image and two denoised versions. . . . 67

6.1 Locality in the *space* and *wavelet domain*. . . . 70
6.2 Division of a subband in blocks. . . . 71
6.3 Subdivision of an *image* into *channels*, *levels*, *subbands* and *blocks*. 72
6.4 Design of the large-scale image processing system. . . . 73

7.1 Horizontal/vertical lifting in the odd steps: *Red-Black*. . . . 81
7.2 Diagonal lifting in the even steps: *Blue-Yellow*. . . . 82
7.3 Reordering of the interleaved coefficients and multilevel decomposition. . . . 83
7.4 Graphs of the basis functions for the integer Red-Black wavelet transform. . . . 85
7.5 Graphs of the basis functions for the integer Cohen-Daubechies-Feauveau $(2,2)$ tensor product wavelets. . . . 88
7.6 Grey scale test images. . . . 90
7.7 Three-level wavelet decomposition of the test image . . . . 90
7.8 Reconstruction of the original image after denoising. . . . 91

8.1 Schematic overview of the Approximate Proper Orthogonal Decomposition. . . . 98
8.2 Bifurcation diagram for the Brusselator model. . . . 105
8.3 The four sets of input snapshots. . . . 106
8.4 “Strange” input snapshots. . . . 107
8.5 Sum of the eigenvalues vs. number of eigenvalues. . . . 108
8.6 The 4 most important eigenmodes for the POD and for the APOD. 109
8.7 The 4 most important eigenmodes for the POD and for the APOD (cont’d). . . . 110
8.8 First two coefficients of the decomposition of the snapshots into the best basis. . . . 112
8.9 Maximum error in the coefficients. . . . 113

8.10 First two coefficients of the decomposition of the “strange” snapshots into the best basis. . . . . . . . . . . . . . . . . . . . . . . . . 114
8.11 Maximum absolute error in the coefficients. . . . . . . . . . . . . . 115

# List of Tables


6.1 Timings for the wavelet transform of an image of size $3584 \times 3584$. 76

8.1 The snapshots used for testing and their bifurcation parameter values. 105
8.2 Mean percentage of energy after reconstruction of the snapshots vs. number of used eigenmodes for different compression rates. . . 111
8.3 Mean percentage of energy after reconstruction of the "strange" snapshots vs. number of used eigenmodes for different compression rates. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 115
8.4 Computation times for the Approximate Proper Orthogonal Decomposition . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 116
8.5 Comparison of the complexity of the various methods. . . . . . . 116

# Chapter 1

# Introduction

## 1.1 Overview of Wavelet Research

Wavelet theory is a synthesis of ideas from many different research domains. It combines aspects like filter banks, multi-resolution, (bi)orthogonality, . . .

The first wavelet transforms were continuous wavelet transforms. In the Continuous Wavelet Transform (CWT), a signal is analyzed by calculating inner products with test functions. All test functions are generated from one oscillating function with finite support, called a wavelet function. Only one condition is imposed on the wavelet function: its integral must be zero. Different test functions are created by translating and dilating the wavelet function, thus one has a customizable time/frequency window through which the analysis is done. This locality in both time and frequency is advantageous, compared to Fourier analysis [21].

The Discrete Wavelet Transform (DWT) is an adaptation to digital signals, which are discrete [43]. Usually not only the translations are limited to multiples of the sampling distance, but also the dilation (scale) factor becomes dyadic: one can distinguish different resolution levels, with a scale factor of two in between. Hence we have a multi-resolution analysis [34], which is related to pyramidal schemes like the Laplacian pyramid [1].

If the wavelet functions are orthogonal, the DWT is stable. The first orthogonal wavelets with compact support were constructed by Daubechies [12]. These allowed for an efficient implementation, the Fast Wavelet Transform (FWT) or the filter bank algorithm, creating the link with filter banks and Quadrature Mirror Filters (QMF), which are commonly used in subband coding. By relaxing the orthogonality conditions, biorthogonal wavelets could provide for both compact support and symmetry, as is shown by e.g. the biorthogonal wavelets constructed by Cohen, Daubechies and Feauveau [9].

Wavelet packets are an extension of wavelet theory: they provide an adapted basis using the filter bank algorithm, filling the gap between the standard wavelet decomposition tree and the area of subband coding.

The lifting scheme, developed by Sweldens, started as a method to improve a given discrete wavelet transform to obtain specific properties [48]. Later it became an efficient algorithm to calculate any wavelet transform as a sequence of simple lifting steps [14]. Digital signals are usually a sequence of integer numbers, while wavelet transforms result in floating point numbers. For an efficient reversible implementation it is of great importance to have a transform algorithm that converts integers to integers. Fortunately a lifting step can be modified to operate on integers, while preserving the reversibility [6]. Thus the lifting scheme became a method to implement reversible integer wavelet transforms.

The lifting scheme was also extended to a generic method to construct second-generation wavelets [49]. These are wavelets that are no longer created as the translation and dilation of one wavelet function. The lifting scheme allows to construct wavelets for higher dimensional grids, irregular samplings, triangularizations, spheres, … [40, 31].

Wavelets are used in many applications, like signal and image processing, image and video compression, mathematical physics, numerical analysis, communication, system theory, identification, geometrical modeling, computer graphics, theoretical physics, approximation theory, stochastic processes, …. Of course this list is far from exhaustive. We refer to the literature for more information, e.g. [3, 7, 8, 28, 29, 33, 36, 46, 58].

We focused on two applications: image processing and compression to create low-dimensional models for dynamical systems.

We worked out the details of integer wavelet transforms, based on the lifting scheme, for a class of biorthogonal wavelets (Cohen-Daubechies-Feauveau). Based on this we designed and implemented a software library called *WAILI* (Wavelets with Integer Lifting) that provides wavelet transforms and wavelet-based image processing operations on two-dimensional images [56]. Later we added support for very large images and block-based processing [55].

We created a new kind of second-generation wavelets on a rectangular grid, based on a red-black blocking scheme. These Red-Black wavelets are less anisotropic than tensor product wavelets [52].

We reduced the complexity of the Proper Orthogonal Decomposition (POD) by biorthogonal wavelet packet compression and evaluated the resulting Approximate POD by analyzing a large-scale dynamical system, described by partial differential equations [53].

## 1.2 Overview of the Thesis

In Chapter 2 we give an introduction to the concepts multi-resolution analysis and wavelets from the classical viewpoint. The lifting scheme, discussed in Chapter 3, not only provides a framework to create so-called second-generation wavelets, but also allows to implement classical wavelet transforms. It has several advantages over the classical computation schemes and provides additional features like a true integer transform. We have worked out the details of integer wavelet transforms for a class of biorthogonal wavelets (Cohen-Daubechies-Feauveau)

The largest share of this thesis is about the application of wavelets in image processing. Many image processing operations can be done on the wavelet transformed data. In Chapter 4 we discuss some of them. One of the most important image processing operations is image compression. It is obvious that large images need to be stored and handled in a compressed form. Wavelets can provide for this since wavelet-based techniques offer better compression rates than other techniques.

Chapter 5 discusses the design and features of *WAILI*, a software library that implements the integer wavelet transforms described in Chapter 3. We developed WAILI as a basic software library to provide wavelet transforms and wavelet-based operations on two-dimensional images. The library is written in C++. In application areas like Geographical Information Systems (GIS), image sizes are measured in gigabytes and even terabytes. In Chapter 6 we describe the extension of WAILI to block-based processing, which allows for the handling of such very large images and for a parallel implementation.

In Chapter 7 we show how the lifting scheme can be used to create a new kind of second-generation wavelets on a rectangular grid, based on a red-black blocking scheme. These wavelets have some appealing properties compared to tensor products of traditional one-dimensional wavelets.

Besides applications in image and signal processing, wavelets are also an important tool for the creation of low-dimensional models for dynamical systems. Instead of analyzing the original partial differential equations (PDEs) that describe the dynamical system, one can analyze a low-dimensional model that approximates the original equations. The Proper Orthogonal Decomposition (POD) allows to create a "best" basis for such a model, derived from solutions of the PDEs, but it is an expensive operation. In Chapter 8 we propose a method using biorthogonal wavelet packet compression to reduce the complexity of the POD. The resulting Approximate POD is evaluated by constructing a low-dimensional model for a large-scale dynamical system.

## 1.3 Situation of the Research

Most of this work was carried out in the scope of the project "Wavelet-Based Interactive Video Communication and Image Database Consulting" (ITA/950244). This project involves three partners (Eurosense, Barco, and K.U.Leuven) and has the following goals:

- The development of a software library for wavelet transforms, including support for very large images and images composed of subimages.
- The implementation of image compression and processing operations.
- The integration in a Geographical Information System (GIS).
- A hardware assisted implementation of the wavelet transform for real-time full-motion video.

The research results presented in Chapter 8, about the Approximate Proper Orthogonal Decomposition, predate the above project. It is related to the ongoing research for the analysis of periodic solutions in large-scale dynamical systems described by partial differential equations by other members of the research group Scientific Computing of the Computer Science department.

# Chapter 2

# Wavelets

In this chapter we introduce wavelets and the discrete wavelet transform from the classical viewpoint, based on the concept multi-resolution analysis. The Fast Wavelet Transform (FWT) allows to calculate a wavelet transform in an efficient way. We will not go into all deep details, but we will limit ourselves to the topics that are important for the rest of this thesis.

## 2.1 What are Wavelets?

The main idea behind wavelet analysis is to decompose a signal $f$ into a basis of functions $\Psi_i$:

$$f = \sum_i a_i \Psi_i.$$

To have an efficient representation of the signal $f$ using only a few coefficients $a_i$, it is very important to use a suitable family of functions $\Psi_i$. The functions $\Psi_i$ should match the features of the data we want to represent.

Real-world signals usually have the following features: they are both limited in time (time-limited)[1] and limited in frequency (band-limited). Time-limited signals can be represented efficiently using a basis of block functions (Dirac delta functions for infinitesimal small blocks). But block signals are not limited in frequency. Band-limited signals can be represented efficiently using a Fourier basis, but sines and cosines are not limited in time.

What we need is a compromise between the pure time-limited and band-limited basis functions, a compromise that combines the best of both worlds: wavelets ("small waves").

[1] Or space-limited in the case of images.

## 2.2 Why wavelets?

One of the main features of wavelets that are important for the applications that we shall consider is their good decorrelation:

- Wavelets are localized in both the space/time and scale/frequency domains. Hence they can easily detect local features in a signal.
- Wavelets are based on a multi-resolution analysis. A wavelet decomposition allows to analyze a signal at different resolution levels (scales).
- Wavelets are smooth, which can be characterized by their number of vanishing moments. A function defined on the interval $[a,b]$ has $n$ vanishing moments if
$$\int_a^b f(x)x^i\,dx = 0,$$
for $i = 0,1,\ldots n-1$. The higher the number of vanishing moments, the better smooth signals can be approximated in a wavelet basis.

Furthermore there exist fast ($O(n)$) and stable (wavelets can be orthogonal or biorthogonal) algorithms to calculate the discrete wavelet transform and its inverse.

## 2.3 Wavelets and Multi-Resolution Analysis

Classically, discrete wavelets are introduced in the concept of a multi-resolution analysis. We introduce this concept first.

### 2.3.1 Multi-Resolution Analysis

Consider the space $\mathbf{L}^2$, the vector space of square integrable functions in $\mathbf{R}$:

$$\mathbf{L}^2 = \left\{ f : \int_{-\infty}^{+\infty} f^2(x)\,dx < \infty \right\}.$$

In a multi-resolution analysis [34] we decompose $\mathbf{L}^2$ in nested subspaces $V_j$

$$\cdots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \cdots$$

such that the closure of their union is $\mathbf{L}^2$

$$\overline{\bigcup_{j=-\infty}^{+\infty} V_j} = \mathbf{L}^2,$$

and their intersection contains only the zero-function

$$\bigcap_{j=-\infty}^{+\infty} V_j = \{0\}.$$

In the dyadic case, i.e. when each subspace $V_j$ is twice as large as $V_{j-1}$, a function $f(x)$ that belongs to one of these subspaces $V_j$ has the following properties:

$$f(x) \in V_j \quad \Leftrightarrow \quad \text{dilation } f(2x) \in V_{j+1}, \tag{2.1}$$
$$f(x) \in V_0 \quad \Leftrightarrow \quad \text{translation } f(x+1) \in V_0. \tag{2.2}$$

If we can find a function $\varphi(x) \in V_0$ such that the set of functions consisting of $\varphi(x)$ and its integer translates

$$\{\varphi(x-k)\}_{k\in\mathbf{Z}}$$

form a basis for the space $V_0$, we call it a *scaling function* or *father function*. For the other subspaces $V_j$ (with $j \neq 0$) we define:

$$\varphi_{j,k}(x) \equiv 2^{j/2}\,\varphi\big(2^j x - k\big)\,.$$

### 2.3.2 Wavelet Functions

Because the subspaces $V_j$ are nested:

$$V_j \subset V_{j+1},$$

we can decompose $V_{j+1}$ in $V_j$ and $W_j$, the orthogonal complement of $V_j$ in $V_{j+1}$:

$$\begin{aligned} V_j \oplus W_j &= V_{j+1}, \\ W_j &\perp V_j. \end{aligned}$$

The direct sum of the subspaces $W_j$ is equal to $\mathbf{L}^2$:

$$\overline{\bigcup_{j=-\infty}^{+\infty} V_j} = \overline{\bigoplus_{j=-\infty}^{+\infty} W_j} = \mathbf{L}^2.$$

This means that $V_j$ is a "coarse-resolution" representation of $V_{j+1}$, while $W_j$ carries the "high-resolution" difference information between $V_{j+1}$ and $V_j$.

If we can find a function $\psi(x) \in W_0$ that obeys the translation property (2.2), i.e.

$$\psi(x) \in W_0 \Leftrightarrow \text{translation } \psi(x+1) \in W_0,$$

and such that the set of functions consisting of $\psi(x)$ and its integer translates

$$\{\psi(x-k)\}_{k\in\mathbf{Z}}$$

form a basis for the space $W_0$, we call it a *wavelet function* or *mother function*. For the other subspaces $W_j$ (with $j \neq 0$) we define:

$$\psi_{j,k}(x) \equiv 2^{j/2}\,\psi\big(2^j x - k\big)\,.$$

### 2.3.3 The Fast Wavelet Transform (FWT)

Because both $V_0$ and $W_0$ are subspaces of $V_1$:

$$V_0 \subset V_1 \text{ and } W_0 \subset V_1,$$

we can express $\varphi(x)$ and $\psi(x)$ in terms of the basis functions of $V_1$:

$$\begin{aligned}
\varphi(x) &= 2\sum_k h_k\,\varphi(2x-k),\\
\psi(x) &= 2\sum_k g_k\,\varphi(2x-k).
\end{aligned}$$

Due to the multi-resolution analysis, these relations are also valid between $V_{j+1}$, $V_j$ and $W_j$ for arbitrary $j$.

We call the $h_k$ and $g_k$ the filter coefficients that uniquely define the scaling function $\varphi(x)$ and the wavelet $\psi(x)$.

Since $V_{j+1} = V_j \oplus W_j$, we can express a function $f(x)$ that is written in terms of the basis functions of $V_{j+1}$ in terms of the basis functions of $V_j$ and $W_j$ also:

$$\begin{aligned}
f(x) &= \sum_k \lambda_{j+1,k}\,\varphi_{j+1,k}(x),\\
&= \sum_l \lambda_{j,l}\,\varphi_{j,l}(x) + \sum_l \gamma_{j,l}\,\psi_{j,l}(x).
\end{aligned}$$

with the transform coefficients $\lambda_{j,l}$ and $\gamma_{j,l}$ defined by:

$$\begin{aligned}
\lambda_{j,l} &= \sqrt{2}\sum_k h_{k-2l}\,\lambda_{j+1,k},\\
\gamma_{j,l} &= \sqrt{2}\sum_k g_{k-2l}\,\lambda_{j+1,k}.
\end{aligned}$$

This operation is known as the *Fast Wavelet Transform* (FWT). It is called “fast” because its complexity is $O(n)$, i.e. the amount of work is proportional to the signal length. The filter $h_k$ is a low-pass filter, while $g_k$ is a high-pass filter.

The inverse wavelet transform can be obtained in a similar way. This algorithm is known as the filter bank algorithm (Figure 2.1).

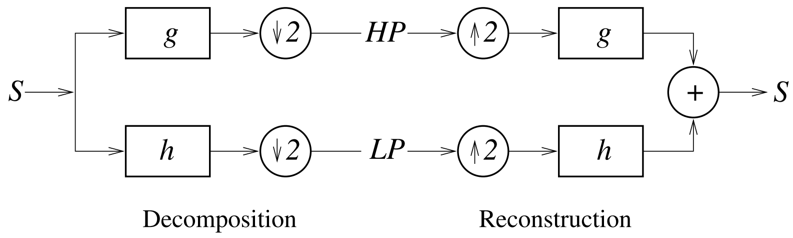



Figure 2.1: The filter bank algorithm for orthogonal wavelets: the signal $S$ is filtered and downsampled to get a low-pass signal $LP$ and a high-pass signal $HP$. It can be reconstructed by upsampling and filtering with the correct filters.

### 2.3.4 Orthogonal Wavelets

If the $\varphi_{j,k}(x)$ and $\psi_{j,k}(x)$ are orthonormal:

$$\begin{aligned} V_j &\perp W_j, \\ \langle \varphi_{j,l}, \varphi_{j,l'} \rangle &= \delta_{l-l'}, \\ \langle \psi_{j,l}, \psi_{j',l'} \rangle &= \delta_{j-j'} \delta_{l-l'}, \end{aligned}$$

then we can calculate the coefficients of the decomposition

$$f(x) = \sum_l \lambda_{j,l}\, \varphi_{j,l}(x) + \sum_l \gamma_{j,l}\, \psi_{j,l}(x)$$

by taking inner products with the scaling and wavelet functions:

$$\begin{aligned} \lambda_{j,l} &= \langle f, \varphi_{j,l} \rangle, \\ \gamma_{j,l} &= \langle f, \psi_{j,l} \rangle. \end{aligned}$$

The decomposition in the wavelet basis is guaranteed to be stable: if the function $f(x)$ is slightly changed, this will cause only a slight change in the coefficients $\lambda_{j,l}$ and $\gamma_{j,l}$. Parseval's identity holds:

$$\begin{aligned} \|f\|^2 &= \sum_k \lambda^2_{j+1,k}, && (2.3) \\ &= \sum_l \lambda^2_{j,l} + \sum_l \gamma^2_{j,l}. && (2.4) \end{aligned}$$

Examples of orthogonal wavelets are the family of orthogonal wavelets constructed by Daubechies [12]. The scaling function $\varphi(x)$ and wavelet function $\psi(x)$ for the member with 2 vanishing moments (also known as "D4" because the corresponding wavelet filter has 4 taps or filter coefficients) is shown in Figure 2.2.

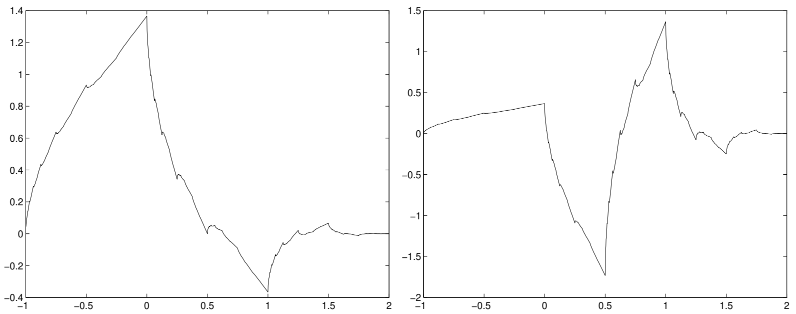


Figure 2.2: Orthogonal scaling function $\varphi(x)$ and wavelet $\psi(x)$ with 2 vanishing moments, as constructed by Daubechies.

### 2.3.5 Biorthogonal Wavelets

If we relax the orthogonality conditions to biorthogonality conditions we can obtain wavelets with some special properties that are not possible with orthogonal wavelets. In the biorthogonal case we have two multi-resolution analyses, a primal and a dual:

- primal: $V_j$, $W_j$, $\varphi_{j,k}$, $\psi_{j,k}$,
- dual: $\widetilde{V}_j$, $\widetilde{W}_j$, $\widetilde{\varphi}_{j,k}$, $\widetilde{\psi}_{j,k}$.

The biorthogonality conditions imply:

$$\begin{aligned} \widetilde{V}_j &\perp W_j, \\ V_j &\perp \widetilde{W}_j, \\ \langle \widetilde{\varphi}_{j,l}, \varphi_{j,l'} \rangle &= \delta_{l-l'}, \\ \langle \widetilde{\psi}_{j,l}, \psi_{j',l'} \rangle &= \delta_{j-j'} \delta_{l-l'}. \end{aligned}$$

Because the biorthogonal wavelets form a Riesz basis

$$A\|f\|^2 \leq \sum_l \lambda_{j,l}^2 + \sum_l \gamma_{j,l}^2 \leq B\|f\|^2, \tag{2.5}$$

with $A$ and $B$ positive constants[2], the decomposition in the biorthogonal wavelet basis is still stable and the coefficients $\lambda_{j,l}$ and $\gamma_{j,l}$ of the decomposition are now obtained by calculating inner products with the dual basis functions:

$$\begin{aligned} \lambda_{j,l} &= \langle f, \widetilde{\varphi}_{j,l} \rangle, \\ \gamma_{j,l} &= \langle f, \widetilde{\psi}_{j,l} \rangle. \end{aligned}$$

[2]In the orthogonal case we have $A = B = 1$, i.e. we have Parseval's identity.

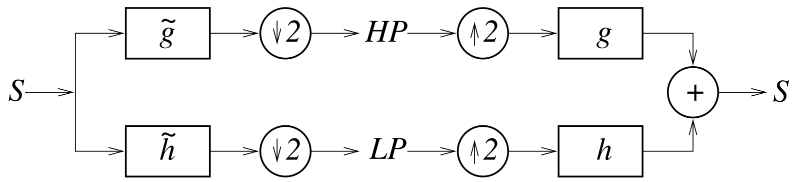



Figure 2.3: The filter bank algorithm for biorthogonal wavelets.

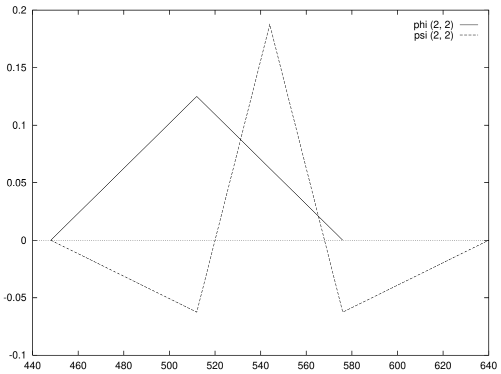



Figure 2.4: The primal scaling function $\varphi(x)$ and wavelet $\psi(x)$ for the Cohen-Daubechies-Feauveau $(2,2)$ biorthogonal wavelets. The dual scaling function and wavelet are not shown because of their irregular nature.

We can still use the filter bank algorithm if we use the dual filter pair $(\widetilde{h},\widetilde{g})$ (related to the dual multi-resolution analysis) for the decomposition and the primal filter pair $(h,g)$ (related to the primal multi-resolution analysis) for the reconstruction (Figure 2.3).

Examples of biorthogonal wavelets are the family of biorthogonal wavelets constructed by Cohen, Daubechies and Feauveau [9]. The primal scaling and wavelet functions for the member with 2 vanishing moments for both the primal and the dual wavelet are shown in Figure 2.4.

### 2.3.6 The Wavelet Decomposition Tree

The subspaces $V_j$ are nested, and each of them can be split in two subspaces $V_{j-1}$ and $W_{j-1}$. If we start with the highest resolution subspace $V_0$, we perform the following decompositions:

$$V_0 = V_{-1} \oplus W_{-1},$$

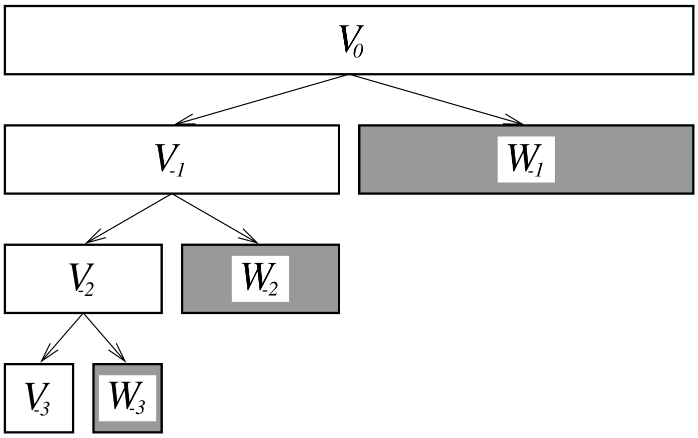



Figure 2.5: The wavelet decomposition tree.

$$
\begin{aligned}
&= V_{-2} \oplus W_{-2} \oplus W_{-1}, \\
&= V_{-3} \oplus W_{-3} \oplus W_{-2} \oplus W_{-1}, \\
&\vdots
\end{aligned}
$$

and we obtain a decomposition tree as in Figure 2.5.

Since the amount of data to transform is halved at each resolution level, the total complexity of the full wavelet decomposition is still $O(n)$. As a comparison, the Fast Fourier Transform (FFT) has a complexity of $O(n \log n)$.

### 2.3.7 Higher Dimensions

The above wavelets were defined on the real line **R**, i.e. on a one-dimensional domain. To create wavelets on higher dimensional domains, we can follow two approaches:

- We can perform the wavelet transform independently for each dimension. Because the wavelet transform can be written as a multiplication with a wavelet transform matrix and due to the associativity of the matrix product, the exact order is not important.

  In the two-dimensional case we get a decomposition as shown in Figure 2.6. The square variant is most used and in that case the basis functions are tensor products of the one-dimensional basis functions, i.e. after one transform step we have:

$$
\begin{array}{c|c}
\varphi(x) \otimes \varphi(x) & \varphi(x) \otimes \psi(x) \\
\hline
\psi(x) \otimes \varphi(x) & \psi(x) \otimes \psi(x)
\end{array}
$$

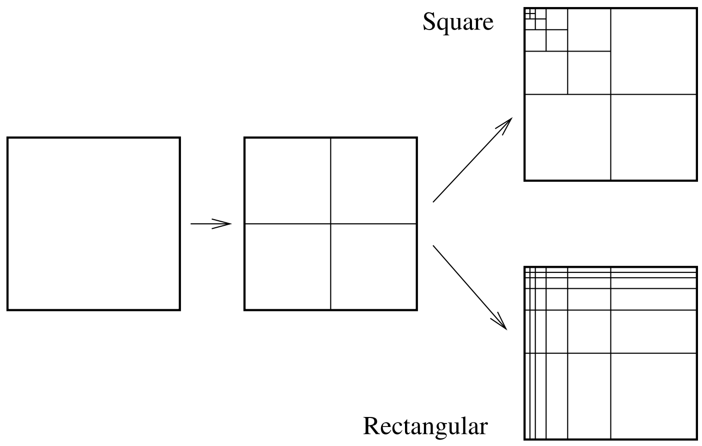



Figure 2.6: The two-dimensional wavelet transform: the square and the rectangular variant.

- An alternative is to use real multi-dimensional wavelets, which are less anisotropic with respect to directional information in the original data, e.g. Red-Black wavelets (see Section 7.2).

## 2.4 The Wavelet Transform and Translation Invariance

Due to the subsampling in the filter bank algorithm a wavelet transform is not translation invariant: if a signal is delayed or advanced its wavelet transform is not simply a delayed or advanced version of the wavelet transform of the original signal. Only if the delay or advance is a multiple of $2^n$, with $n$ the number of transform levels, the wavelet transform will be a delayed or advanced version of the original transformed signal. In two dimensions, it is even worse because this condition needs to be true in both the horizontal and vertical directions.

If each transform level is considered independently, one step of a wavelet transform is translation invariant if the translation is limited to an even number of pixels. For the one-dimensional case and one transform level, this is shown in Figure 2.7.

## 2.5 The Redundant Wavelet Transform

The wavelet transform can be converted into a translation invariant transform by removing the subsampling step. This is called the *redundant wavelet transform* (or

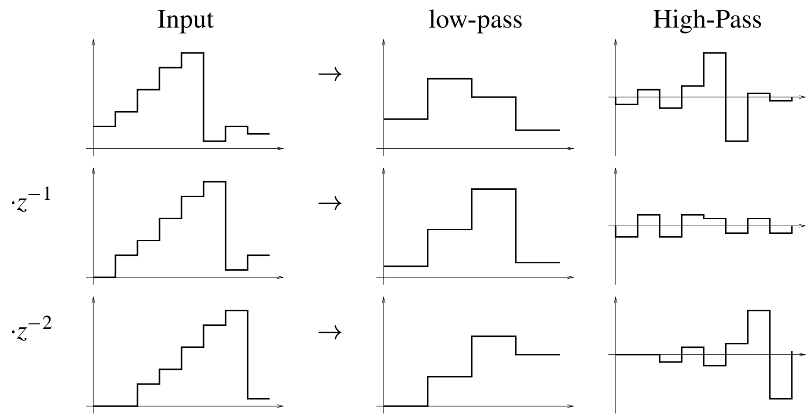



Figure 2.7: Translation invariance of the wavelet transform. When delaying the input signal by one time step, the resulting wavelet transform is completely different, while a delay of the input signal by two time steps results in a delayed version (by one time step at the coarser level) of the transformed original signal.

stationary, or non-decimated wavelet transform) [37]. It is important for applications that can suffer from translation variance, like e.g. feature detection and noise reduction.

In the redundant wavelet transform one gets rid of the decimation step, causing all subbands to have the same size as the size of the input data set. At each resolution level, the filters have to be upsampled[3] to keep a consistent multi-resolution analysis.

Because all subbands have the same size, the computational complexity is no longer $O(n)$, but $O(pn)$, with $p \leq \log n$ the number of transform levels. The redundant transform also requires more memory ($O(pn)$) to store the subbands. For the inverse transform, we have to calculate some kind of mean value due to the redundancy in the forward transform.

## 2.6 Wavelet Packets

In every step of a wavelet transform the low-frequency data of the signal is split in a low- and a high-frequency part, as can be seen in Figure 2.5. After decomposition the low-frequency part with the lowest index and the high-frequency parts will be stored.

In a wavelet packet transform, one also splits the high-frequency part in a "low"

[3] A filter is upsampled by putting zeros between the successive filter coefficients.

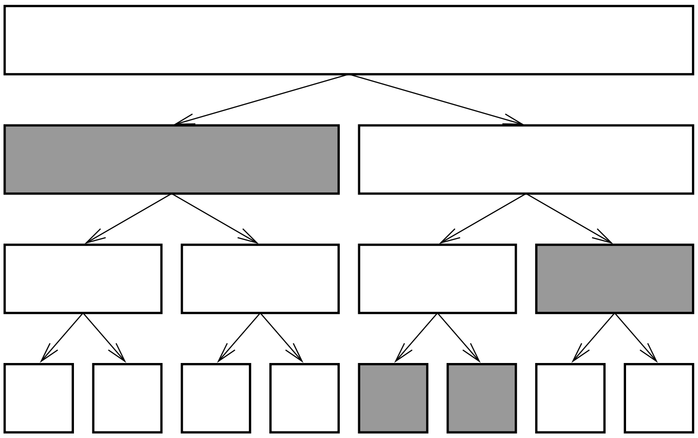


Figure 2.8: A wavelet packet decomposition tree.

and a "high" frequency part [10]. This produces a decomposition tree as shown in the Figures 2.8 (1D) and 2.9 (2D), and leads to a redundant tree of possible basis functions. A wavelet packet transform can be calculated with a complexity of $O(n \log n)$, compared to $O(n)$ for an ordinary wavelet transform.

The aim of wavelet packet analysis is to choose a "best basis", i.e. to find the set of functions that best decorrelates the input data and that is a subset of the basis functions in the decomposition tree [61].

The selection of the best basis is equivalent to the answer to the question "Do we best split this part into a low- and a high-frequency part or not?". We can make these decisions by starting at the bottom level of the tree in Figure 2.8 or 2.9 and comparing the "cost" of two (1D) respectively four (2D) neighboring "child" squares in this level with the "cost" of the "parent" square in the higher level. The "cost" is a measure for the number of bits we would need to store the data in the corresponding basis, i.e. a measure for the achieved decorrelation.

We retain the cheapest squares and proceed in the same way with the higher level, until we arrive at the top level. The basis functions that correspond to the retained squares form the best basis.

To calculate the "cost" of a matrix $v = [v_{ij}]$, several possibilities are available. We mention a few of them:

- **1-Norm**-based cost function: the cost of storing a matrix is proportional the the 1-norm of the matrix:
$$\text{cost} = \sum_{i,j} |v_{ij}|.$$

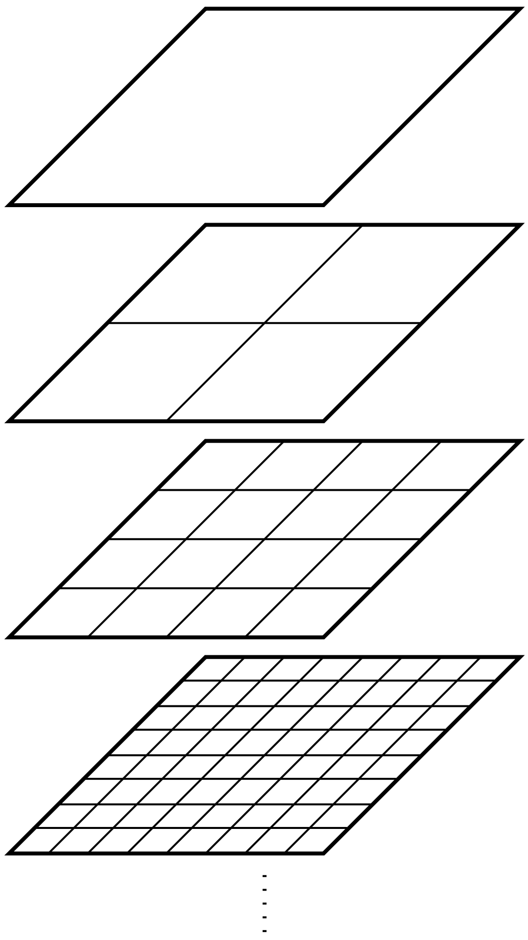


Figure 2.9: The two-dimensional wavelet packet decomposition tree.

- **First order Shannon Entropy**-based cost function: the first order entropy is the theoretical number of bits per symbol needed to encode a string of symbols by considering individual symbols [41]:

$$\text{cost} = -\sum_{i,j} v_{ij} \log |v_{ij}|.$$

- **Threshold**-based cost function:

$$\text{cost} = \#V_T \qquad \text{with } V_T = \left\{ v_{ij} \mid |v_{ij}| > T \right\},$$

with $T$ a given threshold.

- **Logarithm of Energy**-based cost function:

$$\text{cost} = \sum_{i,j} \log(v_{ij})^2,$$

which is equivalent with

$$\text{cost} = \sum_{i,j} \log |v_{ij}|.$$

This is a variant of the first order Shannon entropy.

# Chapter 3

# The Lifting Scheme

In this chapter we introduce the lifting scheme. The lifting scheme is an efficient implementation of a wavelet transform algorithm. Furthermore it can easily be converted to operate on integer data. The inverse transform is trivial to find and always provides perfect reconstruction of the input data.

All classical wavelet transforms can be implemented using the lifting scheme by a factorization in lifting steps. We work out the details of integer wavelet transforms for a class of biorthogonal wavelets (Cohen-Daubechies-Feauveau).

## 3.1 Introduction

The lifting scheme is an algorithm to calculate wavelet transforms in an efficient way. It was developed first as a method to improve a given wavelet transform to obtain some specific properties. Later it was extended to a generic method to create so-called second-generation wavelets. In the one-dimensional case, these are wavelets that are not necessarily dilates and translates of one function. They are much more flexible and can be used to define a wavelet basis on an interval or on an irregular grid, or even on a sphere.

Second-generation wavelets are more general in the sense that all the classical wavelets can be generated by the lifting scheme. In fact, the decomposition of a classical wavelet filter into lifting steps can be easily obtained via the Euclidean algorithm [14].

The theory behind the type of wavelets described in Section 2.3 relies heavily on Fourier analysis, while the lifting scheme can also be used to introduce wavelets without using the concept of Fourier transform. This allows to explain the basic ideas behind wavelets in a much simpler way.

Several introductions to the lifting scheme are available [48, 49, 40, 47].

## 3.2 Predict and Update

The wavelet transform of a one-dimensional signal is a multi-resolution representation of that signal where the wavelets are the basis functions which at each resolution level decorrelate the signal.

Thus at each level, the (low-pass part of the) signal is split into a high-pass and a low-pass part. These high-pass and low-pass parts are obtained by applying corresponding wavelet filters. In general these filters are coupled if certain conditions are to be fulfilled, like for example perfect reconstruction: there have to exist filters to perform the inverse transform without any data loss.

The lifting scheme is an efficient implementation of these filtering operations at each level when computing a discrete wavelet transform. So suppose that the low-resolution part of a signal at level $j+1$ is given and that it consists of a data set which we represent by $\lambda_{j+1}$. This set is transformed into two other sets at level $j$: the low-resolution part $\lambda_j$ and the high-resolution part $\gamma_j$. This is obtained first by just splitting the data set $\lambda_{j+1}$ into two separate data subsets. Traditionally this is done by separating the set of even samples and the set of odd samples. Such a splitting is sometimes referred to as the *lazy wavelet transform*. Doing just this of course does not improve our representation of the signal. Therefore, the next step is to recombine these two sets in several subsequent lifting steps which decorrelate the two signals.

Lifting steps usually come in pairs of a primal and a dual lifting step. A *dual lifting* step can be seen as a prediction: the data $\gamma_j$ are "predicted" from the data in the subset $\lambda_j$. When the signals are still highly correlated, then such a prediction will usually be very good, and thus we do not have to keep this information in both signals. That is why we can store only the part of $\gamma_j$ that differs from its prediction (the prediction error). Thus $\gamma_j$ is replaced by $\gamma_j - \mathcal{P}(\lambda_j)$ where $\mathcal{P}$ represents the prediction operator. This is the real decorrelating step.

However, the new representation has lost certain basic properties, which one usually wants to keep, like for example the mean value of the signal. To restore this property, one needs a *primal lifting* step, whereby the set $\lambda_j$ is updated with data computed from the (new) subset $\gamma_j$. Thus $\lambda_j$ is replaced by $\lambda_j + \mathcal{U}(\gamma_j)$ with $\mathcal{U}$ some updating operator.

In general, several such lifting steps can be applied in sequence to go from level $j+1$ to level $j$.

The principles that were just explained become very clear in the case of the simple CDF $(2,2)$ transform that we give below in Section 3.2.2. The odd samples are predicted by linear interpolation of the neighboring even samples and are therefore replaced by the interpolation error. The even samples are updated to preserve the mean value of the signal.

To recapitulate, let us consider a simple lifting scheme with only one pair of lifting steps to go from level $j+1$ to level $j$.

**Splitting** (*lazy wavelet transform*) Partition the data set $\lambda_{j+1}$ into two distinct data sets $\lambda_j$ and $\gamma_j$.

**Prediction** (*dual lifting*) Predict the data in the set $\gamma_j$ by the data set $\lambda_j$ and replace $\gamma_j$ by the prediction error:

$$\gamma_j \leftarrow \gamma_j - \mathcal{P}(\lambda_j).$$

**Update** (*primal lifting*) Update the data in the set $\lambda_j$ by the data in set $\gamma_j$:

$$\lambda_j \leftarrow \lambda_j + \mathcal{U}(\gamma_j).$$

These steps can be repeated by iteration on the $\lambda_j$, creating a multi-level transform or multi-resolution decomposition.

### 3.2.1 The Inverse Transform

One of the great advantages of the lifting scheme realization of a wavelet transform is that it decomposes the wavelet filters into extremely simple elementary steps, and each of these steps is very easily invertible. As a result, the inverse wavelet transform can always be obtained immediately from the forward transform. The inversion rules are obvious: revert the order of the operations, invert the signs in the lifting steps, and replace the splitting step by a merging step. Thus, inverting the three step procedure above results in

**Inverse update** $$\lambda_j \leftarrow \lambda_j - \mathcal{U}(\gamma_j),$$

**Inverse prediction** $$\gamma_j \leftarrow \gamma_j + \mathcal{P}(\lambda_j),$$

**Merge** $$\lambda_{j+1} \leftarrow \lambda_j \cup \gamma_j.$$

### 3.2.2 Example: Linear Prediction

From the second-generation viewpoint, one transform step of a discrete one-dimensional signal $x = \{x_k\}$ looks like:

**Splitting** Split the signal $x$ (i.e. $\lambda_{j+1}$) into even samples (i.e. $\lambda_j$) and odd samples (i.e. $\gamma_j$):

$$\begin{aligned} s_i &\leftarrow x_{2i}, \\ d_i &\leftarrow x_{2i+1} \end{aligned}$$

**Prediction** Predict the odd samples using linear interpolation:

$$d_i \leftarrow d_i - \frac{1}{2}(s_i + s_{i+1}).$$

**Update** Update the even samples to preserve the mean value of the samples:

$$s_i \leftarrow s_i + \frac{1}{4}(d_{i-1} + d_i).$$

As a result, the signal $s = \{s_k\}$ is a coarse representation of the original signal $x$, while the signal $d = \{d_k\}$ contains the high-frequency information that is lost when going from resolution level $j+1$ to resolution level $j$.

The same kind of interpolation scheme can be used for interpolating polynomials of higher degree. These lifting schemes need some special care at the boundaries if one wants the wavelets to live explicitly on the discrete set where the data are defined. A whole family of lifting schemes can be constructed in this way [50], of which the above example is just the most simple case.

Note that this transform works on one-dimensional data. For two-dimensional data, it can be applied row and column wise, resulting in a tensor product wavelet transform. This is a separable transform.

Later we will see that this scheme turns out to be the lifting equivalent of the popular Cohen, Daubechies and Feauveau [9] classical biorthogonal wavelets with two vanishing moments for both the primal and dual wavelet (hence named CDF $(2,2)$ wavelet).

Thanks to the lifting scheme, the accompanying wavelet transform can be implemented in an efficient way (see e.g. [57]).

### 3.2.3 Flexibility

This scheme is much more flexible than the one used for classical wavelets, based on Fourier analysis. Using lifting one can create wavelets for higher dimensions, on irregular samplings, intervals, triangularizations, ... [40].

A simple method to construct two-dimensional wavelets using lifting will be given in Section 7.2.

## 3.3 Unifying the Two Paradigms

The previous section introduced the lifting scheme from a second-generation viewpoint. However, all classical wavelet transforms can be implemented using this powerful scheme.

### 3.3.1 The Filter Bank Algorithm

The basic block in a discrete wavelet transform (DWT) [26] is a filter bank, consisting of 2 filters (cfr. Figure 2.3). A discrete signal $S$ is filtered by a high-pass filter $\widetilde{g}$ and a low-pass filter $\widetilde{h}$, and downsampled. The results are a high-pass signal

$HP$ and a low-pass signal $LP$, each containing half as much samples as the input signal $S$.

For the inverse transform, first the signals $HP$ and $LP$ are upsampled by putting zeros in between every sample. After that they are filtered by the filters $g$ and $h$ and the result is added together. In the case of perfect reconstruction — we only consider perfect reconstruction filter banks here — the resulting signal is equal to the original signal $S$, and the filters $g$, $h$, $\widetilde{g}$ and $\widetilde{h}$ are called *wavelet filters*. The filters $\widetilde{g}$ and $\widetilde{h}$ are called dual filters w.r.t. $g$ and $h$.

The signal $LP$ resembles the original signal $S$ very much: it is a representation of $S$ at a lower resolution level. On the other hand, $HP$ contains the detail information that is lost by making the transition to that lower resolution level [34].

If the low-frequency information is treated as a new signal and passed through the same filter bank, and this step is repeated several times, after some iterations a very low-frequency signal is retained. Together with the detail information for the different resolution levels, it represents the original signal decomposed in several resolution levels. This is called a *wavelet transform*.

### 3.3.2 Polyphase Representations

The $z$-transform of a signal $x = \{x_k\}$ is defined as

$$x(z) = \sum_k x_k z^{-k},$$

whenever this makes sense. Thus, the $z$-transform of a finite impulse response (FIR) filter $h = \{h_{k_b}, \ldots, h_{k_e}\}$ is a Laurent polynomial $h(z)$ given by

$$h(z) = \sum_{k=k_b}^{k_e} h_k z^{-k}.$$

Assuming $h_{k_b}, h_{k_e} \neq 0$, we define the degree of such a Laurent polynomial as

$$|h(z)| = k_e - k_b.$$

For consistency we set the degree of the zero polynomial equal to $-\infty$.

Filtering a signal $x$ by a filter $h$ is easily described in the $z$-domain as an ordinary multiplication: $y(z) = h(z)x(z)$.

Subsampling a signal $x = \{x_k\}$ corresponds to keeping only the even samples $x^s = \{x_{2k}\}$. Thus its $z$-transform is given by

$$x^s(z) = x_e(z) = \sum_k x_{2k} z^{-k}$$

where $x_e(z)$ is the even part of $x(z)$, i.e. we define $x_e(z^2)$ as

$$x_e(z^2) = \frac{x(z) + x(-z)}{2}.$$

Defining the odd part $x_o(z^2)$ in a similar way

$$x_o(z^2) = \frac{z}{2}[x(z) - x(-z)] = \sum_k x_{2k+1} z^{-2k}$$

we can decompose $x(z)$ as

$$x(z) = x_e(z^2) + z^{-1} x_o(z^2).$$

The filtering part of the filter bank decomposition described above can be written as

$$\begin{bmatrix} lp(z) \\ hp(z) \end{bmatrix} = \begin{bmatrix} h(z) \\ g(z) \end{bmatrix} x(z).$$

while the subsampling step corresponds to

$$\begin{aligned} LP(z^2) &= \frac{lp(z) + lp(-z)}{2} = lp_e(z^2), \\ HP(z^2) &= \frac{hp(z) + hp(-z)}{2} = lp_o(z^2). \end{aligned}$$

This can be combined as

$$\begin{bmatrix} LP(z^2) \\ HP(z^2) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} h(-z) & h(z) \\ g(-z) & g(z) \end{bmatrix} \begin{bmatrix} x(-z) \\ x(z) \end{bmatrix}.$$

However, by first filtering and then subsampling, we first compute all the coefficients and then throw away half of the work done. It would be more efficient if the subsampling step could be done before the filtering, which means that we only compute the even parts of $lp$ and $hp$. Obviously

$$lp_e(z) = [h(z)x(z)]_e = h_e(z)x_e(z) + z^{-1} h_o(z) x_o(z).$$

Thus $\lambda(z) = lp_e(z)$ and $\gamma(z) = hp_e(z)$ are obtained as

$$\begin{bmatrix} \lambda(z) \\ \gamma(z) \end{bmatrix} = P(z) \begin{bmatrix} x_e(z) \\ z^{-1} x_o(z) \end{bmatrix},$$

where $P(z)$ is the *polyphase matrix*

$$P(z) = \begin{bmatrix} h_e(z) & h_o(z) \\ g_e(z) & g_o(z) \end{bmatrix}.$$

A filter pair $(h, g)$ is *complementary* if the corresponding polyphase matrix $P(z)$ has determinant 1. For wavelet filter pairs, this is always the case because of the perfect reconstruction condition.

The *lazy wavelet transform* is the most simple wavelet transform. It does nothing but split the signal in even and odd components. Its polyphase matrix is the unit matrix:

$$P_{\text{lazy}}(z) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

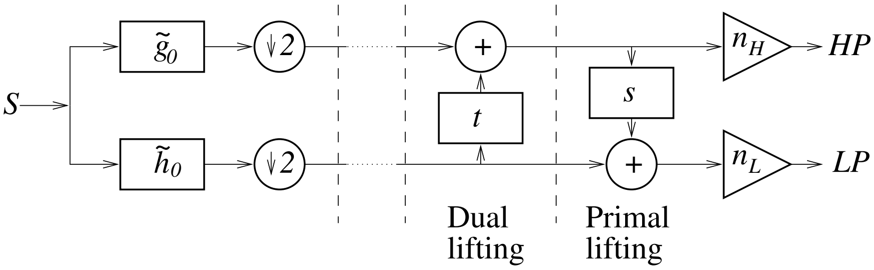



Figure 3.1: The lifting scheme: starting from wavelet filters $\widetilde{g}_0$ and $\widetilde{h}_0$ we *lift* the signal to obtain another wavelet decomposition. The final step is a normalization with factors $n_H$ and $n_L$.

### 3.3.3 Primal and Dual Lifting

The lifting scheme [48, 49, 47] found its roots in a method to improve a given wavelet transform to obtain some specific properties.

With *primal lifting*, starting from two complementary finite filters $h$ and $g$, a new finite filter $h^{\text{new}}$ complementary to $g$ is created:

$$h^{\text{new}}(z) = h(z) + s(z^2)g(z),$$

or, using the polyphase representation:

$$P^{\text{new}}(z) = \begin{bmatrix} h_e^{\text{new}}(z) & h_o^{\text{new}}(z) \\ g_e(z) & g_o(z) \end{bmatrix} = \begin{bmatrix} 1 & s(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} h_e(z) & h_o(z) \\ g_e(z) & g_o(z) \end{bmatrix}.$$

With *dual lifting*, starting from two complementary finite filters $h$ and $g$, a new finite filter $g^{\text{new}}$ complementary to $h$ is created:

$$g^{\text{new}}(z) = g(z) + t(z^2)h(z),$$

or, using the polyphase representation:

$$P^{\text{new}}(z) = \begin{bmatrix} h_e(z) & h_o(z) \\ g_e^{\text{new}}(z) & g_o^{\text{new}}(z) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ t(z) & 1 \end{bmatrix} \begin{bmatrix} h_e(z) & h_o(z) \\ g_e(z) & g_o(z) \end{bmatrix}.$$

This process can be repeated, obtaining a *cakewalk* of alternating primal and dual lifting steps, as shown in Figure 3.1. If the initial $P(z)$ is the unit matrix — i.e. we start from the lazy wavelet transform — the result is a product of unit upper and lower triangular $2 \times 2$ matrices of Laurent polynomials.

### 3.3.4 Decomposition into Lifting Steps

Daubechies and Sweldens [14] proved that *any* polyphase matrix representing a wavelet transform with finite filters can be factored in a finite product of unit upper and lower triangular $2 \times 2$ matrices, and a diagonal normalization matrix:

$$P(z) = \underbrace{\begin{bmatrix} K_1 & 0 \\ 0 & K_2 \end{bmatrix}}_{\text{normalization}} \prod_{i=m}^{1} \left\{ \overbrace{\begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix}}^{\text{primal lifting}} \underbrace{\begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix}}_{\text{dual lifting}} \right\}. \tag{3.1}$$

with $s_i(z)$ and $t_i(z)$ Laurent polynomials and $K_1$ and $K_2$ normalization factors. It can be shown [14] that, based on a Euclidean type algorithm, it is possible to obtain any wavelet transform by performing a lazy wavelet transform followed by alternating primal and dual lifting steps, and a normalization.

From this it immediately follows that the inverse transform can be written as

$$P^{-1}(z) = \prod_{i=1}^{m} \left\{ \begin{bmatrix} 1 & 0 \\ -t_i(z) & 1 \end{bmatrix} \begin{bmatrix} 1 & -s_i(z) \\ 0 & 1 \end{bmatrix} \right\} \begin{bmatrix} 1/K_1 & 0 \\ 0 & 1/K_2 \end{bmatrix}.$$

How can we find the $s_i(z)$ and $t_i(z)$? To decompose the polyphase matrix using primal lifting as

$$P(z) = \begin{bmatrix} h_e(z) & h_o(z) \\ g_e(z) & g_o(z) \end{bmatrix} = \begin{bmatrix} h_e(z) & h_o^{\text{new}}(z) \\ g_e(z) & g_o^{\text{new}}(z) \end{bmatrix} \begin{bmatrix} 1 & s(z) \\ 0 & 1 \end{bmatrix}$$

we have to find Laurent polynomials $s(z)$, $h_o^{\text{new}}(z)$, and $g_o^{\text{new}}(z)$ such that

$$\begin{aligned} h_o(z) &= s(z)h_e(z) + h_o^{\text{new}}(z), \\ g_o(z) &= s(z)g_e(z) + g_o^{\text{new}}(z), \end{aligned}$$

with (recall $|\cdot|$ means degree)

$$\begin{aligned} |h_o^{\text{new}}(z)| &< |h_o(z)|, \\ |g_o^{\text{new}}(z)| &< |g_o(z)|. \end{aligned}$$

This corresponds to two *long division with remainder* schemes of Laurent polynomials, with a common quotient $s(z)$. This corresponds to a Euclidean algorithm applied to $h_o(z)$ and $h_e(z)$. The same Euclidean algorithm can be applied to $g_o(z)$ and $g_e(z)$, as is illustrated in the example below. This is a consequence of the perfect reconstruction condition. For more details see [14]. Note that the division of Laurent polynomials is not unique, so we may have multiple possibilities.

The scheme for dual lifting is similar.

## Example

The analyzing filter pair for the Cohen-Daubechies-Feauveau biorthogonal wavelets [9] with 2 vanishing moments for both the primal and dual wavelet functions (CDF $(2,2)$) is (up to a normalization factor $\sqrt{2}$)

$$\begin{aligned}
\widetilde{h}(z) &= -\frac{1}{8}z^{-2}+\frac{1}{4}z^{-1}+\frac{3}{4}+\frac{1}{4}z-\frac{1}{8}z^{2},\\
\widetilde{g}(z) &= \frac{1}{4}z^{-2}-\frac{1}{2}z^{-1}+\frac{1}{4}.
\end{aligned}$$

The corresponding polyphase matrix is

$$P(z)=\begin{bmatrix} -\frac{1}{8}z^{-1}+\frac{3}{4}-\frac{1}{8}z & \frac{1}{4}+\frac{1}{4}z \\ \frac{1}{4}z^{-1}+\frac{1}{4} & -\frac{1}{2} \end{bmatrix}.$$

To decompose this matrix using dual lifting as

$$P(z)=\begin{bmatrix} h_e^{\mathrm{new}}(z) & \frac{1}{4}+\frac{1}{4}z \\ g_e^{\mathrm{new}}(z) & -\frac{1}{2} \end{bmatrix}\begin{bmatrix} 1 & 0 \\ t(z) & 1 \end{bmatrix}$$

we have to find Laurent polynomials $t(z)$, $h_e^{\mathrm{new}}(z)$, and $g_e^{\mathrm{new}}(z)$ such that

$$\begin{aligned}
-\frac{1}{8}z^{-1}+\frac{3}{4}-\frac{1}{8}z &= h_e^{\mathrm{new}}(z)+t(z)\left(\frac{1}{4}+\frac{1}{4}z\right),\\
\frac{1}{4}z^{-1}+\frac{1}{4} &= g_e^{\mathrm{new}}(z)+t(z)\left(-\frac{1}{2}\right).
\end{aligned}$$

Thus we calculate the quotient $t(z)$ and the remainder $h_e^{\mathrm{new}}(z)$ of a long division. This division is not unique: there are three solutions. We have chosen:

$$\begin{array}{ccc|l}
-\frac{1}{8}z^{-1} & +\frac{3}{4} & -\frac{1}{8}z & \frac{1}{4}+\frac{1}{4}z \\ \cline{4-4}
-\frac{1}{8}z^{-1} & -\frac{1}{8} & & -\boldsymbol{\frac{1}{2}z^{-1}}-\boldsymbol{\frac{1}{2}} \\ \cline{1-2}
 & +\frac{7}{8} & -\frac{1}{8}z & \\
 & -\frac{1}{8} & -\frac{1}{8}z & \\ \cline{2-3}
 & +\mathbf{1} & &
\end{array}$$

This choice will allow us to perform further lifting steps later. A solution to this problem is thus given by

$$\begin{aligned} t(z) &= -\frac{1}{2}z^{-1} - \frac{1}{2}, \\ h_e^{\text{new}}(z) &= 1, \\ g_e^{\text{new}}(z) &= 0, \end{aligned}$$

and thus

$$P(z) = \begin{bmatrix} 1 & \frac{1}{4} + \frac{1}{4}z \\ 0 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\frac{1}{2}z^{-1} - \frac{1}{2} & 1 \end{bmatrix}.$$

We can continue the decomposition to obtain

$$P(z) = \begin{bmatrix} 1 & 0 \\ 0 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{4} + \frac{1}{4}z \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\frac{1}{2}z^{-1} - \frac{1}{2} & 1 \end{bmatrix}.$$

Summarized, the analyzing step can be performed by a dual lifting step, followed by a primal lifting step and a normalization. If we omit the normalization factors $K_1$ and $K_2$, it can be implemented using the following algorithm:

| **Forward transform** | | **Inverse transform** | |
|---|---|---|---|
| Splitting | $s_i \leftarrow x_{2i}$<br>$d_i \leftarrow x_{2i+1}$ | Inverse primal lifting | $s_i \leftarrow s_i - \frac{1}{4}(d_{i-1} + d_i)$ |
| Dual lifting | $d_i \leftarrow d_i - \frac{1}{2}(s_i + s_{i+1})$ | Inverse dual lifting | $d_i \leftarrow d_i + \frac{1}{2}(s_i + s_{i+1})$ |
| Primal lifting | $s_i \leftarrow s_i + \frac{1}{4}(d_{i-1} + d_i)$ | Merging | $x_{2i} \leftarrow s_i$<br>$x_{2i+1} \leftarrow d_i$ |

## Other examples

The decompositions in lifting steps of some other wavelet transforms of this family are shown in Appendix A.

### 3.3.5 Advantages of the Lifting Scheme

Summarized, the lifting scheme has the following immediate advantages, when compared to the classical filter bank algorithm:

- It is faster (but still $O(n)$, of course). In the case of CDF $(2,2)$, the original filters $\widetilde{h}(z)$ and $\widetilde{g}(z)$ count $5+3=8$ filter coefficients, while the lifting decomposition gives us $s(z)$ and $t(z)$ counting only $2+2=4$ filter coefficients.

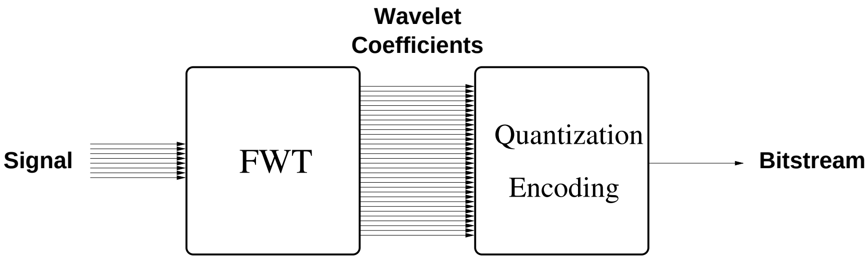



Figure 3.2: The floating point wavelet transform in a compression application.

- The inverse transform is trivial to find. One does not need to explicitly find the corresponding analysis filters.
- The lifting scheme is easier to understand and implement.
- The lifting scheme is more generic and allows for the creation of second-generation wavelets.

Furthermore all wavelet filters can be implemented using the lifting scheme. Thus giving a specific wavelet transform, an implementation using the lifting scheme can always be derived using the Euclidean algorithm. This algorithm forms the bridge between the classical wavelet transforms and the lifting scheme.

## 3.4 The Integer Wavelet Transform

In many applications (e.g. image compression and processing, video, audio, ...) the input data consist of integer samples. In addition the storage and encoding of integer numbers is easier, compared to floating point numbers. Unfortunately all of the above transforms assume that the input samples are floating point values. Even if the input values actually are integer, doing the filtering operations on these numbers will transform them in rational or real numbers because the filter coefficients need not be integers. To obtain an efficient implementation of the discrete wavelet transform, it is of great practical importance that the wavelet transform is represented by a set of integers as well (possibly up to some scaling factor). Of course, this can always be obtained by scaling and rounding the real numbers, but this rounding process will loose information, and after rounding, the original signal can not be reconstructed from its transform without an error.

Especially from a compression point of view, it looks a bit silly to first expand the integer input data from e.g. 8 bits per sample to 32 bit wide floating point numbers, while the goal is to reduce the total amount of data (cfr. Figure 3.2).

### 3.4.1 Integer Transforms

Fortunately the lifting scheme can be modified easily to a transform that maps integers to integers and that is reversible, and thus allows a perfect reconstruction [6, 57]. This is done by adding some rounding operations, at the expense of introducing a non-linearity in the transform.

Since even the most elementary lifting steps involve for example divisions by 2, we obtain in general prediction values $\mathcal{P}(\lambda_j)$ and update values $\mathcal{U}(\gamma_j)$ which are not integer. We shall round these numbers to integers (for example the nearest integer) and indicate this operation by curly braces. Thus, we actually compute rounded values:

$$\begin{aligned}\gamma_j &\leftarrow \gamma_j - \left\{\mathcal{P}(\lambda_j)\right\}, \\ \lambda_j &\leftarrow \lambda_j + \left\{\mathcal{U}(\gamma_j)\right\}\end{aligned}$$

Because an operation of the form

$$a^{\text{new}} \leftarrow a^{\text{old}} + \{b\}$$

does not change $b$, independent of the kind of rounding operation, $a^{\text{old}}$ can always be reconstructed without any errors as

$$a^{\text{old}} \leftarrow a^{\text{new}} - \{b\}.$$

To see that perfect reconstruction is always possible, we note the following. The last computation in a forward transform step is to compute $\lambda_j \leftarrow \lambda_j + \left\{\mathcal{U}(\gamma_j)\right\}$. Thus the first computation in the inverse transform step is to recompute $\left\{\mathcal{U}(\gamma_j)\right\}$, and this will give exactly the same result as before if $\gamma_j$ has been perfectly reconstructed so far. Therefore $\lambda_j \leftarrow \lambda_j - \left\{\mathcal{U}(\gamma_j)\right\}$ reconstructs exactly the original $\lambda_j$. Thus also $\left\{\mathcal{P}(\lambda_j)\right\}$ can be reproduced in the reconstruction step, exactly as in the forward step, and thus also $\gamma_j \leftarrow \gamma_j + \left\{\mathcal{P}(\lambda_j)\right\}$ will give the original $\gamma_j$ back. This shows that each step of the lifting scheme with rounding, is perfectly invertible and thus the whole signal is perfectly reconstructible:

$$\begin{aligned}\lambda_j &\leftarrow \lambda_j - \left\{\mathcal{U}(\gamma_j)\right\}, \\ \gamma_j &\leftarrow \gamma_j + \left\{\mathcal{P}(\lambda_j)\right\}\end{aligned}$$

This is one of the most amazing features of integer lifting: whatever deterministic rounding operation is used, the lifting operation is *always* reversible.

### 3.4.2 Rational Lifting Coefficients

Using the $z$-transform, a lifting step basically looks like

$$x^{\text{new}}(z) \leftarrow x(z) + s(z)y(z).$$

Because the signal part $y(z)$ is not changed by the lifting step, the result of the filter operation can be rounded:

$$x^{\text{new}}(z) \leftarrow x(z) + \{s(z)y(z)\},$$

while retaining a 100% reversible operation:

$$x(z) \leftarrow x(z)^{\text{new}} - \{s(z)y(z)\}.$$

In some cases, all filter coefficients of $s(z)$ are rational and the corresponding lifting step can be written as

$$x_i \leftarrow x_i - \frac{1}{a}\sum_j b_j y_j,$$

where $a, b_j \in \mathbf{Z}$, i.e. they are integers. This is true for various wavelet transforms, among which the Cohen-Daubechies-Feauveau biorthogonal wavelets.

Such a lifting step can be modified in one of the following ways:

**Full rounding** The result of the division by $a$ is rounded:

$$\widetilde{x}_i \leftarrow x_i - \left\{\frac{\sum_j b_j y_j}{a}\right\}.$$

Here $\widetilde{x}_i$ is an integer approximation to the $x_i$ obtained with floating point lifting.

**Without rounding** We avoid the division by $a$ by multiplying the other terms with $a$:

$$\widetilde{x}_i \leftarrow a x_i - \sum_j b_j y_j.$$

Here $\widetilde{x}_i = ax$, and thus the dynamic range of the wavelet coefficients will increase. This has to be taken into account in later steps. Note that in this case no real rounding is performed, and thus this can be considered to be an "exact" implementation of the floating point version, yielding integers.

**Mixed form** We combine the rounding and multiplication steps of both methods:

$$\widetilde{x}_i \leftarrow a_1 x_i - \left\{\frac{\sum_j b_j y_j}{a_2}\right\},$$

with

$$\begin{aligned} a_1, a_2 &\in \mathbf{Z}, \\ a_1 \cdot a_2 &= a. \end{aligned}$$

This variant can be used if we want to have more control over the dynamic range of the resulting $\widetilde{x}_i$.

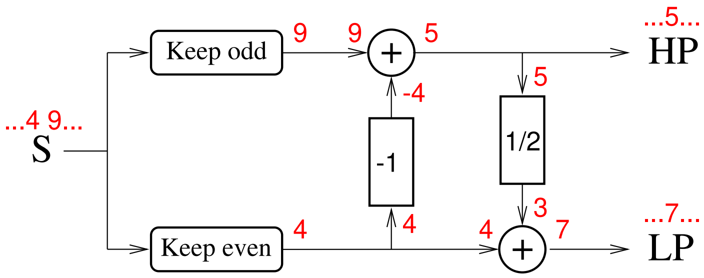



Figure 3.3: Graphical representation of the integer Haar wavelet transform.

Note that in all of the three cases above the modified lifting step is still reversible, and thus the perfect reconstruction feature is still present.

If $a, b_j \notin \mathbf{Z}$, we can still use the "full rounding" method to obtain a transform that maps integers to integers.

In most cases we will use "full rounding", except when we want to control the dynamic range of the result.

## Examples

**CDF** $(1,1)$ This is the well-known Haar transform. The integer version looks like:

| Forward transform | | Inverse transform | |
|---|---|---|---|
| Splitting: | $s_i \leftarrow x_{2i}$<br>$d_i \leftarrow x_{2i+1}$ | Inverse primal lifting: | $s_i \leftarrow s_i - \left\{\frac{d_i}{2}\right\}$ |
| Dual lifting: | $d_i \leftarrow d_i - s_i$ | Inverse dual lifting: | $d_i \leftarrow d_i + s_i$ |
| Primal lifting: | $s_i \leftarrow s_i + \left\{\frac{d_i}{2}\right\}$ | Merging: | $x_{2i} \leftarrow s_i$<br>$x_{2i+1} \leftarrow d_i$ |

A graphical representation of the forward transform is shown in Figure 3.3.

**CDF** $(2,2)$ The algorithm for the integer version is:

| Forward transform | | Inverse transform | |
|---|---|---|---|
| Splitting: | $s_i \leftarrow x_{2i}$<br>$d_i \leftarrow x_{2i+1}$ | Inverse primal lifting: | $s_i \leftarrow s_i - \left\{\frac{1}{4}(d_{i-1} + d_i)\right\}$ |
| Dual lifting: | $d_i \leftarrow d_i - \left\{\frac{1}{2}(s_i + s_{i+1})\right\}$ | Inverse dual lifting: | $d_i \leftarrow d_i + \left\{\frac{1}{2}(s_i + s_{i+1})\right\}$ |
| Primal lifting: | $s_i \leftarrow s_i + \left\{\frac{1}{4}(d_{i-1} + d_i)\right\}$ | Merging: | $x_{2i} \leftarrow s_i$<br>$x_{2i+1} \leftarrow d_i$ |

### 3.4.3 Normalization

In the above examples, we again omitted the normalization factors $K_1$ and $K_2$ (see Equation 3.1). However, in many applications (e.g. compression) it is important to know the magnitudes of the normalized wavelet coefficients. They indicate the importance of the corresponding wavelet components.

Furthermore, if we want the energy of the signal to be retained in the wavelet coefficients — or at least an approximation in the non-orthogonal case — we have to use normalized filter coefficients that are $\sqrt{2}$ times as large as the filter coefficients we used. Thus the real normalization factors are $\sqrt{2}K_1$ for the low-pass band, and $\sqrt{2}K_2$ for the high-pass band. Note that the normalization factors for the low-pass band accumulate, due to the iteration on the low-pass band.

In other cases, we may want that the mean value of the low-pass band coefficients is the same as the mean value of the original signal. This will be true if we make sure that $K_1 = 1$. E.g. if a digital image is transformed, the low-pass band will be a low-resolution version of the original image. Note that the dynamic range of the pixel values may increase due to the non-convexity of the low-pass filter.

## 3.5 Boundary Treatment

Real-world signals do not extend infinitely in time or space, but are limited to a finite interval (Figure 3.4). However, filter bank algorithms assume infinite signal lengths. Padding the signal with zeros is not acceptable, since it would lead to much more wavelet coefficients than original samples. If one truncates the transformed function to yield the same number of wavelet coefficients as samples, this will badly distort the signal and thus render perfect reconstruction impossible. What we need is some kind of signal extension, or take into account the finiteness of the signal by slightly changing the transform near the boundaries of the signal.

### 3.5.1 Classical Extension Methods

In classical signal processing applications, it is common practice to extend the data for the computations near the border, by replicating the signal using one of the following methods:

#### Periodic Extension

A first solution is the periodic extension of the signal: we extend the finite signal by putting copies of itself in front of and behind the original signal (Figure 3.5).

After a wavelet transform we can simply discard the coefficients that lie outside of the interval in which the original signal was defined, since these wavelet

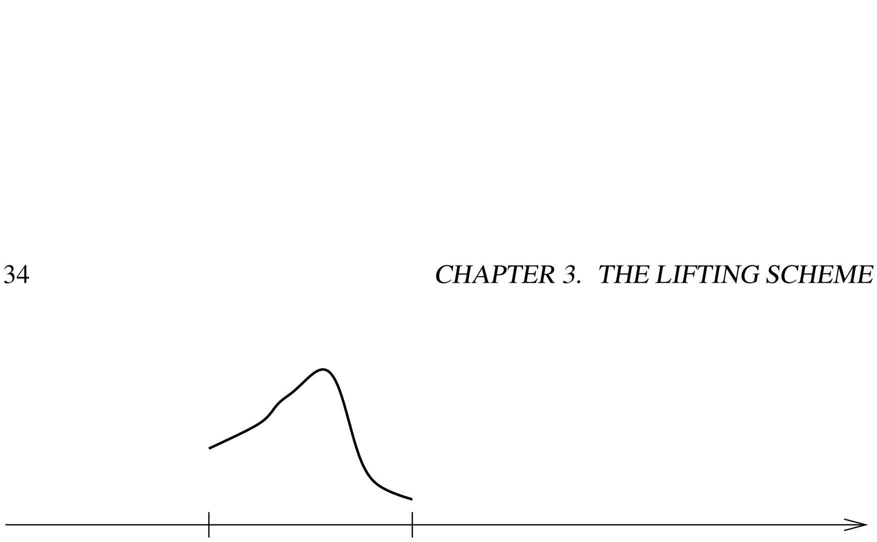


Figure 3.4: A finite signal.

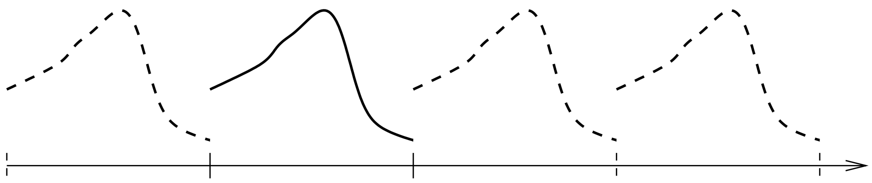


Figure 3.5: Periodic extension of a finite signal.

coefficients are the same as the retained coefficients, and thus can be recovered without any problems.

However, unless the first and the last sample have the same value, we introduce unwanted discontinuities at the boundaries of the original signal. These discontinuities will locally enlarge the wavelet coefficients and make compression of the signal more difficult.

**Symmetric Extension**

A better solution is the symmetric extension of the signal: we extend the finite signal by mirroring it around its endpoints, which makes the discontinuities disappear (Figure 3.6) [4]. But the higher order derivatives may still be discontinuous.

Symmetric extension can be considered as periodic extension applied to a concatenation of the original signal and a mirrored copy of the original signal. Hence after filtering one has to retain twice as much coefficients. Fortunately one can discard half of them if the filters are symmetric, yielding the same number of coefficients as the original signal length. That is why we use biorthogonal wavelets instead of orthogonal wavelets, since they can be symmetric.

While the periodic extension of a discrete (sampled) signal is straightforward and unique, there are four possible ways to extend a discrete signal symmetrically. One can duplicate the first and the last sample value or not. These cases are denoted by the notation $(a,b)$. If $a = 1$, the first sample value is not duplicated, while $a = 2$ indicates that the first sample value is duplicated. The value of $b$ indicates the possible duplication of the last sample value (Figure 3.7).

However, to ensure to get a perfect reconstruction, one has to apply the correct

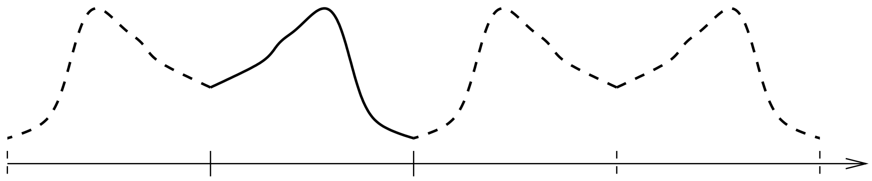


Figure 3.6: Symmetric extension of a finite signal.

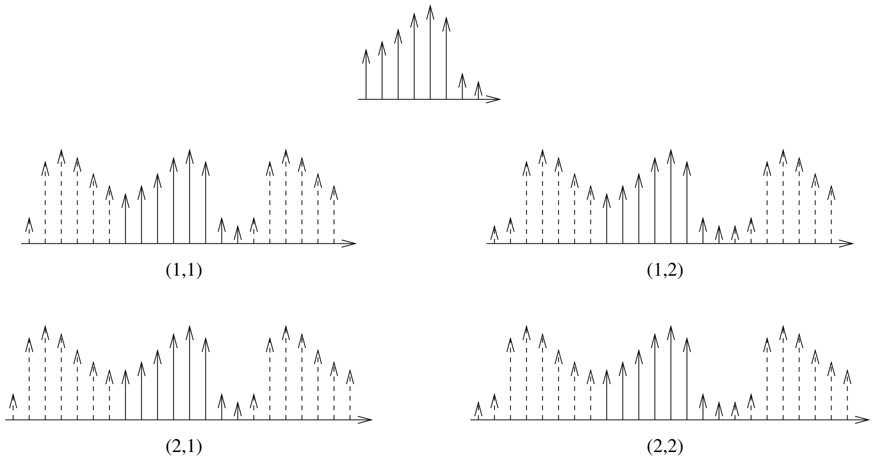



Figure 3.7: The four cases of symmetric signal extension of a discrete finite signal.

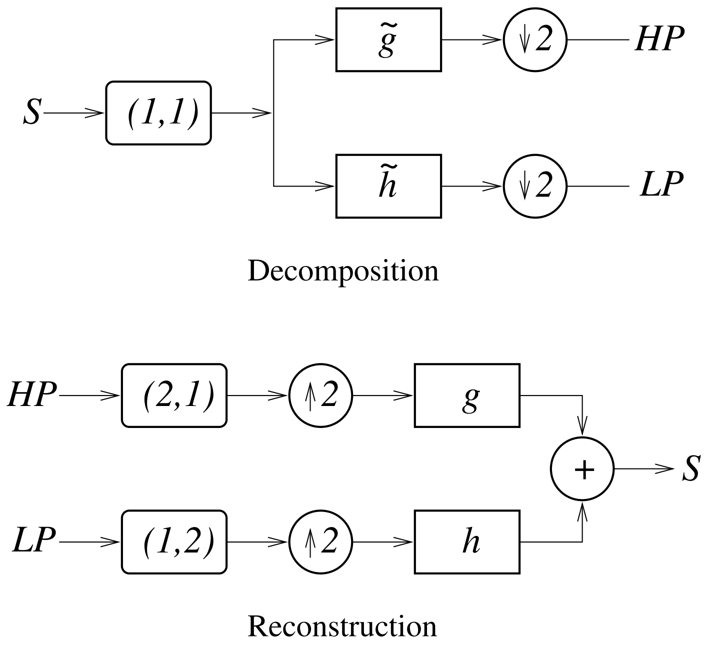



Figure 3.8: The filter bank scheme for the symmetric wavelet transform: filter operations are indicated by boxes, extension operators by rounded boxes and up/downsampling or addition operations by circles.

extension in every step of the decomposition and reconstruction, as indicated in Figure 3.8 [4].

### 3.5.2 Signal Extension with Lifting

Although the lifting scheme allows to transform signals with a finite length without extending the signal, we did not choose to take this approach. Instead we use the classical symmetric extension because it is easier to implement and suffices for the applications we have in mind. In each lifting step, we extend the signal symmetrically. Depending on the first and last sample being “even” or “odd’, we can consider four cases, as presented graphically in Figure 3.9:

**FELO** First even, last odd,

**FELE** First even, last even,

**FOLO** First odd, last odd,

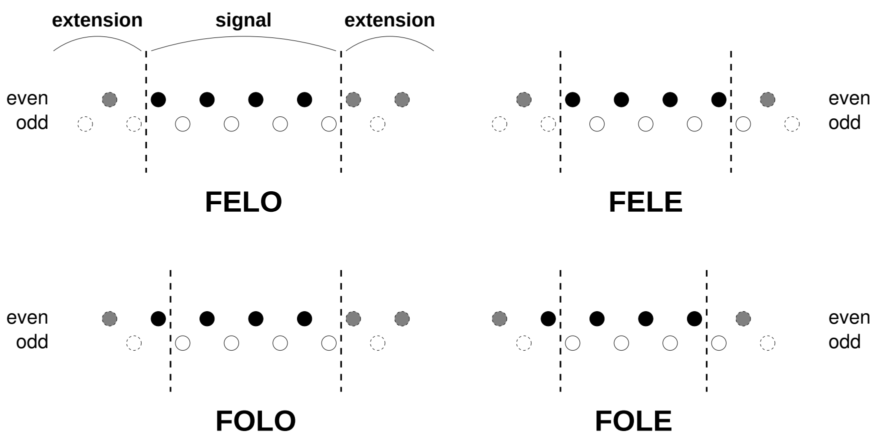



Figure 3.9: The four cases of symmetric signal extension using lifting.

**FOLE** First odd, last even.

This is another nice property of the second-generation viewpoint of the lifting scheme: symmetric extension is always possible! We are no longer limited to symmetric (biorthogonal) wavelets. Even if one has a lifting decomposition for a non-symmetric wavelet filter bank (e.g. orthogonal Daubechies wavelets), the implementation using lifting steps can use symmetric extension.

## 3.6 Conclusions

The lifting scheme allows to implement reversible integer wavelet transforms in an efficient way. All classical wavelet transforms can be implemented using the lifting scheme by a factorization in lifting steps. We worked out the details of integer wavelet transforms for a class of biorthogonal wavelets (Cohen-Daubechies-Feauveau).

# Chapter 4

# Wavelet-Based Image Processing

In this chapter we will first explain how digital images are represented inside a computer system. After that we will discuss some image processing operations that give superior results when implemented using wavelets, or that just fit within the wavelet framework. We implemented all these operations in the software library WAILI, which will be described in Chapter 5.

One of the most important image processing operations is image compression. Wavelets also can provide for this.

## 4.1 Digital Images

A grey scale image can be considered as a rectangle where every point in the image has a luminance in the continuous range from black to white. To store an image in a computer, we need a finite precision representation. This is done in two steps: discretization and quantization.

### 4.1.1 Discretization

First we need to discretize the image to a finite number of pixels. A pixel is a small rectangle with a constant luminance. As a result we obtain a matrix of pixels. The number of pixels per length unit is called the resolution. The higher the resolution the better the quality of the discretized image, but the more memory needed.

### 4.1.2 Quantization

Every pixel has a luminance in the continuous range from black to white. To obtain a finite precision representation we restrict the pixel values to a finite number of quantization levels. Continuous grey scale images are often quantized to 256 (8 bit) levels: black is 0, white by 255. Grey values between black and white are mapped to an integer number in the range 0 to 255. More quantization levels improve the quality of the quantized image, but again require more memory.

We end up with a discrete representation of the original grey scale image, consisting of a matrix of integer luminance values, which can be manipulated by a computer.

### 4.1.3 Color images

Color images can be represented by three matrices: one for each primary color component red ($R$), green ($G$) and blue ($B$). Using these three primary colors one can represent most of the visible colors.

However, the human eye is much more sensitive to luminance differences than to color differences. One often uses a different decomposition, similar to the color encoding scheme used in the color television system. Now a color image is represented by the following three matrices:

**Luminance matrix $Y$:** This is the luminance of the image, a weighted mean of the three primary color components:

$$Y = c_R \cdot R + c_G \cdot G + c_B \cdot B,$$

with $c_R + c_G + c_B = 1$ (typically $c_R = 0.3$, $c_G = 0.59$ en $c_B = 0.11$).

**Chrominance matrix $U$:** This is the red difference, i.e. the difference of the red color component and the luminance:

$$U = R - Y.$$

**Chrominance matrix $V$:** This is the blue difference, i.e. the difference of the blue color component and the luminance:

$$V = B - Y.$$

Thus the luminance matrix $Y$ is a grey scale version of the original color image, while the two chrominance matrices $U$ and $V$ contain the color information. The choice of the red and blue difference (and not the green difference) as the chrominance matrices was influenced by the higher sensitivity of the human eye to green light. Other linear combinations of the 3 primary color components are also in common use.

The original primary color components $R$, $G$ and $B$ can be easily reconstructed from the luminance and chrominance matrices.

### 4.1.4 Generalization

To make a more general statement, a digital image contains one or more matrices (channels) of integer values, one for each color component. Note that channels are not limited to the traditional three color components. They can contain any spectral data. Each channel can have a different size, to accommodate for subsampled chrominance channels, which are commonly used in e.g. broadcast applications.

### 4.1.5 $L^*a^*b^*$

Other color representations are also possible. Non-linear color representations like $L^*a^*b^*$ take into account the non-linearity in the response of the human eye to light [38].

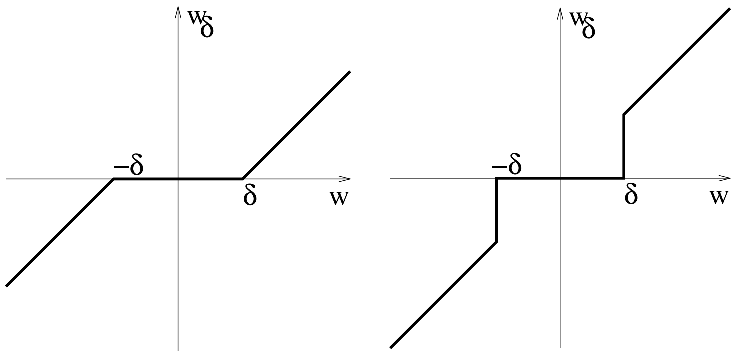
## 4.2 Wavelet Denoising

### 4.2.1 Thresholding

An important application domain of wavelet theory is denoising. In a classical wavelet-based noise reduction algorithm first a discrete wavelet transform is computed. Then, the input wavelet coefficients are manipulated and finally an inverse transform yields the result. The *manipulation* of a coefficient often depends on the *class* to which the coefficient belongs. Therefore the algorithm first classifies the coefficients before the actual manipulation. For the allocation of each coefficient to a class, the algorithm uses a *criterion*, which is often binary: it divides the coefficients into two classes. The first group contains important, regular coefficients, while the other group of coefficients is cataloged as "not essential and too much affected by noise".

The most straightforward procedure uses the *absolute value* of the wavelet coefficients as a regularity measure: a coefficient close to zero contains little information and is relatively strongly influenced by noise. This is true because we consider the high-pass bands only and because small local variations in a signal are usually caused by noise. The basic idea is the same as for compression algorithms: the method assumes that a regular image can be represented by a small number of large coefficients. Donoho e.a. [18] showed that this method has statistical optimality properties. The absolute value is however a local measure: every coefficient is considered separately. More sophisticated algorithms often use a more global criterion.

When using such a threshold procedure, all wavelet coefficients with absolute value below a certain threshold $\delta$ are classified as "noisy". The algorithm replaces them by zero. The coefficients above the threshold are shrunk with the same value $\delta$. Figure 4.1 compares this "soft-thresholding" operation with the "hard-thresholding" alternative. The hard-thresholding procedure does not shrink




(a) (b)

Figure 4.1: Noise reduction by wavelet shrinking. (a) Soft-thresholding: a wavelet coefficient $w$ with an absolute value below the threshold $\delta$ is replaced by 0. Coefficients with higher absolute values are shrunk. (b) Hard-thresholding: Coefficients with an absolute value above the threshold are kept.

the coefficients with absolute value above the threshold. Although at first sight this may seem a more natural approach, soft-thresholding is a more continuous operation, and it is mathematically more tractable.

### 4.2.2 Threshold selection

The central issue in a threshold procedure is the selection of an appropriate threshold. If this threshold is too small, the result is still noisy. On the other hand, a large threshold also removes important image features and thus causes a bias. It is intuitively clear that the more noise is expected for a given coefficient, the higher the optimal threshold should be. Most procedures compute one threshold for a group of coefficients, based on the statistical properties of this group. Therefore, they assume additive, stationary noise within this group. If the input noise is additive and stationary, then it is easy to prove that the noise on the wavelet coefficients is additive and stationary within each subband and at each resolution level [27, 23]. This property does not hold true exactly for integer transforms, due to the non-linearity introduced by the rounding operations.

The optimal threshold for subband $j$ $\delta_{j,\text{opt}}$ minimizes the mean square error of the result as compared with the unknown, noise-free coefficients:

$$R_j(\delta) = \frac{1}{N_j} ||w_{j,\delta} - v_j||^2,$$

where $N_j$ is the number of coefficients in subband $j$, $w_{j,\delta}$ is the vector of thresholded noisy coefficients and $v_j$ is the vector of noise-free coefficients. The expected value of this error is called *Risk* function.

The "universal threshold" $\delta_j^U$ by Donoho and Johnstone [16] explicitly proposes a threshold value proportional to the amount of noise $\sigma_j$.

$$\delta_j^U = \sqrt{2 \log N_j}\, \sigma_j.$$

In fact, for finite $N_j$ this is an upper bound for the optimal threshold. This choice is asymptotically optimal: for infinitely long signals it is optimal, and thus the signal length has to be sufficiently large to obtain good results.

Another well known threshold selection procedure is based on "Stein's Unbiased Risk Estimator" (SURE) [17].

Based on the work of Jansen, we use the minimizer of the "Generalized Cross Validation" (GCV) function [60, 24, 23, 25]

$$GCV_j(\delta) = \frac{\frac{1}{N_j} \|w_j - w_{j,\delta}\|^2}{\left[\frac{N_{j0}}{N_j}\right]^2},$$

where $N_{j0}$ is the number of wavelet coefficients that were replaced by zero. Unlike most other selection procedures it does not need an estimate of the actual noise level $\sigma_j$ in subband $j$. Moreover, computational complexity is linear. The method is also asymptotically optimal, but for finite $N_j$ it gives better estimates for the optimal threshold than $\delta_j^U$.

Figure 4.2 illustrates the principle of generalized cross validation for a typical case. Artificial, stationary, colored noise was added to a clean test image. Both the noisy and noise-free image were transformed, using an integer version of the biorthogonal Cohen-Daubechies-Feauveau $(2,2)$ wavelet algorithm. At each level and for each subband, the GCV as a function of the threshold was compared with the exact mean square error. Although the conditions for a successful application of GCV are not strictly fulfilled by the non-linear character of the integer transform, both functions have approximately the same minimum. Note that in practical problems, a comparison with the exact mean square error is impossible, since the noise-free image is unknown.

### 4.2.3 Integer Minimization Problem

In a denoising implementation for a classical floating point wavelet transform the process of finding the threshold value that minimizes the GCV is a floating point minimization problem. Hence the minimization process is stopped when a given precision is reached.

However, in our case we use integer wavelets transform. Only a limited set of threshold values are meaningful due to the finite precision of the integer data type. This means that we can always calculate the exact (integer) threshold value that minimizes the GCV in a few iteration steps. In our implementation, we used an integer variant of the Fibonacci minimizer [20].

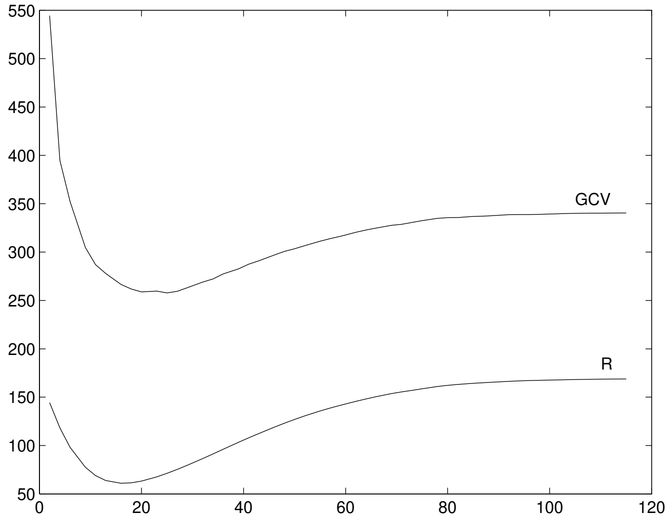



Figure 4.2: *GCV* and mean square error of the result in function of the threshold $\delta$. Both functions have approximately the same minimum.

## 4.3 Image Scaling

After one step of a wavelet transform on an image, it is clearly visible that the low-pass subband is a downscaled version (by a factor of 2) of the original image. Thus a wavelet transform can be used to downscale an image with a factor of $2^n$ (Figure 4.3). Because of the low-pass filtering, no aliasing occurs.

Consequently, the inverse wavelet transform can be used to upscale an image with a factor of $2^n$, if we consider all wavelet coefficients in the high-pass subbands to be zero[1]. This gives superior results than simple pixel replication (Figure 4.4), since no blocking effect occurs.

Hence we can down- or upscale an image by a factor of $2^n$ in the wavelet domain, by removing or adding subbands.

To scale an image with a factor that is not a power of two, one can scale it using a wavelet transform until it is larger than the desired image size, and use classical (linear) interpolation to downscale it until the image has the wanted size [62]. As a side note, upscaling an image using the CDF $(2,x)$ wavelet transform is equivalent to upscaling using linear interpolation (Figure 4.5). Other wavelet transforms correspond to other interpolation rules (e.g. CDF $(4,x)$ is equivalent to cubic in-

[1]It could be better to try to predict the coefficients in the high-pass subbands based on the values of the coefficients in the subbands at the lower resolution levels instead of simply setting them to zero, but this is much more difficult.

256 × 256 (original) 128 × 128 64 × 64 32 × 32

Figure 4.3: Downscaling of an image using simple subsampling (upper row) and using the CDF $(2,2)$ (middle row) and CDF $(4,4)$ wavelet transforms (lower row). Aliasing occurs with simple subsampling, while the wavelet-based method does not suffer from aliasing because the low-pass filter in a wavelet transform smoothes the image. These images were created using WAILI.


Figure 4.4: Upscaling of an image using pixel replication (upper row) and using the CDF $(2,2)$ (middle row) and CDF $(4,4)$ wavelet transforms (lower row). Pixel replication shows a clearly visible blocking effect, which is not present when using the wavelet-based method. These images were created using WAILI.

(a) (b)

Figure 4.5: Upscaling of an image with a factor 5 using (a) pixel replication and using (b) the CDF $(2,2)$ wavelet transform.

terpolation).

## 4.4 Crop and Merge

Accessing a rectangular part of an image is trivial in the spatial domain. Although wavelets are localized in the spatial domain, crop and merge operations on wavelet transformed images are much more difficult to implement because the wavelet transform is not translation invariant (see Section 2.4). Due to the importance of crop and merge operations we had to circumvent this problem, and our solution will be given in Section 5.3.1.

## 4.5 Image Compression

### 4.5.1 Why Compression?

The amount of digital images we use is growing continuously, and video and television transmission becomes more and more digital. Digital images become larger and larger, because there is an increased need for large scenes or because the required resolution increases. On the other hand, we use more and more digital image sequences in multi-media applications.

To illustrate the need for compression, we give some examples:

- To store a color image of a moderate size, e.g. $512 \times 512$ pixels, one needs 0.75 MB of disk space.
- A 35 mm slide digitized with a resolution of 12 $\mu$m requires 18 MB.
- One second of digital PAL video requires 27 MB.

- A digital orthophotomap of a Belgian province, with a resolution of one pixel per $m^2$ requires about 15 GB (Belgium takes 90 GB).

To store these images, and make them available over networks (e.g. the Internet), compression techniques are needed. The use of compression techniques also allows the processing of larger images using the current hardware, while the processing of images of a given size can be performed on cheaper hardware platforms.

### 4.5.2 Compression Techniques

**Lossless versus Lossy Compression**

Compression methods can be divided in two classes: lossless and lossy compression techniques:

- *Lossless compression* guarantees that the original signal can be reconstructed without any errors. This is important for applications like the compression of text. For images, lossless compression is often used as the second step, after the lossy part.
- With *lossy compression*, we can obtain higher compression rates by not requiring the exact data to be reconstructed. Indeed, because the human visual system is not sensitive to some kinds of errors, the compression potential is much higher when we allow for small reconstruction errors.

Although the integer wavelet transform can be used for lossless compression due to its 100% invertible nature (in contrast to floating point wavelet transform implementations), image compression will usually be lossy due to the high compression rates that are required, except in sensitive applications areas like medical imaging where any data loss is not acceptable.

**A Lossy Compression Scheme**

Lossy compression schemes are commonly based on transform coding. Such a coding scheme consists of three steps, as is shown in Figure 4.6.

**Transform** The original signal is transformed using a decorrelating transform. If one can represent the original signal $f$ as

$$f = \sum_i a_i \psi_i,$$

using only a few basis functions $\psi_i$, one needs only the few coefficients $a_i$ to reconstruct the signal $f$. Popular transforms are the Fast Fourier Transform (FFT) and its derivatives (e.g. the Discrete Cosine Transform (DCT)), and of course wavelet transforms.

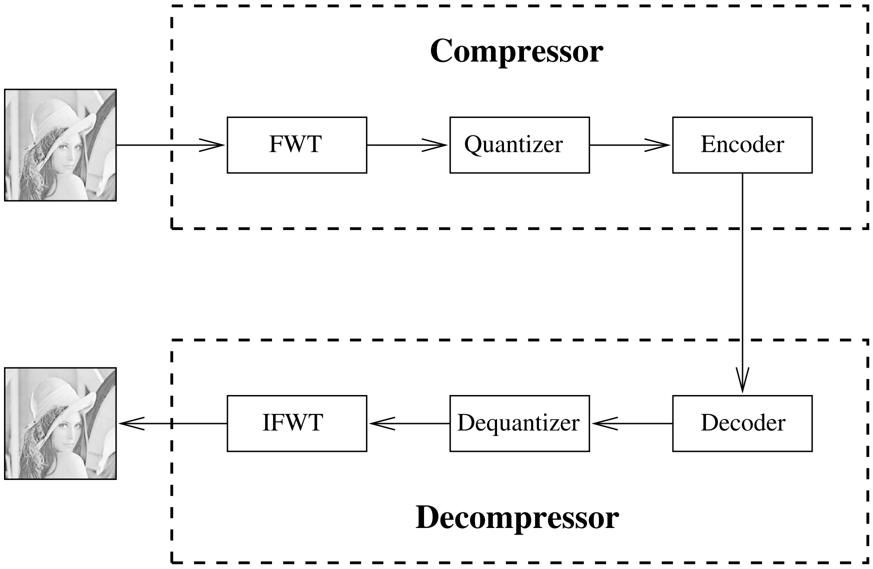



Figure 4.6: General scheme of a lossless image compression algorithm using wavelet transforms.

**Quantization** The quantization step defines which coefficients will be neglected, and with which accuracy the retained coefficients will be stored. More important coefficients will be represented with a higher accuracy. The least important coefficients (very close to zero) can be replaced by zero. This is the lossy step in a lossy compression scheme, if we assume the transform step is invertible without any loss.

Mathematically, Parseval's identity (Equation 2.4, in the case of an orthogonal basis), or the fact that we have a Riesz basis (Equation 2.5, in the biorthogonal case), guarantee that the largest coefficients are the most important coefficients. Neglecting a coefficient with a small magnitude gives a reconstruction error proportional to the magnitude of the coefficient.

**Encoding** Afterwards the quantized coefficients (symbols) will be stored in a bit stream in an efficient way. Usually one uses some sort of entropy encoding to convert the coefficients to a bit stream.

### Color Images

Color images have additional compression potential because the human eye is less sensitive to changes in chrominance than to changes in luminance. Thus color images can be compressed with higher compression rates than the corresponding grey scale images.

### 4.5.3 Features of Image Compression using Wavelets

The key features of wavelet-based compression schemes are:

- Wavelets provide good compression ratios [15, 5]. Especially for high compression ratios, wavelets perform much better than competing technologies like JPEG [59], both in terms of signal-to-noise ratio and visual quality. Unlike JPEG, they show no blocking effect but allow for a graceful degradation of the whole image quality, while preserving the important details of the image. The next version of the JPEG standard (JPEG 2000) will incorporate wavelet based compression techniques.

- The wavelet transform is a fast operation (linear to the amount of data), especially when implemented using the lifting scheme. The wavelet transform is symmetric: both the forward and the inverse transform have the same complexity, allowing fast compression and decompression routines.

- Multi-resolution allows for progressive transmission and zooming, without the need for extra storage. One can e.g. first transmit a thumbnail image, and gradually transmit and decompress more data to increase the resolution and overall image quality.

- Wavelets can not only be used for image compression, but also for various image processing operations. The possibility to combine image processing and compression is a very appealing factor. However, image processing cannot be done on the final encoded data stream: it must be done before the wavelet coefficients are quantized or encoded. But even then we win because the wavelet transform is a common factor in both image processing and image compression.

### 4.5.4 Zerotree Encoding

Most compression methods are targeted at "small" images. In that case the individual subbands are too small to get a high compression gain from compressing them separately. Furthermore, even after the decorrelating step of a wavelet transform, there is still correlation present among spatially related wavelet coefficients in subbands at different resolution levels in the multi-resolution decomposition. That's why compression algorithms based on wavelets often use a variant of zerotree encoding [42, 39].

Zerotree encoding combines the subbands to exploit the remaining correlation. If a coefficient at a low-resolution level is small (or even zero), chances are high

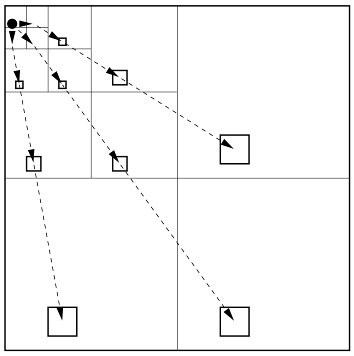


Figure 4.7: Zerotree encoding: a tree of coefficients at different resolution levels that correspond to one low-pass coefficient.

that the corresponding coefficients at higher resolution levels are also small (or zero). In that case the whole tree of coefficients that offsprings from one low-pass coefficient can be encoded in an efficient way (see Figure 4.7).

Zerotree encoding permits the construction of an embedded coder. Such a coder produces a bit stream in which the bits are generated in order of importance, so it can be truncated wherever desired, to meet specific bandwidth or storage requirements, while providing the best reconstruction quality for that size of the truncated bit stream. This is a form of progressive transmission: first the most important parts of the image are transmitted, while later parts of the bit stream contain the necessary information to gradually increase the image resolution and overall quality.

## 4.6 Conclusions

Many image processing operations like noise reduction and image scaling can be done on wavelet transformed images. Furthermore wavelets can also be used for image compression, giving better performance (higher signal-to-noise ratios, better visual quality and higher compression rates) than other compression methods. This combination of image processing and compression is an important feature of wavelet-based processing.

# Chapter 5

# Design of the Software Library WAILI

WAILI[1] (Wavelets with Integer Lifting) is a software library — written in C++ —providing wavelet transforms and wavelet-based operations on two-dimensional images of various kinds [57]. We developed WAILI in the scope of the project "Wavelet-Based Interactive Video Communication and Image Database Consulting", with applications situated in image processing.

In this chapter we will discuss the goals that had to be met in WAILI. We will report about the design decisions that were taken to meet these goals as close as possible. Furthermore the internal structure of the library is explained.

WAILI is available in source form for research purposes [56]. Currently it is in use at ca. 250 research institutes.

## 5.1 Design Criteria

The project we designed WAILI for required the development of a software library for wavelet transforms and wavelet-based image processing operations. The library had to possess the following properties:

- support arbitrary image sizes,
- support images containing multiple channels, possibly each with a different size,

[1] It is written *WAILI*, but pronounced *WILY*, except in Dutch, where it's pronounced *PIEFPAK*.

- use an integer lossless wavelet transform, based on the lifting scheme,
- allow the transform of both the columns and rows of an image, or either the columns or the rows.
- allow to cut a rectangular subimage out of a larger transformed image, and to replace a rectangular area in a large wavelet-transformed image,
- allow noise reduction,
- allow simple compression, using thresholding of subbands,
- provide edge enhancement of images, by scaling the wavelet coefficients in a particular subband,
- provides some simple image operations like addition and subtraction.
- be object oriented (written in C++, because C++ offers the combination of flexibility, fast execution speeds and portability),

## 5.2 WAILI: Wavelets with Integer Lifting

### 5.2.1 The Lifting Scheme

We chose to implement two-dimensional tensor product wavelet transforms using the integer version of the lifting scheme.

Although the lifting scheme allows to transform signals with a finite length without extending the signal, we did not choose to take this approach. Instead we use the classical symmetric extension [4] because it is easier to implement and suffices for the applications we have in mind.

All arithmetic operations are done with 16 bit input data and 16 bit results. This is sufficient for applications where the input data is 8 bit wide. Of course this can easily be changed if necessary.

### 5.2.2 Cohen-Daubechies-Feauveau Biorthogonal Wavelets

The wavelets we use are a subclass of the Cohen-Daubechies-Feauveau family of biorthogonal wavelets [9]. We have chosen these because of their key benefits, which are:

- They have finite support. This preserves the locality of image features.
- The scaling function $\varphi(x)$ is always symmetric, and the wavelet function $\psi(x)$ is always symmetric or antisymmetric. This is important for image processing operations since the detection of features in an image does not depend on their left/right or up/down orientation.

- Its filter coefficients are dyadic, i.e. they are of the form $z/2^n$, with $z \in \mathbf{Z}$ and $n \in \mathbf{N}$. This simplifies the implementation. But unfortunately this feature is not always preserved by the decomposition in lifting steps.

We chose not to use wavelets with more than 6 vanishing moments to restrict the filter lengths. Longer filters have less locality and thus perform worse in image processing applications, in spite of their increase in smoothness.

WAILI provides the following wavelet transforms of this family. The notation $(n,\tilde{n})$ means that the primal wavelet has $n$ vanishing moments, while the dual wavelet has $\tilde{n}$ vanishing moments:

**(1, x):** $(1,1)$, $(1,3)$, $(1,5)$

**(2, x):** $(2,2)$, $(2,4)$, $(2,6)$

**(4, x):** $(4,2)$, $(4,4)$, $(4,6)$

We deliberately did not implement any of the $(3,x)$ or $(5,x)$ wavelet transforms because their lifting steps require divisions by 3 or 5 — which are difficult to implement in hardware —, or because the normalization factor $K_1$ requires a division. $(6,x)$ are not implemented either because they require more than 16 bits (for 8 bit input data).

### 5.2.3 Image Size Constraints

Ideal signals have a length of $2^n$ samples. Only in that case we can apply the wavelet transform for $n$ consecutive levels and guarantee that the low-pass and high-pass bands have the same size at each resolution level. Unfortunately we do not live in an ideal world, and real-world signals have arbitrary length. If the signal length is odd, or becomes odd at some resolution level, either the low-pass or high-pass band will have one more sample than the other. Which band will contain the most samples must be defined by some convention. A first solution is to standardize on the number of "even" samples (in lifting terminology) not being smaller than the number of "odd" samples. However, we did not take this approach. The reasons for this will become clear in Section 5.3.1, which discusses the implementation of crop and merge operations in WAILI.

Extended to two dimensions, ideal images are square with a width of $2^n$ pixels. In that case one can perform the same number of transform levels in both the horizontal and vertical directions. But real images are usually rectangular, and can have aspect ratios (width/height) that differ a lot from 1. This means the number of transform levels can be different for the columns and rows of the image. This adds even more complexity.

### 5.2.4 Images and Wavelet Transforms

Performing wavelet transforms on images involves several components (C++ classes).

#### *Image* Objects

An *Image* consists of one or more independent channels, thus allowing for different sizes and wavelet transform types per channel. No interpretation or format is imposed on the channels and its data. The actual meaning of the image data can be freely chosen by the user. Examples are grey scale, RGB, YUV or Lab color, etc.

#### *Channel* Objects

The basic building block of the library is the *Channel*. A channel is a rectangular matrix containing one-valued pixels. A channel can be non-transformed (an *NTChannel*), or wavelet-transformed (an *LChannel* — lifted channel).

Since a wavelet transform is a recursive transform, a LChannel contains some subchannels (subbands), which can be either non-transformed or wavelet-transformed. The number of subchannels in an LChannel depends on the type of wavelet transform. One can have the following combinations (all of these are derived from the Channel abstract base class):

***LChannelCR*** Obtained by transforming both the columns and rows of an NTChannel. As a result, one has 4 subbands:

- **LL** Low-pass band in both the horizontal and the vertical direction,
- **LH** Low-pass band in the vertical direction, high-pass in the horizontal direction,
- **HL** High-pass band in the vertical direction, low-pass in the horizontal direction,
- **HH** High-pass band in both the horizontal and the vertical direction.

***LChannelC*** Obtained by transforming only the columns of an NTChannel. As a result, one has 2 subbands:

- **L** Low-pass band in the vertical direction,
- **H** High-pass band in the vertical direction.

***LChannelR*** Obtained by transforming only the rows of an NTChannel. As a result, one has 2 subbands:

- **L** Low-pass band in the horizontal direction,
- **H** High-pass band in the horizontal direction.

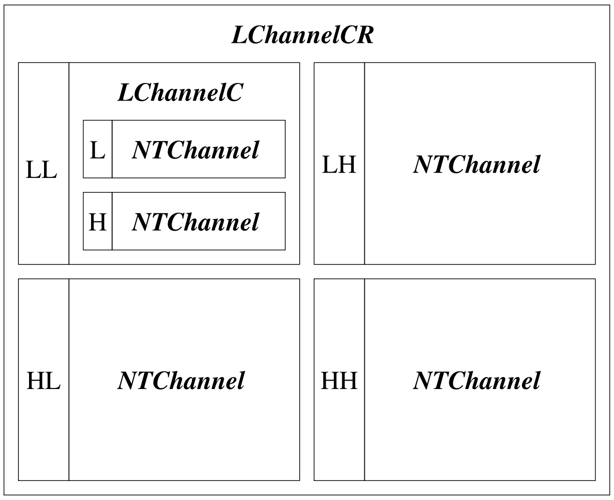



Figure 5.1: Example of a channel after two transform levels. In the first step both the columns and the rows are transformed, in the second step only the columns are transformed.

Figure 5.1 shows an example of a channel after two transform levels. And of course one can use a different wavelet at each transform level. A LChannel remembers the wavelet transform that was used to be able to perform the inverse transform later.

### *Wavelet* Objects

A *Wavelet* represents the filters and lifting steps associated with a specific wavelet transform. Different wavelet transforms are implemented through inheritance. Some wavelet transforms of the Cohen-Daubechies-Feauveau family are implemented. One can add his own favorite wavelet transform if a decomposition in integer lifting steps for it is available, simply by deriving a new class from the Wavelet class that calls the lifting operations with the appropriate lifting coefficients.

### *Lifting* Objects

A *Lifting* object performs lifting steps. The provided lifting operations are generic in the sense that one has to pass the lifting coefficients to be used. But due to the object oriented nature of C++ one can easily plug in highly optimized versions with hardcoded lifting coefficients for specific wavelet transforms to achieve maximum

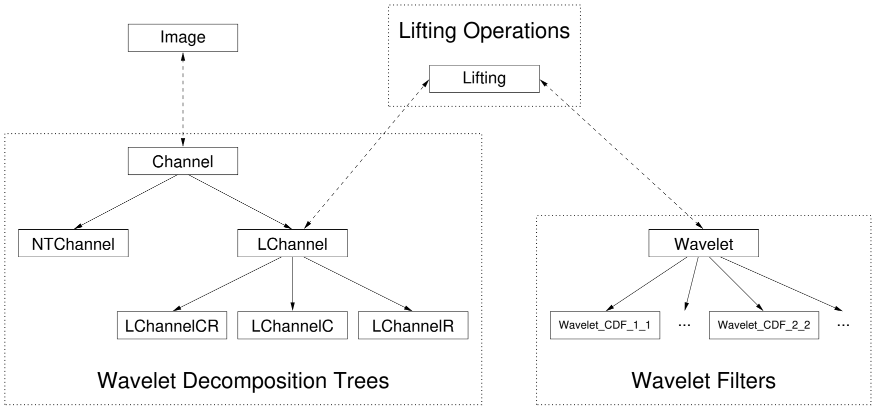



Figure 5.2: Relations among the major components of WAILI. Inheritance relations are indicated by solid arrows, computational relations are indicated by dashed arrows.

performance.

The relations among the major components of WAILI are shown in Figure 5.2.

## 5.3 Wavelet-Based Operations

Within WAILI, we implemented some image processing operations on wavelet-transformed images.

### 5.3.1 Crop and Merge on Wavelet-Transformed Images

The wavelet and scaling functions are local in the spatial and the frequency domain. If one wants to cut a small rectangle $B$ out of a large image $A$, one can exploit the spatial locality of the basis functions.

At each transform level $l$, we only need the coefficients that influence the pixel values of $B$. This is a rectangle of coefficients that is only slightly larger than $B$ (scaled to the transform level $l$). The extra coefficients are needed due to the overlap of the basis functions at the borders of $B$.

Using these technique we efficiently implemented the following operations on wavelet transformed images:

**Crop** Get a rectangular part of an image, at a specified position and with a specified size.

**Merge** Paste a rectangular image into a larger image, at a specified position.

The major point of difficulty here is that a wavelet transform is not translation invariant, due to the different treatment of even and odd samples in the lifting scheme (see Section 2.4). We came up with the following scheme. If each transform level is considered independently, one step of a wavelet transform is translation invariant if the translation is limited to an even number of pixels. Thus we associate with every image global *coordinates* (a horizontal and vertical offset for the upper left pixel), depending on the transform level. Hence we no longer always put the upper left pixel of the image in the even/even data set during the lazy wavelet transform step, but instead we partition all pixels in the image based on the parities of their global coordinates. This is equivalent to considering two (actually four because of the two-dimensional aspect) versions of the wavelet transform: an "even" and an "odd' version. This way we can merge two images without retransforming one of them if the parities of the offsets match at each level. If they do not match, we have to retransform one of them. The same is done for the crop operation.

The main idea behind this scheme is that in many cases the coordinates of the subimage that will be pasted into another image are *known in advance*, so it can be transformed correctly. An example of this is the creation of one large image by gluing together several separately created subimages.

### 5.3.2 Scaling of Images

In Section 4.3 we described how images can be scaled in the wavelet domain. If a digital image is transformed, the low-pass band will be a low-resolution version of the original image. In this way we can downscale images by a power of 2. Alternatively, we can use the inverse transform — with all high-pass coefficients being zero — to upscale an image by a power of 2.

If we want to scale an image with a factor that is not a power of two, we scale the image using a power of 2 factor such that the result is larger than the wanted result. After that a classical interpolation algorithm is used to downscale the image to the wanted size.

### 5.3.3 Noise Reduction

Wavelet soft-thresholding is used to reduce additive, stationary noise. We select a threshold for each transform level and for each subband separately. To select these thresholds we use generalized cross validation (See Section 4.2.2). Since this is an asymptotical method, it only performs well for sufficiently large subbands. Experiments pointed out that 1000 coefficients in a subband is about the minimum for a good threshold estimation. Therefore, WAILI does not perform the denoising operation on subbands with less than 1000 coefficients. These are the subbands at coarse levels. In most cases, most of the noise appears at finer levels, and so, leaving the coarse levels untouched generally gives good results.

## 5.4 Implementation Issues

### 5.4.1 In-Place Transform

A wavelet transform implementation using the lifting scheme consists of two parts:

- A splitting step (lazy wavelet transform)
- One or more lifting steps

The splitting step is just a partitioning of the data and does not require a physical data transport, and the lifting steps operate on the two partitions only. Thus the wavelet transform can be implemented "in-place" by using the lifting scheme. No additional memory is required. The same is true for the inverse transform. Traditional filter bank algorithms do require additional memory, to store both the original and the transformed signal during the computations.

Because after each transform level one iterates on the "even" partition, one will get an "interleaved" representation in the wavelet domain. Indeed, after two transform levels the low-pass subband is formed by every fourth coefficient, after three transform levels by every eight coefficient, and so on. This interleaving causes a non-optimal memory access pattern because the wavelet transform operates on data that is not contiguous in memory. However, the locality in space is preserved: neighboring coefficients correspond to neighboring features of the original signal.

Alternatively the wavelet coefficients can be reordered to store the coefficients belonging to the same subband together.

### 5.4.2 The Two-Dimensional Wavelet Transform: Mallat Ordering

The two-dimensional wavelet transform of a matrix is given by transforming the columns and rows of the matrix independently using the one-dimensional transform. This results in a tensor product transform at each transform level. Instead of with 2 subbands (low-pass and high-pass), one will end up with 4 subbands: `LL` — low-pass in both the horizontal and vertical directions, `LH` — low-pass in the vertical, high-pass in the horizontal direction, `HL` and `HH`. When iterated on the LL subband, the result is a multi-resolution decomposition as shown in Figure 5.3. This ordering of the subbands at the different resolution levels is called the "Mallat" ordering [35].

Since the data belonging to the same subband are stored together, the Mallat ordering is advantageous if one is interested in multi-resolution analysis, or to provide progressive transmission. One can easily start with a low-resolution representation of the original image and gradually increase the resolution by transmitting additional high-pass subbands.

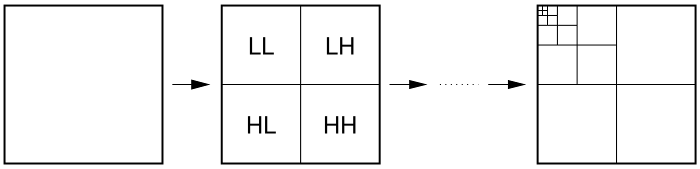



Figure 5.3: The two-dimensional wavelet transform: iteration on the `LL` subband, showing the wavelet coefficients in "Mallat" ordering.

In our applications we are interested in multi-resolution analysis and in progressive transmission, and we wanted to use the Mallat ordering. Hence we chose not to implement the lifting scheme "in-place", but to physically split the input data into two partitions (actually four, due to the two dimensions of images) in the splitting step. This does require additional memory, but has the additional advantage of operating on contiguous data, thus avoiding trashing of the data cache.

### 5.4.3 Transform Order

Normally the order of transform (first columns, then rows, or vice versa) at each transform level does not matter, but because of the rounding in the lifting steps, transform order does become important. Thus we always transform the columns first, followed by the rows. For the inverse transform, the order is inverted.

#### Lifting Steps on the Columns of a Matrix

Since matrices are stored in memory row by row, lifting the columns one after each other is not a good idea. By taking advantage of locality in memory and caching, it is faster to vectorize the calculations so they operate on rows. We use the following algorithm:

for all rows $r$
    for all columns $c$
        $x(c,r) \leftarrow \{a_0 \cdot y(c,r-1) + a_1 \cdot \mathrm{y}(c,r)\}$

This assumes we want to lift the coefficients in $x$ using the coefficients in $y$, with lifting coefficients $a_0$ and $a_1$. Of course we have to take special care at the borders of the matrix.

#### Lifting Steps on the Rows of a Matrix

This can be done in the straightforward manner:

for all rows $r$
  for all columns $c$
    $x(c,r) \leftarrow \{a_0 \cdot y(c-1,r) + a_1 \cdot y(c,r)\}$

### 5.4.4 Normalization of the Wavelet Coefficients

Ideally we would like the normalization factor $K_1$ (see Equation 3.1) for the low-pass data to be $\sqrt{2}$ in all cases. Indeed, then the dynamic range of the low-pass data remains the same if we forget about the normalization factor. If the input samples are 8-bit values, the low-pass data after applying a wavelet transform step (the "mean" values) will be approx. 8-bit values too. However, there may be a slight bit growth in the low-pass band because the low-pass filter operation calculates a non-convex linear combination of the input data.

Of course we will have to remember the accumulated normalization factors for every subband if we want to process the wavelet-transformed data later. One way to do this is to consider each wavelet coefficient (i.e. the value after filtering) to be the mantissa $m$ of a floating point number in base $\sqrt{2}$. The exponent $e$ of this floating point number is $\log_{\sqrt{2}}$ of the accumulated normalization factor, which depends on the subband. Thus the real normalized value of each coefficient becomes:

$$x = m \cdot \left(\sqrt{2}\right)^e .$$

A special and simpler case of this is the two-dimensional wavelet transform where we always apply the transform to both the rows and the columns of a matrix. Since $\sqrt{2} \cdot \sqrt{2} = 2$, the normalization factor becomes 2, and the floating point numbers in base $\sqrt{2}$ can easily be converted to base 2.

**Example**

For CDF $(2,2)$, the lifting steps used are:

$$\begin{aligned} d_i &\leftarrow d_i - \left\{ \frac{s_i + s_{i+1}}{2} \right\}, \\ s_i &\leftarrow s_i + \left\{ \frac{d_{i-1} + d_i}{4} \right\}, \end{aligned}$$

Implying normalization factors

$$\begin{aligned} K_1 &= \sqrt{2} = \left(\sqrt{2}\right)^{+1}, \\ K_2 &= \frac{\sqrt{2}}{2} = \left(\sqrt{2}\right)^{-1}. \end{aligned}$$

In two dimensions, we have 4 subbands $(LL, HL, LH, HH)$ after one transform step, with normalizations factors:

$$\begin{aligned}
K_{LL} &= K_1 \cdot K_1 = \left(\sqrt{2}\right)^{+2} = 2 \\
K_{LH} &= K_1 \cdot K_2 = \left(\sqrt{2}\right)^{+0} = 1 \\
K_{HL} &= K_2 \cdot K_1 = \left(\sqrt{2}\right)^{+0} = 1 \\
K_{HH} &= K_2 \cdot K_2 = \left(\sqrt{2}\right)^{-2} = \frac{1}{2}
\end{aligned}$$

For further transform steps, we have to take into account that the normalization factors accumulate due to the iteration on the low-pass subbands. Applying two wavelet transform steps, as shown in fig. 5.4, gives the following results. $x_S^{(l)}$ and $m_S^{(l)}$ represent a coefficient value respectively a mantissa in subband $S$ at decomposition level $l$.

$$\begin{array}{llcll}
\boxed{\textbf{Level 0}} & LL : x_{LL}^{(0)} & = & m_{LL}^{(0)} & = m_{LL} \cdot 2^{+0} \\
\boxed{\textbf{Level 1}} & LL : x_{LL}^{(1)} & = & m_{LL}^{(1)} \cdot \left(\sqrt{2}\right)^{+2} & = m_{LL}^{(1)} \cdot 2^{+1} \\
& LH : x_{LH}^{(1)} & = & m_{LH}^{(1)} \cdot \left(\sqrt{2}\right)^{+0} & = m_{LH}^{(1)} \cdot 2^{+0} \\
& HL : x_{HL}^{(1)} & = & m_{HL}^{(1)} \cdot \left(\sqrt{2}\right)^{+0} & = m_{HL}^{(1)} \cdot 2^{+0} \\
& HH : x_{HH}^{(1)} & = & m_{HH}^{(1)} \cdot \left(\sqrt{2}\right)^{-2} & = m_{HH}^{(1)} \cdot 2^{-1} \\
\boxed{\textbf{Level 2}} & LL : x_{LL}^{(2)} & = & m_{LL}^{(2)} \cdot \left(\sqrt{2}\right)^{+2} \cdot \left(\sqrt{2}\right)^{+2} & = m_{LL}^{(2)} \cdot 2^{+2} \\
& LH : x_{LH}^{(2)} & = & m_{LH}^{(2)} \cdot \left(\sqrt{2}\right)^{+2} \cdot \left(\sqrt{2}\right)^{+0} & = m_{LH}^{(2)} \cdot 2^{+1} \\
& HL : x_{HL}^{(2)} & = & m_{HL}^{(2)} \cdot \left(\sqrt{2}\right)^{+2} \cdot \left(\sqrt{2}\right)^{+0} & = m_{HL}^{(2)} \cdot 2^{+1} \\
& HH : x_{HH}^{(2)} & = & m_{HH}^{(2)} \cdot \left(\sqrt{2}\right)^{+2} \cdot \left(\sqrt{2}\right)^{-2} & = m_{HH}^{(2)} \cdot 2^{+0}
\end{array}$$

Within WAILI, the exponent $e$ is called a "shift" because it indicates how many places the mantissa $m$ has to be shifted to the left (in base $\sqrt{2}$!) to obtain its real value $x$. With each subband, a shift is associated, depending on the type of wavelet and the transform depth. In the special two-dimensional case where always both rows and columns are transformed, the shifts can be interpreted as binary shifts after dividing them be 2.

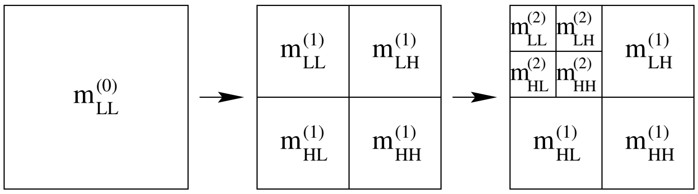



Figure 5.4: Two steps of the two-dimensional wavelet transform.

## 5.5 Summary of the Functionality of WAILI

Summarized, the software library WAILI provides the following functionality:

**Wavelet transforms** Wavelet transforms of images based on the integer version of the lifting scheme. One can transform both the columns and rows of an image, or only the columns or rows.

**Image processing operations** Crop and merge, noise reduction based on generalized cross validation, scaling, soft and hard thresholding, and edge enhancement.

**Basic operations** Addition and subtraction of images.

**Utility operations** Calculation of peak signal to noise ratio (PSNR), first order Shannon entropy, correlation matrices, minima and maxima in an image, histograms, and GCV values.

**Image import/export** Import of images in PNM and TIFF formats, and export of images in PNM format.

WAILI includes an interactive demo program that shows the usage of some of these operations, like e.g. wavelet transforms, thresholding and noise reduction. C++ source code for a simple image compression example using WAILI is present in Appendix B.

Including the demo program and the examples, WAILI counts ca. 12500 lines of C++ code.

## 5.6 Evaluation of the Compression Potential

In this section we evaluate the decorrelation achieved by some different kinds of wavelet transforms, as implemented in WAILI. This decorrelation is a measure for the compression potential.

### 5.6.1 Description of the Compression Method

We used a very simple lossy compression method, namely hard-thresholding of the wavelet coefficients. Thus all wavelet coefficients with a magnitude smaller than the threshold value are replaced by zeros.

### 5.6.2 Measure of the Compression Rate

The achieved compression rate is estimated based on the first order entropy of the thresholded wavelet coefficients.

The first order Shannon entropy $H$ for a string of symbols $i$ from a set $\mathcal{S}$ is defined by:

$$H = -\sum_{i \in \mathcal{S}} p_i \log_2 p_i,$$

with $p_i$ the probability of the appearance of the symbol $i$. The first order entropy $H$ is the theoretical minimum number of bits per symbol needed to encode the string by considering individual symbols [41].

Note that encoding techniques that combine multiple symbols (e.g. zerotree encoding) can increase the encoding efficiency beyond the limit as indicated by the first order entropy.

### 5.6.3 Measure of the Image Quality

The image quality after performing the inverse wavelet transform is measured by the Peak Signal to Noise Ratio (PSNR). The PSNR of a reconstructed signal $x_i^*$, compared to the original signal $x_i$, is defined by

$$\mathrm{PSNR} = 20 \log_{10} \left( \frac{x_{\mathrm{peak}}}{\mathrm{RMSE}} \right) dB,$$

with $x_{\mathrm{peak}} = \max_i |x_i|$ and RMSE the Root Mean Square Error:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=0}^{n} (x_i - x_i^*)^2}{n}},$$

with $n$ the signal length.

Other quality measures are also possible. Visual quality is important in many application fields, but it is subjective and difficult to measure. The 1-norm $\sum_i |x_i - x_i^*|$ is another option. We chose the PSNR because it is simple to calculate and commonly used. Furthermore the PSNR is related to the properties of (bi)orthogonal wavelets: the identity of Parseval (see Equation 2.4) and the similar formula for a Riesz basis (see Equation 2.5) suggest that neglecting the wavelet coefficients with the smallest magnitudes is a good compression approach if one wants to keep a high PSNR.

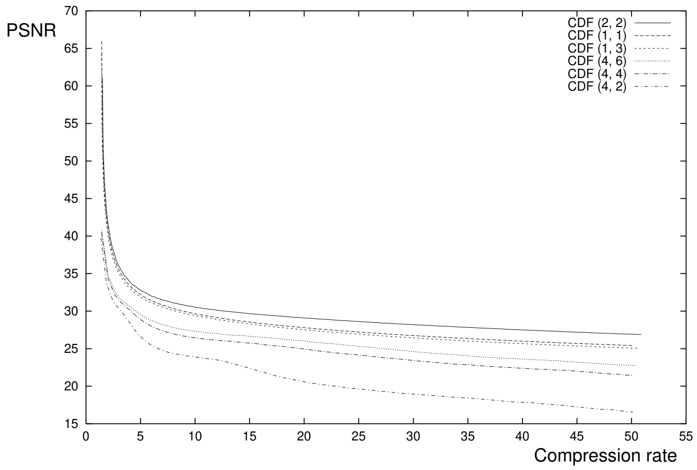



Figure 5.5: Comparison of compression rate vs. quality of reconstruction (PSNR, in dB) for different types of wavelets. The graphs for CDF $(2,4)$ and CDF $(2,6)$, and CDF $(1,5)$ are not shown because they are very similar to the graphs for CDF $(2,2)$ respectively CDF $(1,3)$.

### 5.6.4 Results

As test image, we used an 8-bit grey scale aerial picture of our campus, counting $768 \times 768$ pixels. The image was wavelet transformed using the integer wavelet transform routines of WAILI, up to 6 transform levels.

Figure 5.5 compares the compression rate vs. the quality of reconstruction (PSNR) for different types of wavelet transforms of the Cohen-Daubechies-Feauveau family of biorthogonal wavelets.

However, comparing the reconstructed image after decompression with the input image is not completely fair because the input image contains noise. Due to the methodological similarity between the noise reduction and compression operations, it is quite logical that the visual image quality at low compression rates is better than the visual quality of the noisy input image. To take this into account, Figure 5.6 shows the performance of compressing the original input image and two denoised versions:

**Global denoising** The first image was denoised by soft-thresholding with a global threshold (per subband), calculated using Generalized Cross Validation (see Section 4.2.2).

**Adaptive denoising** The second image was denoised by soft-thresholding with a local threshold, calculated using Generalized Cross Validation on blocks of

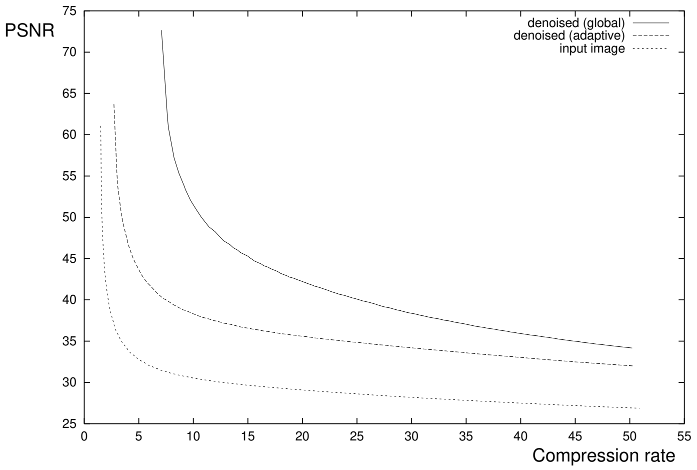



Figure 5.6: Comparison of compression rate vs. quality of reconstruction (PSNR, in dB) for CDF $(2,2)$ wavelets for the original input image and two denoised versions.

$32 \times 32$ coefficients. This method adapts better to the local properties of the image and gives a better visual result, according to experts specialized in image processing (see Section 6.3.1).

In both cases the denoising was performed using the CDF $(2,2)$ wavelet transform.

Based on the experiments described above, we can formulate the following conclusions:

- CDF $(2,x)$ performs better than CDF $(1,x)$, and CDF $(1,x)$ performs better than CDF $(4,x)$.
- The 3 members of the CDF $(2,x)$ family ($(2,2)$, $(2,4)$ and $(2,6)$) perform about equally well. Since CDF $(2,2)$ is the cheapest to implement, it becomes our first choice.
- Within the CDF $(1,x)$ family, $(1,1)$ (Haar) performs best, followed by $(1,3)$ and $(1,5)$.
- Within the CDF $(4,x)$ family, there are very clear performance gaps between $(4,6)$ and $(4,4)$, and between $(4,4)$ and $(4,2)$.
- Compressing denoised images allows for higher compression rates, as expected.

## 5.7 Comparison with Other Wavelet Software

Several software packages and tools implementing wavelet transforms are available. The available wavelet software can be divided in three classes: software libraries to be linked with your own applications, extension packages for common integrated technical computing environments like Matlab and Mathematica, and stand-alone tools for data analysis.

Most or all of them implement classical first-generation wavelet transforms on floating point data. They are usually targeted at signal processing (one dimension: speech, sound, ...) or image processing (two dimensions), providing simple operations like thresholding (for compression or noise reduction), or at numerical analysis applications. Some of them offer the wavelet packet transform as a straightforward extension to the fast wavelet transform. The Discrete Wavelet Transform (DWT) is most popular due to its simplicity and the discrete nature of digital signals, while some software also provides the Continuous Wavelet Transform (CWT) [13].

WAILI differs in that it is a software library providing *integer wavelet transforms* implemented using the *lifting scheme*. LIFTPACK [19] is a software library that also implements the lifting scheme, but it differs substantially from WAILI:

- WAILI uses integer wavelet transforms, while LIFTPACK uses floating point transforms,
- WAILI uses the “Mallat” ordering for the wavelet coefficients, while LIFTPACK implements the in-place transform with an interleaved ordering,
- WAILI uses symmetric extension at the image borders, while LIFTPACK modifies the wavelet bases near the borders.
- WAILI is targeted at image processing and includes some image processing operations, while LIFTPACK is meant for generic wavelet processing, including numerical analysis applications.

# Chapter 6

# Large-Scale Image Processing

In this chapter we present a scheme to process very large images that do not fit in the memory of a single computer. Such images are divided into blocks. Because of the spatial locality properties of wavelets these blocks can be processed quasi independently, allowing efficient parallel programming. The blocking is almost completely transparent to the user. Our implementation is based on the software library WAILI we developed before.

## 6.1 Introduction

In application areas like Geographical Information Systems (GIS), image sizes are measured in gigabytes (GB) and even terabytes (TB). For example, an aerial color image of Belgium, where each pixel corresponds to one square meter, consumes 90 GB of storage. It is obvious that such images need to be stored and handled in a compressed form. Wavelets can provide for this since wavelet-based techniques offer better compression rates than other techniques. Moreover, many image processing operations can be done in the wavelet domain. Some of them (e.g. noise reduction and edge enhancement) can even be done better in the wavelet domain than in the spatial domain.

We describe an extension of the software library WAILI to block-based processing, which allows for a parallel implementation [55].

## 6.2 Very Large Images and Tiling

We define an image to be *very large* if it does not fit in the memory of the computer system that is processing it. Note that by this definition, whether an image is very large or not depends not only on the image but also on the computer system. Examples of very large images are:

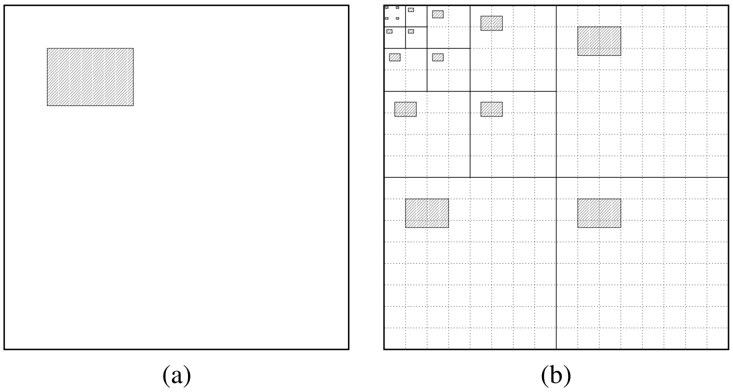



Figure 6.1: Locality in the (a) *space* and (b) *wavelet domain*. The dotted lines in (b) indicate block boundaries.

- An aerial color picture of Belgium, digitized at 1 pixel per square meter (90 GB) on a high-powered workstation,
- An aerial picture of a city on a set top box or kiosk system.

### 6.2.1 Tiling

In order to process very large images, they have to be divided in blocks (tiling). For this division only the spatial domain is important, and since wavelets are localized in the spatial domain, one needs only a limited subset of the input image to calculate a subset of the wavelet coefficients. The same is true for the inverse transform. From Figure 6.1, it is clear that only a fraction of the blocks in the right part of the figure are needed to reconstruct the rectangle in the left part.

Since not all blocks have to be present in main memory at the same time, this allows for the successful processing of very large images on small uni-processor machines with limited memory. Moreover, the processing of the blocks can be distributed among multiple machines and/or CPUs.

Within WAILI, each subband is divided separately in blocks. Blocks are square and block sizes must be powers of two (except near the borders of the image). Because small blocks cause more bookkeeping overhead, it is recommended to use the same block size at different resolution levels.

Since we want to process both complete images and subimages that are part of a large image, the upper left corner of a (sub)image is not always located at the origin $(0,0)$, but may have arbitrary coordinates (see Figure 6.2). To facilitate the wavelet transforms, the internal boundaries of the blocks must always be aligned to the block size, i.e. internal blocks must start at coordinates that are multiples of

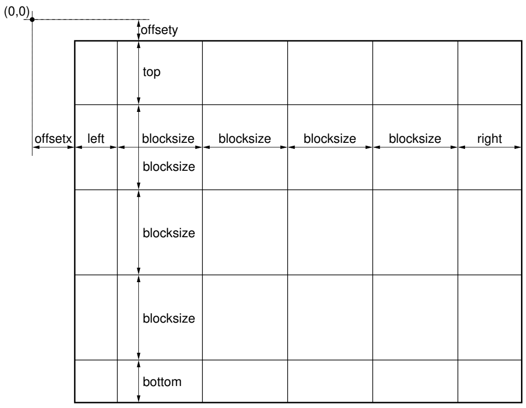



Figure 6.2: Division of a subband in blocks. The margins (*top*, *bottom*, *left* and *right*) are chosen such that the coordinates of the block boundaries are always multiples of the block size.

the block size. Hence, not only the blocks at the right and bottom borders of the image may be rectangular and smaller than the block size, but also at the left and upper borders.

The whole wavelet decomposition is divided in blocks like shown in Figure 6.3. Within each channel (color images consist of multiple channels), the tree forms a "Mallat" ordering (cfr. Figure 5.3). A block within an image is thus uniquely defined by 5 indices: *channel* , resolution *level*, *subband*, *column* and *row*.

To implement wavelet transforms on tiled images, one has to consider the following steps:

**Splitting step (lazy wavelet transform)** To split a block in "even" and "odd" coefficients, in both the horizontal and vertical directions, one needs that block only. So this step is trivial to parallelize.

**Lifting steps** To calculate the lifting operation on a border coefficient, some coefficients from an adjacent block are needed. Thus in a parallel implementation there must be communication of border coefficients (exchange of "overlap regions", with a width depending on the wavelet transform). Since in a primal lifting step the "even" blocks (i.e. the blocks with the even coefficients) are updated by the "odd" blocks, and vice versa in a dual lifting step, these updates can be done independently in each block, except for the communication just mentioned. Hence, this step can be parallelized easily. The parallel efficiency grows with increasing block size due to the "perimeter

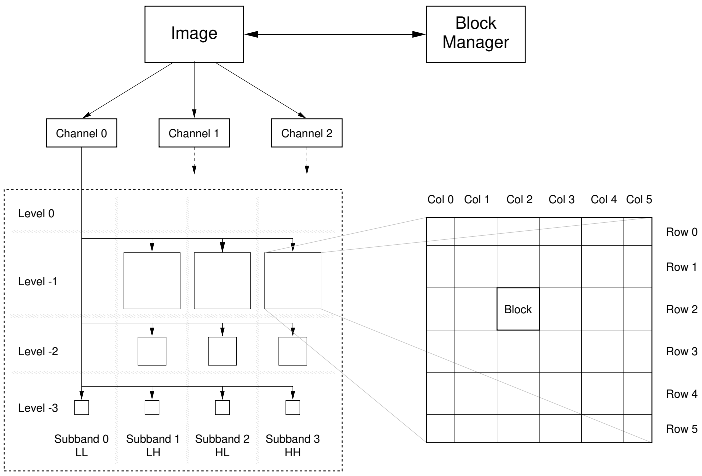



Figure 6.3: Subdivision of an *image* into *channels*, *levels*, *subbands* and *blocks*.

effect".

### 6.2.2 Block Management

We started from the software library WAILI that we developed before and added support for very large images that are divided in blocks. This is nearly completely transparent to the user: all traditional operations can be performed, just like on small non-blocked images (see Appendix B).

The "block manager" is responsible for managing the blocks of which a very large image is composed. To access a part of the (transformed) image, the block manager is asked for blocks and retrieves them, e.g. from disk or via a network. If a block is modified, it has to be updated on the storage device. Blocks are passed to the compression module to remove redundancy among the wavelet coefficients within each block. The block manager for sequential processing also keeps a cache of recently used blocks to speed up processing (for parallel processing we assume all blocks are simply distributed across all processing nodes). An overview of the complete system is given in Figure 6.4.

To obtain a parallel version, we only need to adapt the block manager and introduce communication of the border coefficients during the lifting steps.

### 6.2.3 Implementation in WAILI

Since the basic block of a wavelet decomposition hierarchy is a subband (subband), tiled images are implemented by deriving two subclasses of the NTChannel class

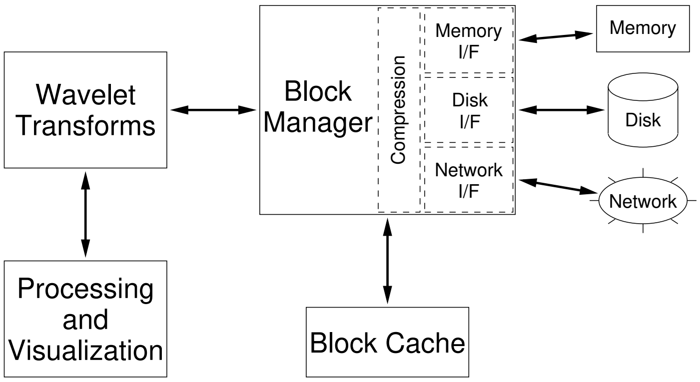



Figure 6.4: Design of the large-scale image processing system.

(see Section 5.2.4):

***NTChannelR*** A regular subband, consisting of one block only.

***NTChannelB*** A blocked subband, consisting of several blocks.

With each image, a *BlockMgr* object is associated, which takes care of managing the individual blocks. The blocks of an image can be stored in main memory, in files on disk, . . . , depending on the type of the BlockMgr.

By adding support for tiled images to WAILI, the code size increased from ca. 12500 to ca. 21000 lines of C++ code.

# 6.3 Image Processing on Tiled Images

Many image processing operations can be performed on wavelet-transformed images. If large images are divided in blocks, it is feasible that we can still perform those image processing operations on the tiled images.

## 6.3.1 Block-Based Denoising

WAILI uses soft-thresholding to reduce additive, stationary noise. The threshold is selected at each transform level and for each subband separately, using generalized cross validation [57]. Since this is an asymptotical method, it only performs well for sufficiently large subbands.

For wavelet-transformed images that are divided into blocks, the denoising technique is applied independently on each block of each subband. Because only the high-pass subbands are processed and blocks belonging to low-pass subbands are left alone, it does not become visible where the borders between the blocks are located.

Visual inspection by experts in the field showed that one gets better results, compared to the case where complete subbands are denoised. This is explained by the threshold now being more adapted to the local properties of the image. Thus denoising is not only easily parallelizable, it also yields a better image quality.

### 6.3.2 Image Compression

Due to their decorrelating properties, wavelets are well suited for the compression of images and signals. If a sufficiently smooth image is transformed, most high-pass coefficients will be zero or very small.

Compression algorithms based on wavelets often use a variant of "zerotree encoding" (see Section 4.5.4). However, for very large images, each zerotree contains coefficients from a multitude of blocks. Thus zerotree encoding will cause a block access pattern that is not feasible and the application of zerotree encoding is limited. A possible solution is to use a two-level encoding strategy:

1. For the lowest resolution levels, zerotree encoding can still be used because the subbands at these levels consist of only one block.

2. For higher resolution levels, a block-based encoding can be used. At these levels the subbands contain several blocks. If the block size is sufficiently large, each block will contain enough coefficients to be well compressible. Note that these subbands are high-pass subbands, with many small or zero coefficients and (typically) only a few large coefficients. Since all blocks can be compressed independently, this step is trivially parallelizable.

### 6.3.3 Crop and Merge

WAILI already provided crop and merge operations on wavelet-transformed images. After we had added support for very large images, these crop and merge operations could be performed on tiled images, too. Since the tiling of subbands is completely transparent and because only a limited subset of all blocks are required, crop and merge are performed in an efficient way.

### 6.3.4 A GIS Application

In GIS applications, a very large aerial image is usually created by combining multiple small aerial pictures. A wavelet-transformed tiled version of the very large image can be created using the following recipe:

1. Create an empty very large image, containing no data (i.e. a grey image).

2. For all aerial pictures $i$
   (a) Load aerial picture $i$ into memory.
   (b) Transform aerial picture $i$, taking care of the absolute coordinates and resolution of the picture.
   (c) Paste the wavelet-transformed aerial picture $i$ in the very large image.

After these steps, one has a wavelet-transformed image that is exactly the same as if one had first glued together all aerial images, and transformed the result. However, our method requires much less memory.

Thanks to the multi-resolution properties of the wavelet transform, one can now select an arbitrary rectangular subimage of the large image, at any dyadic resolution level, and cut it out of the large image, for display or print. If an intermediate resolution is required, the subimage must be scaled. One can even use the above techniques to "walk" through the image (panning), and zoom in or out depending on the degree of interest.

## 6.4 Transform Results

To get an impression of the sequential performance of our wavelet transform code, we transformed an image of $3584 \times 3584$ pixels, using a block size of $128 \times 128$ pixels, for a total of $28 \times 28$ blocks. We used the CDF $(2,2)$ biorthogonal wavelets. All calculations were done on 16 bit integer data on a single processor machine (Intel Pentium II, 350 MHz) with sufficient memory to avoid paging. Note that after 2 transform levels, the subband size is no longer divisible by the block size and one has to take care of rectangular blocks.

Each level of the CDF $(2,2)$ wavelet transform of $n$ blocks consists of 5 steps:

- Lazy wavelet transform in both the horizontal and vertical directions. In order to maintain the block size, 4 blocks are jointly transformed into 4 new blocks. In total $n$ blocks are transformed.
- Lifting steps on the columns of the image: a dual lifting step ($n/2$ blocks) and a primal lifting step ($n/2$ blocks).
- Lifting steps on the rows of the image: a dual lifting step ($n/2$ blocks) and a primal lifting step ($n/2$ blocks).

Timings are shown in Table 6.1. An additional overhead of about 0.6 seconds was caused by the supply of blocks by the block manager. However, in a parallel implementation we can assume that the blocks are distributed over all processors, so this overhead can be spread over the processors too.

In a parallel implementation, the lazy wavelet transform requires no communication if the 4 neighboring blocks are transformed by the same processor. If we

| Transform Level | Lazy Transform # Calls | Time | Lifting Steps # Calls | Time | Total Time | Cumulative Total Time |
|---|---|---|---|---|---|---|
| 1 | 784 | 0.26 | 1568 | 5.59 | 5.85 | 5.85 |
| 2 | 196 | 0.02 | 392 | 1.34 | 1.22 | 7.07 |
| 3 | 49 | 0.01 | 128 | 0.33 | 0.34 | 7.41 |

Table 6.1: Timings for the wavelet transform of an image of size $3584 \times 3584$. Each call corresponds to the processing of one elementary block. All timings are in seconds.

would use $14 \times 14 = 196$ CPUs, the lazy wavelet transform for the first level could be calculated in 1.33 ms and the calculation time for the 4 lifting steps would be 28.5 ms. The lifting steps require communication of the border coefficients. For the CDF $(2,2)$ transform, in each lifting step 2 strips of 128 coefficients must be exchanged between processors. On current parallel systems (e.g. IBM SP2), the communication cost for such short messages is dominated by the startup time ($\mathcal{O}(50\,\mu\text{s})$), which is negligible compared to the calculation cost.

However, the amount of data to be transformed is divided by 4 at each level. Hence, at the second level, we can keep the block size equal to 128, but then only one fourth of the CPUs can be kept busy. Thus the execution time for the second level will also be approx. 30 ms. A similar reasoning holds for the other levels, but in our example rectangular blocks will appear at level 3, causing an additional overhead.

We can conclude that a parallel implementation can be efficient if the number of transform steps is small, and the number of processors is not too large, relative to the image size.

## 6.5 Conclusions

We have described a strategy to perform wavelet-based image compression and processing on very large images. The images are split in blocks, which can be handled nearly independently.

The (integer) wavelet transform is implemented using the lifting scheme, which can be parallelized easily and efficiently, since only communication of coefficients at the borders of the blocks is required.

The method is implemented as an extension to the software library WAILI. Blocks are accessed and handled under supervision of a block manager. Although the software has not been parallelized yet, the limited communication and the existing block manager allow an easy and efficient parallelization.

The division in blocks also has a positive influence on the performance of our denoising algorithm. Because denoising works independently on the blocks, it adapts better to the local properties of the image.

# Chapter 7

# The Red-Black Wavelet Transform

In this chapter we present a new kind of second-generation wavelets on a rectangular grid. These wavelets are constructed using two-dimensional lifting which is based on a red-black blocking scheme. Compared to classical tensor product wavelets on the same grid, these new wavelets show less anisotropy. The performance of the Red-Black wavelets is compared to tensor product wavelets in an image denoising application.

## 7.1 Introduction

Classical one-dimensional wavelet transforms can be extended to more dimensions using tensor products, yielding a separable multi-dimensional transform. A disadvantage of this technique is the introduction of an anisotropy in the wavelet decomposition. In the two-dimensional case, a tensor product wavelet transform will favor horizontal, vertical and diagonal features of the original data. Other features are not that easily detected.

Non-separable wavelets can provide a solution to this. They allow for the construction of two-dimensional wavelets on e.g. lattices (e.g. on the so called "quincunx" lattice [30]) or on hexagonal grids [44]. All these are heavily based on Fourier transforms, just like the classical wavelets.

Second-generation wavelets designed using the lifting scheme are another option. The lifting scheme is a generic method to create wavelets on intervals, irregular samplings, meshes, manifolds, . . . , without relying on the Fourier transform. As a bonus the classical one-dimensional wavelets can be constructed using lifting, too.

In this chapter, we present a new kind of second-generation wavelets on a rectangular grid — more specifically, on a quincunx lattice — constructed using the lifting scheme. The performance of the new wavelets is compared to tensor product wavelets in an image denoising application [52].

## 7.2 The Red-Black Wavelet Transform

In Section 3.2, we presented the principle of constructing wavelet transforms by the lifting scheme in a very general setting. It was not assumed that the data sets were one-dimensional for example. The same principle and the same description can be applied for two or more dimensional data as well. As an illustration, we apply these ideas to the two-dimensional case in its simplest possible appearance: a two-dimensional analog of the one-dimensional CDF $(2,2)$ wavelet. We assume that the data are given on a regular rectangular grid. It should be clear form our previous exposition, that this is about the simplest non-trivial case that can be conceived, and that much more complex schemes are possible. However, because in the one-dimensional case, the CDF $(2,2)$ wavelets are widely used, we are convinced that the two-dimensional analog that we present below has the same potentials as the CDF $(2,2)$ wavelet.

### 7.2.1 A Two-Step Method

Our approach is inspired by the well-known Red-Black Gauss-Seidel technique for the iterative solution of linear systems. In the literature [30], the kind of lattice we use is known as a quincunx lattice. However, we prefer the name Red-Black wavelet transform because it is simpler and it is more appropriate to describe the splitting step in the lifting scheme.

The idea, just like in the CDF $(2,2)$ wavelet, is that we first split the data in two subsets. The even/odd splitting from the one-dimensional case is replaced by a checkerboard splitting, with red and black squares. Thus we have a red subset $\lambda_j$ and a black subset $\gamma_j$. Then we predict the values of the black subset $\gamma_j$ from its immediate neighbors in the red set $\lambda_j$. This does not leave much choice and so, a value is predicted from its horizontal and vertical neighbors, i.e., the elements at the N(orth), S(outh), E(ast), and W(est) positions. In accordance with the CDF $(2,2)$ lifting scheme, we take for this prediction an average of these four neighbors. The elements in $\gamma_j$ are replaced by their prediction errors. Next, we have to make an updating step for the data in the red subset $\lambda_j$, which is based on the (new) black values, to preserve the mean value of the data.

For the subsequent resolution level $j-1$, we are left with the (new) red set $\lambda_j$ from the previous level. However, it is clear by looking at Figure 7.1, that these data are arranged along diagonals on the grid. Thus we cannot repeat the same operation on a horizontal/vertical basis as on the previous resolution level, but we

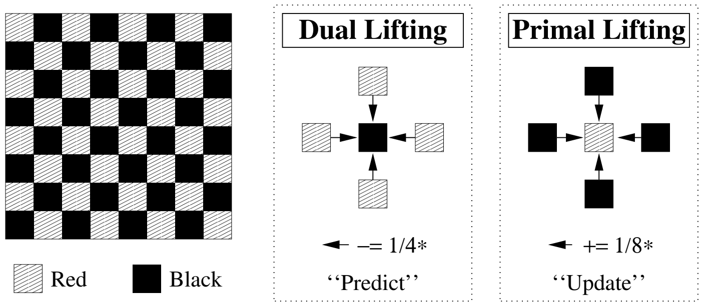



Figure 7.1: Horizontal/vertical lifting in the odd steps: *Red-Black*.

can do it on the diagonals instead. Again we start by splitting $\lambda_j$ into two subsets: the blue subset $\lambda_{j-1}$ and the yellow subset $\gamma_{j-1}$ (see Figure 7.2). The same kind of averaging and updating steps are then computed for the blue-yellow squares. For the next resolution level, we have to decompose the blue subset further, but since this subset is again arranged in a horizontal/vertical manner, we can do again a red-black update. It is seen that we will have alternating red-black and blue-yellow splittings to pass through the subsequent resolution levels. Therefore we say that the Red-Black wavelet transform is a "two level" transform.

We give now the formulas which are based on these ideas in a more schematic presentation. Since we are working on a rectangular grid, the input data set $\lambda_{j+1}$ is an image, i.e. a rectangular matrix $X = x_{i,j}$.

### Horizontal/Vertical Lifting

**Splitting** The image $\lambda_{j+1}$ is split in the *red* squares $\lambda_j$, and the *black* squares $\gamma_j$ (cfr. Figure 7.1).

**Prediction** The black squares are predicted using linear interpolation based on the 4 neighboring red squares:

$$x_{i,j} \leftarrow x_{i,j} - \frac{1}{4}\left(x_{i-1,j} + x_{i,j-1} + x_{i,j+1} + x_{i+1,j}\right), \quad \text{for } i \bmod 2 \neq j \bmod 2.$$

**Update** The red squares are updated using the black squares to preserve the mean value:

$$x_{i,j} \leftarrow x_{i,j} + \frac{1}{8}\left(x_{i-1,j} + x_{i,j-1} + x_{i,j+1} + x_{i+1,j}\right), \quad \text{for } i \bmod 2 = j \bmod 2.$$

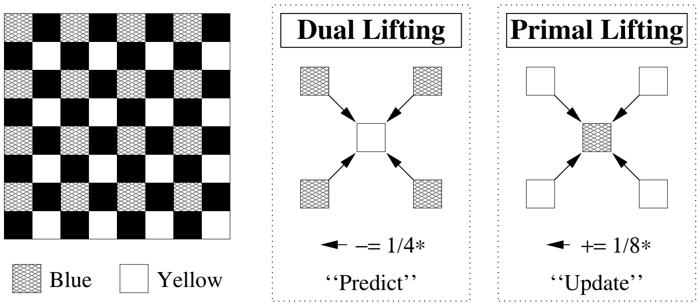



Figure 7.2: Diagonal lifting in the even steps: *Blue-Yellow*.

As a result, because the averaging operation smoothes the data, the values corresponding to the *red* squares are a low-resolution representation of the original image, while the *black* squares contain the detail information. The inverse transform is straightforward.

**Diagonal Lifting**

For this step we consider the checkerboard to be rotated over 45°.

**Splitting** The *red* squares in Figure 7.1 (the data $\lambda_j$) are partitioned in the *blue* (the data $\lambda_{j-1}$) and *yellow* (the data $\gamma_{j-1}$) squares in Figure 7.2.

**Prediction** The yellow squares are predicted using linear interpolation based on the 4 neighboring blue squares:

$$x_{i,j} \leftarrow x_{i,j} - \frac{1}{4}\left(x_{i-1,j-1} + x_{i-1,j+1} + x_{i+1,j-1} + x_{i+1,j+1}\right), \quad \text{for } i \bmod 2 = 1 \text{ and } j \bmod 2 = 1.$$

**Update** The blue squares are updated using the yellow squares to preserve the mean value:

$$x_{i,j} \leftarrow x_{i,j} + \frac{1}{8}\left(x_{i-1,j-1} + x_{i-1,j+1} + x_{i+1,j-1} + x_{i+1,j+1}\right), \quad \text{for } i \bmod 2 = 0 \text{ and } j \bmod 2 = 0.$$

Thus the *blue* squares ($\lambda_{j-1}$) are a low-resolution representation of the original image, while the *yellow* squares contain the detail information. Again, the inverse transform is straightforward.

The next step (horizontal/vertical lifting, again) will be performed on the blue squares only, yielding a multi-resolution decomposition.

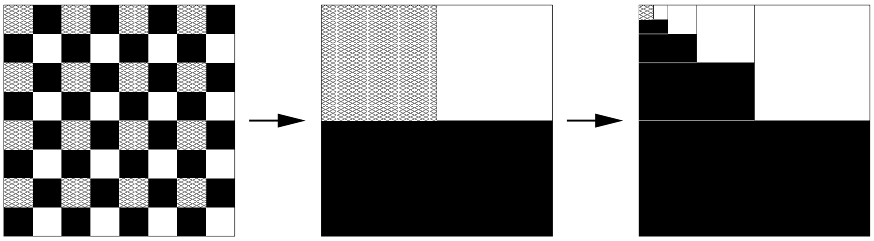


Figure 7.3: Reordering of the interleaved coefficients and multilevel decomposition.

### 7.2.2 Borders

As in the one-dimensional case, one could arrange for the wavelets to live exactly on the grid where the original data are given. However, it is common practice to extend the data symmetrically for the computations near the border [4]. This gives good results in practice for the classical one-dimensional or two-dimensional tensor product wavelets. To avoid further complications, we assume that also in our case the data are extended symmetrically across the borders.

### 7.2.3 Reordering

The above transform is 100% in-place and all coefficients will be interleaved. This is one of the powerful features of the lifting scheme: it does not require additional memory to calculate the forward or inverse transform.

If one wants an ordering more similar to the Mallat ordering, one has to reorder the wavelet coefficients after each diagonal lifting step. For multiple levels, we get the representation of Figure 7.3.

Note that this reordering scheme is not possible if either the number of rows or columns of the image is odd.

### 7.2.4 Integer Wavelet Transform

Because the Red-Black wavelet transform is based on the lifting scheme, a version that maps integers to integers is immediately deduced: all computations giving a floating point number are rounded to an integer, e.g.

$$\mu \leftarrow \mu - \frac{1}{4}(\rho + \sigma + \tau + \upsilon)$$

becomes (curly braces denote rounding):

$$\mu \leftarrow \mu - \left\{ \frac{1}{4}(\rho + \sigma + \tau + \upsilon) \right\}$$

and similarly for all the other steps.

### 7.2.5 Basis Functions

In a classical one-dimensional wavelet transform, all basis functions are derived from two functions — the mother and the father wavelet — by dilation and translation. In the case of one-dimensional biorthogonal wavelets, there is also a dual basis. However, in our construction we use different wavelet transforms for the even and odd steps. As a consequence all basis functions are derived from three[1] functions:

$\psi_{\mathrm{low}}(x,y)$ Basis function corresponding to the low-pass part of the diagonal lifting step.

$\psi_{\mathrm{mid}}(x,y)$ Basis function corresponding to the high-pass part of the diagonal lifting step.

$\psi_{\mathrm{high}}(x,y)$ Basis function corresponding to the high-pass part of the horizontal/ vertical lifting step.

These three functions can be obtained by choosing a blue, a yellow or a red square to take a non-zero value and setting all other coefficients equal to zero. Then the inverse transform is applied to these data, in principle for an infinite number of levels, which results in the basis functions.

Graphs of these basis functions (for the integer variant of the lifting scheme) are shown in Figure 7.4.

### 7.2.6 Other Members of the Family

As we already mentioned at the beginning of this section, we used in the description above very simple prediction and update "molecules". The molecules have the form of a cross: a normal cross for the odd steps and a diagonal cross for the even steps. These give us wavelet functions with 2 vanishing moments, cfr. the Cohen-Daubechies-Feauveau biorthogonal wavelets. By using larger molecules and more advanced prediction and update formulas, one can create smoother basis functions with larger supports.

In parallel with our work, Kovačević and Sweldens completed a similar construction, which considers our example as a special case in a more general multi-dimensional lifting framework [31].

[1] We do not consider the intermediate basis functions corresponding to the low-pass parts after the horizontal/vertical lifting steps.

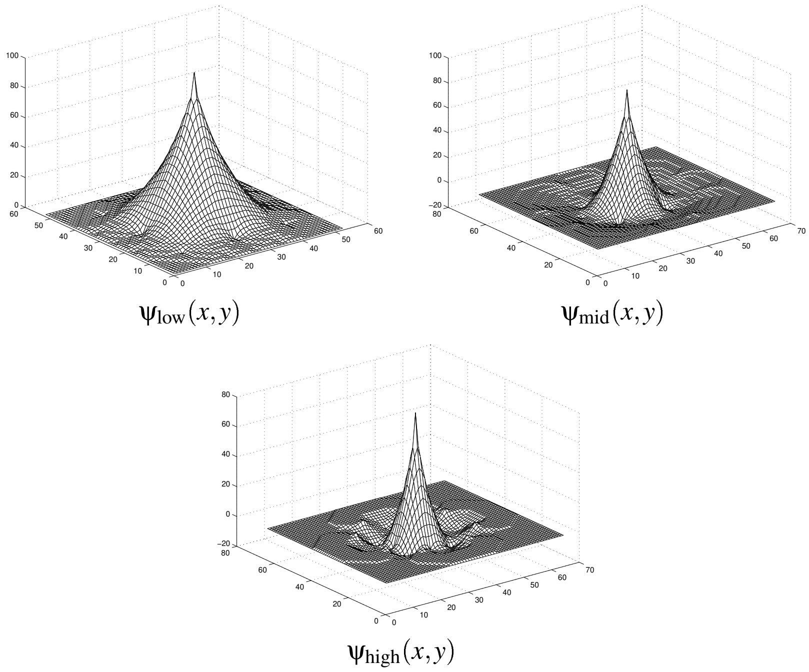



Figure 7.4: Graphs of the basis functions for the integer Red-Black wavelet transform.

## 7.3 Properties of the Red-Black Wavelet Transform

The Red-Black wavelet transform has the following properties:

**2D:** The Red-Black wavelet transform is a true two-dimensional transform. It is not constructed using a tensor product of a one-dimensional transform and thus it is not separable.

**Isotropy:** It is less anisotropic than tensor product wavelets on the same rectangular grid.

The edges which are manifest in the figures of the tensor product wavelets show the pronounced vertical/horizontal orientation, while this is not visible in the figures of the Red-Black wavelets.

This property was our main motivation for developing the Red-Black wavelet transform. Hexagonal data cells would be more anisotropic, but not many input data formats have hexagonal data cells. Most have square or rectangular data cells. And it is not trivial to convert a rectangular grid to a hexagonal grid in a satisfactory way.

**Symmetry:** All basis functions have four symmetry axes: horizontal, vertical, and diagonal.

This follows immediately by construction of these wavelets and is visually checked in the pictures of Figure 7.1.

**Biorthogonality:** The splitting in red and black or blue and yellow pixels creates trivial biorthogonal basis functions. Since lifting steps preserve the biorthogonality properties the transform as a whole has biorthogonal basis functions, too [49].

**Smoothness:** The wavelet functions (both primal and dual) have two vanishing moments: constant, linear and bilinear functions can be represented exactly by the scaling functions. This is similar to the Cohen-Daubechies-Feauveau $(2,2)$ biorthogonal wavelets, from which we borrowed the construction scheme and extended it to two dimensions.

Smoother wavelets with larger support can be constructed by enlarging the prediction and update molecules, and by using multiple lifting steps.

**Implementation:** Both the forward and inverse transform are very simple to implement and have fast execution times ($O(n)$, with $n$ the number of elements in the grid), due to the underlying lifting technology.

## 7.4 The Redundant Wavelet Transform

In the classical first-generation framework, there is a decimation step after each filter step to reduce the amount of data in every subband. In the second-generation case, this is reflected in the splitting step: the original data set is split in two distinct data sets. Thus at each transform step, the amount of work is reduced, compared to the previous step. This makes the total amount of work proportional to the size of the input data ($O(n)$), hence the name *Fast Wavelet transform*.

But for some applications, this is a disadvantage: it makes the transform *translation variant*: if the input data is translated, at each resolution level data samples may migrate to a different data set due to the partitioning in distinct data sets at the subsequent resolution levels.

The solution to this problem is called the redundant wavelet transform (see Section 2.5). Using the lifting scheme, it is still possible to implement the redundant wavelet transform. The following changes are required:

**Copying** Instead of partitioning the data set $\lambda_{j+1}$ into two distinct data sets $\lambda_j$ and $\gamma_j$, we copy $\lambda_{j+1}$ to $\lambda_j$ and $\gamma_j$. Thus both $\lambda_j$ and $\gamma_j$ contain the whole input data set.

**Prediction** We use a different prediction at every transform level to mimic the upsampling of the filters:

$$\gamma_j \leftarrow \gamma_j - \mathcal{P}_j(\lambda_j).$$

**Update** We use a different update at every transform level to mimic the upsampling of the filters:

$$\lambda_j \leftarrow \lambda_j + \mathcal{U}_j(\gamma_j).$$

The operators $\mathcal{P}_j$ and $\mathcal{U}_j$ were given an index, because they compute values that are still based on four samples, but they need not be the "nearest" samples anymore because the position of the elements must be computed by a formula that depends on the resolution level.

## 7.5 Comparison with the CDF $(2,2)$ Tensor Product Wavelets

We compared the performance of the integer Red-Black wavelet transform with the integer Cohen-Daubechies-Feauveau $(2,2)$ tensor product wavelet transform. The basis functions for the latter transform are shown in Figure 7.5.

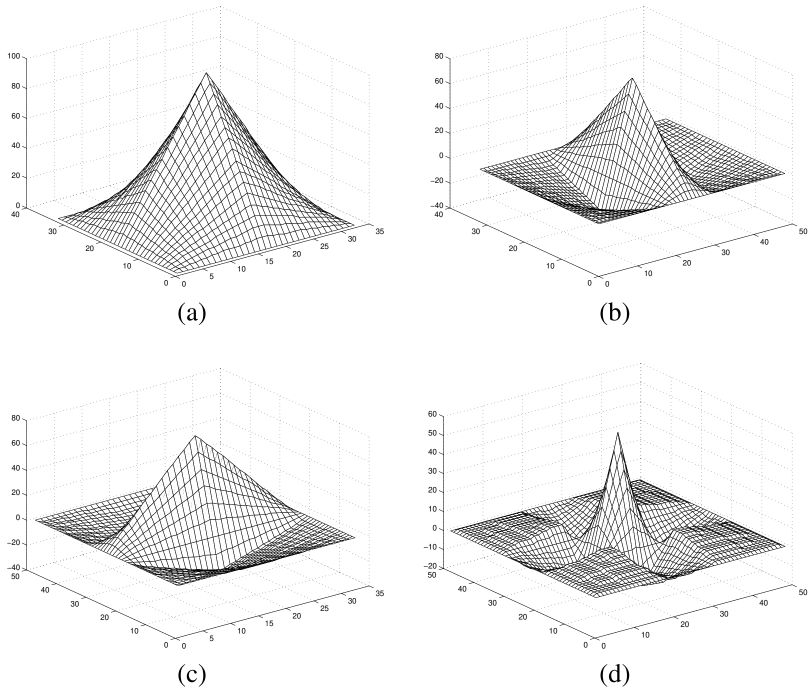


Figure 7.5: Graphs of the basis functions for the integer Cohen-Daubechies-Feauveau $(2,2)$ tensor product wavelets.

### 7.5.1 Test case 1

We considered an image processing operation on an grey scale aerial image: noise reduction using wavelet shrinkage based on Generalized Cross Validation [24].

For Red-Black and CDF $(2,2)$, we tested both the fast wavelet transform and the redundant wavelet transform. The results, obtained by visual inspection, are summarized below:

- The tensor product CDF $(2,2)$ wavelets perform marginally better than the Red-Black wavelets and result in more pronounced edges. This is caused by the larger support for the Red-Black wavelets.

  There is one exception: faint lines that are not horizontal, vertical or diagonal are better preserved by the Red-Black transform, due to the less anisotropic properties.

- As expected, the redundant wavelet transform always performs better than the corresponding fast wavelet transform. Because the redundant wavelet transform does not use subsampling and is translation invariant, it is better in preserving fine edges.

### 7.5.2 Test case 2

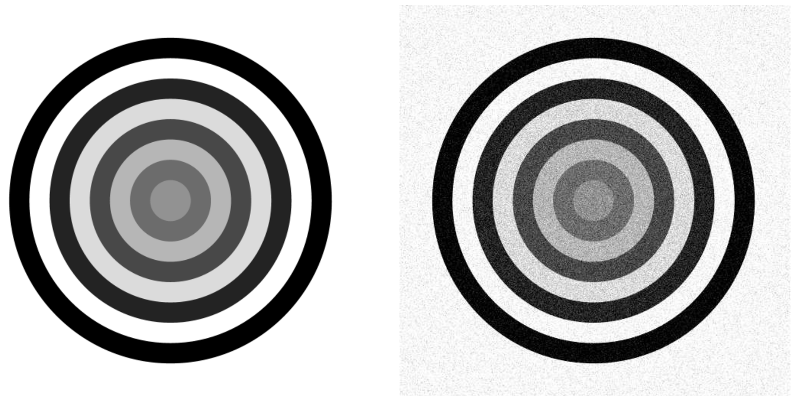
For the second case, we started from an image containing anti-aliased concentric circles, in different shades of grey (Figure 7.6a), and added white noise to it (Figure 7.6b).

The three-level integer wavelet transform of the noise-free image is shown in Figure 7.7, for both the Red-Black and the tensor product CDF $(2,2)$ wavelets. The concentric circles are more pronounced in the decomposition when using the Red-Black wavelets, due to their better anisotropic properties. The tensor product CDF $(2,2)$ wavelets clearly show to detect only horizontal, vertical and diagonal features.

Then we reduced the noise in the noisy image using the same wavelet shrinkage technique as in the first test case, again for both the Red-Black and the tensor product CDF $(2,2)$ wavelets. The resulting images are shown in Figure 7.8. The Peak Signal to Noise Ratio (PSNR) measure indicates that the tensor product CDF $(2,2)$ wavelets perform a bit better (approx. 0.4 dB better). However, according to the human eye, the Red-Black wavelets give sharper concentric circles in low-contrast areas, i.e. in the center of the image.

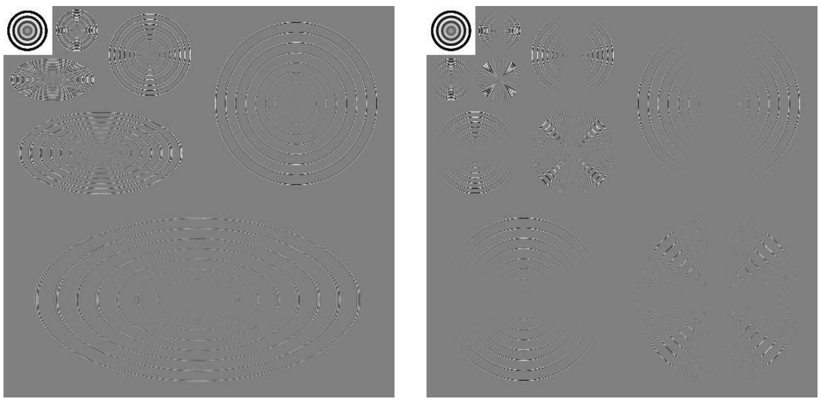
## 7.6 Conclusions

Red-Black wavelets are less anisotropic than tensor product wavelets because they favor the horizontal and vertical directions less than tensor product wavelets. Com-



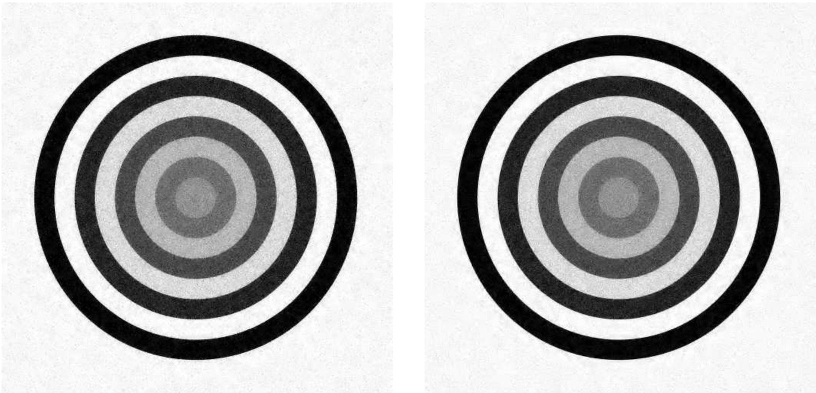
(a) (b)

Figure 7.6: Grey scale test images ($512 \times 512$ pixels): (a) without and (b) with additive white noise ($\sigma = 20$).



(a) (b)

Figure 7.7: Three-level wavelet decomposition of the test image without noise, using (a) Red-Black and (b) tensor product CDF $(2,2)$ wavelets.



(a) (b)

Figure 7.8: Reconstruction of the original image after denoising using (a) Red-Black (PSNR = 26.8 dB) and (b) CDF $(2,2)$ wavelets (PSNR = 27.2 dB). Although the PSNR is lower when using Red-Black wavelets, the Red-Black wavelets give sharper concentric circles in low-contrast areas, i.e. in the center of the image, according to the human eye.

pared to a similar tensor product wavelet, image denoising using Red-Black wavelets performs better for lines that are not horizontal, vertical or diagonal.

Besides image processing, other possible applications are situated in partial differential equations.

The "two level" principle of the Red-Black wavelet transform is not restricted to a quincunx lattice. It can also be extended to triangular or hexagonal regular grids: an update and predict pair on a triangular grid will yield a hexagonal low-pass grid, while an update and predict pair on a hexagonal grid will yield a triangular low-pass grid. Note that one is not limited to planar grids: (semi-)regular grids of triangles or hexagons on a sphere are also possible.

# Chapter 8

# Experiments with a Wavelet-Based APOD

In this chapter we present another aspect of wavelet-based compression. We will use wavelet packet compression to reduce the complexity of the Proper Orthogonal Decomposition (POD) and obtain an Approximate POD (APOD). The performance of this APOD is evaluated by using it to represent and analyze the behavior of a dynamical system. We will show that only very few eigenmodes are needed to allow a good representation of the dynamical system, thanks to the decorrelating properties of a wavelet packet transform.

## 8.1 Introduction

In many domains of applied sciences it is important to analyze the behavior of large-scale dynamical systems, described by partial differential equations. These dynamical systems are processes in the real world, like e.g. chemical reactions. Because of the high-dimensionality of such a system, it is necessary to first reduce the complexity of the system, while retaining its intrinsic properties, to allow for an efficient analysis.

The Proper Orthogonal Decomposition (POD) or Karhunen-Loève Transform (KLT) is a powerful tool to obtain low-dimensional models for large-scale dynamical systems. Starting from a set of solutions (obtained by experiment or computation), called snapshots, the method computes an "optimal" basis of eigenmodes for the snapshots, which can be used to construct a low-dimensional model of the dynamical system. Unfortunately the construction of a POD and also the projection of the original problem onto the eigenmodes are very expensive operations.

We have done experiments with an Approximate Proper Orthogonal Decomposition, based on data compression using symmetric biorthogonal wavelets. A wavelet packet transform allows to decorrelate and compress the snapshots, after which we compute the POD of the compressed data set of much lower dimension, while we still obtain good approximations for the eigenmodes of the original system.

We will present results for the computation of an Approximate POD for a system of partial differential equations (the one-dimensional Brusselator reaction-diffusion model), using symmetric wavelets (Cohen-Daubechies-Feauveau) [53].

## 8.2 The Proper Orthogonal Decomposition

The Proper Orthogonal Decomposition (POD), also known as Karhunen-Loève Transform (KLT), is a tool to analyze a given set of data by decomposing the data into an "ideal" orthonormal basis.

With "ideal" basis we mean an orthonormal basis that gives the best (least squares) approximation using as few basis vectors as possible. The basis vectors are ordered such that the best approximation using only $i$ vectors is obtained by using the first $i$ vectors. Thus they are ordered by decreasing "importance".

Suppose we have $S$ linear independent snapshots $X_i$ of size $N$ (i.e. linear independent vectors with $N$ elements) with $S \ll N$. We assume that the average of the snapshots

$$\overline{X} = \frac{1}{S}\sum_{i=0}^{S-1} X_i$$

is zero. If this would not be the case, we can always subtract the average $\overline{X}$ from all snapshots:

$$X_i \leftarrow X_i - \overline{X}, \qquad i = 0, \dots, S-1.$$

One can show that the eigenvectors $E_k$ of the $N \times N$ covariance matrix $C$

$$C = \frac{1}{S}\sum_{i=0}^{S-1} X_i X_i^T,$$

satisfying

$$CE_k = \lambda_k E_k, \qquad k = 0, \dots, S-1$$

with

$$\lambda_0 \geq \lambda_1 \geq \cdots \geq \lambda_{S-1} > 0$$

form the best basis we are looking for (the $N - S$ remaining eigenvalues of $C$ are zero.). Since $C$ is symmetric, the eigenvalues $\lambda_k$ are real and the eigenvectors $E_k$ are orthogonal. For every eigenvector $E_k$, the corresponding eigenvalue $\lambda_k$ indicates the relative importance of that eigenvector in the best basis. The eigenvectors are also called the "eigenmodes" of the original snapshots.

Although this approach results in the "best basis" and is ideal in terms of decorrelation, it is not ideal in terms of computing time. Constructing the covariance matrix $C$ requires $O(SN^2)$ operations.

Fortunately, one can use an alternative method, called the "method of the snapshots" [45]. Here one uses the property that the eigenvectors $E_k$ are unique linear combinations of the linearly independent snapshots $X_i$ and thus can be written as

$$E_k = \sum_{j=0}^{S-1} a_{k,j} X_j, \qquad k = 0, \ldots, S-1.$$

Now the eigenvalue equation becomes:

$$\begin{aligned} CE_k &= \lambda_k E_k, \\ \frac{1}{S} \sum_{i=0}^{S-1} X_i X_i^T \sum_{j=0}^{S-1} a_{k,j} X_j &= \lambda_k \sum_{j=0}^{S-1} a_{k,j} X_j, \\ \frac{1}{S} \sum_{i=0}^{S-1} X_i \left( \sum_{j=0}^{S-1} X_i^T X_j a_{k,j} \right) &= \sum_{i=0}^{S-1} \lambda_k a_{k,i} X_i. \end{aligned}$$

Because the $X_i$ form a basis, we can drop the summations on both sides of the last equation and we get

$$\sum_{j=0}^{S-1} X_i^T X_j a_{k,j} = \lambda_k^* a_{k,i}, \qquad \left\{ \begin{array}{rcl} i &=& 0, \ldots, S-1, \\ k &=& 0, \ldots, S-1, \\ \lambda_k^* &=& S\lambda_k. \end{array} \right.$$

Define the matrix $C^*$ as

$$C^* = \left[c^*_{i,j}\right] = \left[X_i^T X_j\right], \qquad i, j = 0, \ldots, S-1,$$

and let the vector $A_k = [a_{k,i}]\,, i = 0, \ldots, S-1$, denote the coefficients of the eigenvector $E_k$ in the basis of the snapshots, then the original eigenvalue problem is equivalent to the eigenvalue problem

$$C^* A_k = \lambda_k^* A_k, \qquad \text{with } \lambda_k^* = S\lambda_k.$$

The construction of the matrix $C^*$ requires $O(NS^2)$ operations, which is much cheaper than the construction of the covariance matrix $C$, since $C \ll N$.

If $S > N$, the snapshots $X_i$ are not independent and thus they no longer form a basis. As a consequence, the matrix $C^*$ will have rank $r \leq N$. However, the method of the snapshots can still be used, because we have for $k = r, \ldots, S-1$:

- the additional eigenvalues $\lambda_k$ will all be zero because of the rank deficiency of $C^*$,

- the additional eigenvectors $A_k$ will be unit vectors (up to a constant) because the mean of the snapshots is zero, and the resulting additional eigenmodes $E_k$ will be zero. Note that in a strict sense the zero vector cannot be an eigenmode, but here we are considering it in a more general framework.

**Note:**

We can "normalize" the snapshots

$$X_i \leftarrow \frac{X_i}{||X_i||\sqrt{S}} \tag{8.1}$$

such that

$$\sum_{k=0}^{S-1} \lambda_k^* = 1.$$

Indeed, the sum of the eigenvalues of a matrix is equal to the sum of the diagonal elements of that matrix. In that case eigenmode $E_k$ contains $100 \times \lambda_k^*\%$ of the total energy.

## 8.3 The Approximate Proper Orthogonal Decomposition

The main idea behind the Approximate Proper Orthogonal Decomposition (APOD) is to decorrelate the original snapshots using wavelets and perform the normal Proper Orthogonal Decomposition (POD) on the compressed snapshots, in the hope that we will get a reasonable approximation for the original eigenmodes, with less computational effort.

### 8.3.1 Theory of Operation

To calculate the APOD, we perform the following operations:

1. Transform the snapshots $X_i$ using an appropriate transform $W$ (we will use a wavelet packet transform, as will be mentioned in Section 8.3.2) that decorrelates the original data:

$$\widehat{X}_i = W(X_i), \qquad i = 0, \ldots, S-1.$$

2. Compress the vectors $\widehat{X}_i$ by removing from every vector a number of entries:

$$\widehat{X}_i \triangleright \widehat{X}_i' \in \mathbf{R}^n, \qquad i = 0, \ldots, S-1,$$

with $n \ll N$.

Removing $m\%$ entries gives a compression rate of $\frac{100}{100-m} : 1$. The strategy to decide which entries to remove can be adapted to the set of snapshots, i.e. based on statistical properties of the set, or can be fixed, i.e. based on a previous statistical analysis.

3. Calculate the POD of the compressed snapshots $\widehat{X}_i'$. This gives the eigenmodes $\widehat{E}_i' \in \mathbf{R}^n$, while the eigenvalues $\lambda_i$ indicate the energy distribution over the eigenmodes.

4. Decompress the eigenmodes $\widehat{E}_i'$ by putting zeros at the places where a vector entry was removed in step 2:

$$\widehat{E}_i' \triangleleft \widehat{E}_i, \qquad i = 0, \ldots, S-1.$$

5. Perform the inverse transform $W^{-1}$ on the eigenmodes $\widehat{E}_i$ to get the approximate eigenmodes $E_i^*$, which are an approximation to the eigenmodes $E_i$ of the snapshots $X_i$:

$$E_i^* = W^{-1}(\widehat{E}_i) \approx E_i, \qquad i = 0, \ldots, S-1.$$

If the transform $W$ is not orthogonal but biorthogonal, we not only have eigenmodes $E_i^*$, but also dual eigenmodes $\widetilde{E}_i^*$.

A schematic overview of these operations can be found in Figure 8.1.

The APOD can be used instead of the genuine POD, for two possible goals:

- to reduce the computation time needed to construct (an approximation of) the eigenmodes $E_i$,
- to use the APOD to construct a "low-dimensional" model, based on the eigenmodes $\widehat{E}_i'$.

### 8.3.2 Decorrelating Transforms

We have chosen to use a wavelet packet transform as the decorrelating transform (see Section 2.6). Our motivation is that a wavelet packet transform is closer to a POD than an ordinary wavelet transform due to the additional degrees of freedom in the choice of the basis functions.

If we transform a square image $X = [x_{ij}]$ with $n \times n$ elements, the two-dimensional wavelet packet decomposition tree $Y = [y_{ijk}]$ has $n \times n \times \log n$ elements, with a topology of $\log n$ "levels" of $n \times n$ elements. To compress the representation in the best basis, we remove the $m\%$ coefficients that are the smallest in magnitude.

Since we want to find the best basis not for a single snapshot, but for a whole *set* of snapshots, we must choose the *same* wavelet packet basis for every snapshot. This basis must be based on the statistical properties of the set of snapshots.

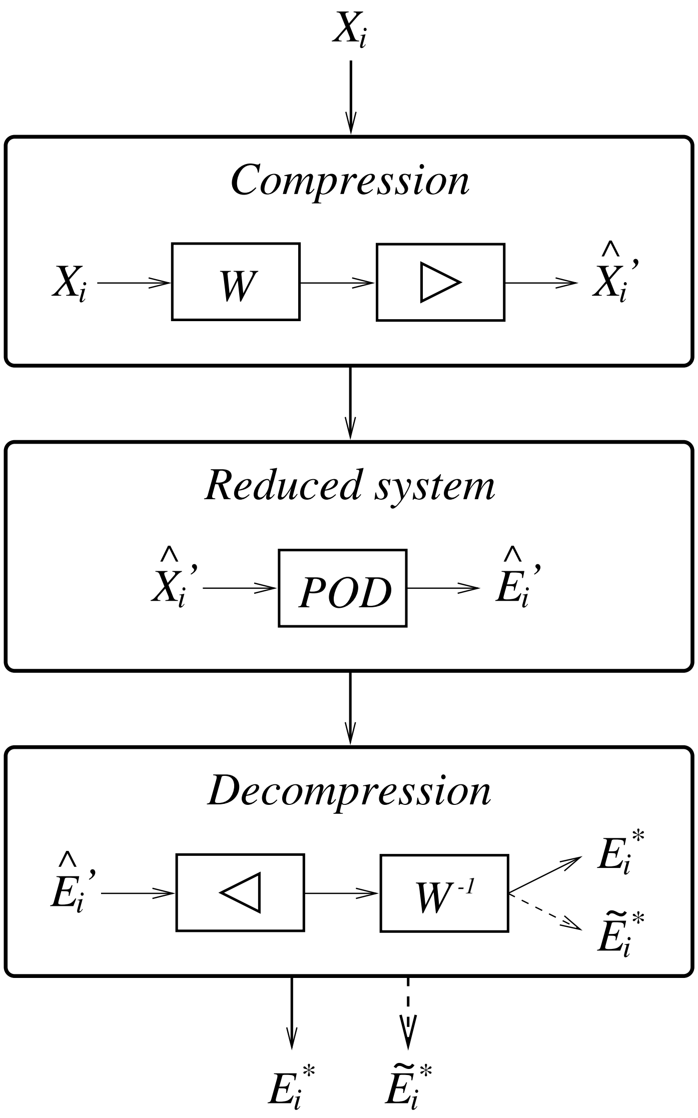



Figure 8.1: Schematic overview of the Approximate Proper Orthogonal Decomposition. The dual eigenmodes $\widetilde{E}_i^*$ are only relevant if $W$ (and $W^{-1}$) are biorthogonal transforms.

To select this basis, we calculate the wavelet packet decomposition tree $Y_l$ of every snapshot $X_l$

$$X_l = [x_{i,l}] \xrightarrow{\text{WPT}} Y_l = [y_{ij,l}],$$

square all entries in the $Y_l$ and sum the corresponding entries of $Y_l$ to obtain a "sum-squared" wavelet packet decomposition tree $Y^*$:

$$Y^* = [y^*_{ij}] \qquad \text{with} \qquad y^*_{ij} = \sum_l y^2_{ij,l}.$$

Then we apply the best basis selection algorithm to the "sum-squared" wavelet packet decomposition tree $Y^*$.

To compress the snapshots in their best basis representation, we remove the coefficients from the $Y_l$ that correspond to the $m\%$ smallest coefficients (in absolute value) in the tree $Y^*$.

Depending on the wavelet filter, the wavelet packet transform can be an orthogonal or a biorthogonal transform.

### Orthogonal Transforms

If the transform $W$ and its inverse $W^{-1}$ are orthogonal transforms, the APOD will yield the same results as the POD if there is no compression. Indeed, because the POD is an *orthogonal* decomposition, applying an orthogonal transform before and its inverse after the POD has no influence on the result.

In the case of compression, we have the following: if we decompose a function in an orthonormal basis $\varphi_i(x)$,

$$f(x) = \sum_k a_k \varphi_k(x),$$

Parseval's identity holds:

$$\sum_k a_k^2 = \|f\|^2.$$

Neglecting the smallest coefficients $a_k$ will result in only a small error after reconstruction. In addition the orthogonality guarantees a good approximation.

Of course the results also depend on the criterion used in step 2.

### Biorthogonal Transforms

If the transform $W$ and its inverse $W^{-1}$ are not orthogonal but only biorthogonal transforms, we cannot guarantee a good approximation of the real POD in all circumstances. In fact, if we do not compress the transformed snapshots, we will not even get the same eigenmodes as with the real POD.

However, if the transform is based on biorthogonal wavelets, the fact that the resulting basis is a Riesz basis will guarantee no longer a perfect but still a good approximation. In a Riesz basis (Equation 2.5), the coefficients of the decomposition show the following property:

$$A\|f\|^2 \leq \sum_k a_k^2 \leq B\|f\|^2,$$

with $A$ and $B$ positive constants. Neglecting the smallest coefficients $a_k$ will still result in only a small controllable error after reconstruction.

Note that in the biorthogonal case we not only have primal eigenmodes $E_i^*$, but also dual eigenmodes $\widetilde{E}_i^*$, which are biorthogonal with respect to the primal eigenmodes, as shown below.

### 8.3.3 Decomposition into the Basis of Approximate Eigenmodes

If we want to write an arbitrary snapshot as a linear combination of the normalized approximate eigenmodes

$$X \approx \sum_i a_i E_i^*,$$

the unknown coefficients $a_i$ can be obtained in either the original or the wavelet domain.

In the original domain, we can calculate inner products with the eigenmodes, or with the dual eigenmodes in the biorthogonal case:

- In case an orthonormal transform is used:

$$a_i = \langle X, E_i^* \rangle.$$

- In case a biorthonormal transform is used:

$$a_i = \langle X, \widetilde{E}_i^* \rangle.$$

However, we can also obtain the coefficients $a_i$ by calculating them in the wavelet domain. Let $\widehat{X}'$ be the result of the wavelet transform and compression of the input image $X$:

$$\begin{aligned} \widehat{X} &= W(X), \\ \widehat{X} &\triangleright \widehat{X}'. \end{aligned}$$

Due to the (bi)orthonormality of the wavelet packet transform and the orthogonality of the eigenmodes $\widehat{E}_i'$ of the transformed and compressed snapshots, the unknown coefficients $a_i$ are given by

$$a_i = \langle \widehat{X}', \widehat{E}_i' \rangle.$$

Indeed, we have in the biorthonormal case:

$$\begin{aligned} a_i &= \langle X, \widetilde{E}_i^* \rangle, \\ &= \langle \sum_j b_j \Psi_j, \sum_k c_{ik} \widetilde{\Psi}_k \rangle, \\ &= \sum_{j,k} b_j c_{ik} \langle \Psi_j, \widetilde{\Psi}_k \rangle, \\ &= \sum_j b_j c_{ij}, \\ &= \langle B, C_i \rangle, \\ &= \langle \widehat{X}', \widehat{E}_i' \rangle, \end{aligned}$$

with $\Psi_i$ and $\widetilde{\Psi}_i$ the biorthonormal basis functions in the wavelet domain, and $B = [b_i]$ and $C = [c_{i,j}]$.

Reconstruction using the wavelet domain is straightforward: first we reconstruct $\widehat{X}'$ in the wavelet domain, followed by a decompression step and the inverse transform $W^{-1}$:

$$\begin{aligned} \widehat{X}' &= \sum_i a_i \widehat{E}_i', \\ \widehat{X}' &\triangleleft \widehat{X}, \\ X &\approx W^{-1}(\widehat{X}). \end{aligned}$$

Which approach for the calculation of the coefficients $a_i$ is most efficient depends on a number of factors. If we denote the number of retained eigenmodes by $s$ and we assume that the images are square and count $\sqrt{N} \times \sqrt{N}$ pixels, we have:

- in the original domain:
  - $s$ inner products in $\mathbf{R}^N$: $O(sN)$,
- in the wavelet domain:
  - $S$ wavelet packet transforms: $O(SN \log \sqrt{N})$,
  - $s$ inner products in $\mathbf{R}^n$: $O(sn)$.

## 8.4 Wavelet Transforms

Let us denote the primal filter pair that is characteristic for a specific wavelet transform by $(g,h)$ and the dual filter pair by $(\widetilde{g},\widetilde{h})$.

We recall that while the periodic extension of a discrete (sampled) signal is straightforward and unique, there are four possible ways to extend a discrete signal symmetrically (Figure 3.7). However, to ensure to get a perfect reconstruction, one has to apply the correct extension in every step of the decomposition and reconstruction, as indicated in Figure 3.8 [4], cfr. Section 3.5.1.

All wavelet transform operations on a vector can be denoted by matrix multiplications. We use the following notations:

**W** Transformation matrix for the forward wavelet transform algorithm, using the filter pair $(g,h)$,

$\mathbf{W^{-1}}$ Transformation matrix for the inverse wavelet transform algorithm, using the filter pair $(g,h)$,

$\mathbf{W^{T}}$ Transposition (adjoint) of the transformation matrix for the forward wavelet transform algorithm, using the filter pair $(g,h)$,

$\mathbf{W^{-T}}$ Transposition (adjoint) of the transformation matrix for the inverse wavelet transform algorithm, using the filter pair $(g,h)$.

The transform matrices for the corresponding dual transforms using the dual filter pair $(\tilde{g},\tilde{h})$ are denoted by adding a tilde, e.g. $\widetilde{W}$.

The properties of these transformation matrices depend on the type of wavelet transform (orthogonal vs. biorthogonal) and the type of signal extension (periodic vs. symmetric). We can distinguish the following cases:

## Case 1

This is the generic case, i.e. for biorthogonal wavelets in combination with symmetric extension:

| **Forward/Inverse FWT** | **Orthogonality** |
|---|---|
| $W^{-1}\cdot\widetilde{W}=I$ | $W^{-T}\perp\widetilde{W}$ |
| $\widetilde{W}\cdot W^{-1}=I$ | $\widetilde{W}^{T}\perp W^{-1}$ |
| $\widetilde{W}^{-1}\cdot W=I$ | $\widetilde{W}^{-T}\perp W$ |
| $W\cdot\widetilde{W}^{-1}=I$ | $W^{T}\perp\widetilde{W}^{-1}$ |

## Case 2

If we use biorthogonal wavelets in combination with periodic extension, we can make the following simplifications:

$$
\begin{aligned}
W^{T} &\equiv W^{-1},\\
W^{-T} &\equiv W,\\
\widetilde{W}^{T} &\equiv \widetilde{W}^{-1},\\
\widetilde{W}^{-T} &\equiv \widetilde{W},
\end{aligned}
$$

and we get:

| Forward/Inverse FWT | Orthogonality |
|---|---|
| $W^{-1} \cdot \widetilde{W} = I$<br>$\widetilde{W} \cdot W^{-1} = I$<br>$\widetilde{W}^{-1} \cdot W = I$<br>$W \cdot \widetilde{W}^{-1} = I$ | $W \perp \widetilde{W}$<br>$\widetilde{W}^{-1} \perp W^{-1}$ |

### Case 3

If we use orthogonal wavelets, we can simplify even further:

$$\begin{aligned} \widetilde{W} &\equiv W, \\ \widetilde{W}^{-1} &\equiv W^{-1}, \end{aligned}$$

and we get:

| Forward/Inverse FWT | Orthogonality |
|---|---|
| $W^{-1} \cdot W = I$<br>$W \cdot W^{-1} = I$ | $W \perp W$<br>$W^{-1} \perp W^{-1}$ |

In most wavelet transform implementations, one has to care only about the *forward/inverse FWT* properties. However, for the APOD we also have to care about the *orthogonality* properties. Indeed, if we use a biorthogonal wavelet transform or wavelet packet transform, we will end up with *primal* and *dual* eigenmodes ($E_i^*$ and $\widetilde{E}_i^*$) that need to be biorthogonal to be able to decompose a random snapshot into a precalculated basis of eigenmodes.

## 8.5 Results

We have evaluated the use of the Approximate Proper Orthogonal Decomposition to represent and to analyze the dynamical behavior of the discretization of a system of partial differential equations (PDEs), namely a reaction-diffusion system.

### 8.5.1 Wavelets

Because of their symmetry, we have chosen for biorthogonal wavelets. In our test cases, we used biorthogonal Cohen-Daubechies-Feauveau wavelets [9], in particular the member with 2 vanishing moments for both the primal and the dual wavelet (Figure 2.4), with symmetric signal extension.

The cost criterion used for choosing the “best basis” among all possible wavelet packet bases is based on the logarithm of the energy

$$\text{cost} = \sum_i \log x_i^2,$$

which is equivalent with

$$\text{cost} = \sum_i \log |x_i|.$$

### 8.5.2 The Dynamical System

The dynamical system we want to analyze is described by a system of partial differential equations. We evaluated the performance of the APOD using the one-dimensional Brusselator reaction-diffusion system [22]:

$$\begin{aligned}
\frac{\partial X}{\partial t} &= \frac{D_X}{L^2}\frac{\partial^2 X}{\partial z^2} + X^2 Y - (B+1)X + A, \\
\frac{\partial Y}{\partial t} &= \frac{D_Y}{L^2}\frac{\partial^2 Y}{\partial z^2} - X^2 Y + BX,
\end{aligned}$$

with Dirichlet boundary conditions

$$\begin{aligned}
X(t,z=0) &= X(t,z=1) &= A, \\
Y(t,z=0) &= Y(t,z=1) &= B/A.
\end{aligned}$$

The reactor length $L$ is used as a free parameter (continuation parameter), while we keep the other parameters fixed:

$$\begin{aligned}
A &= 2.0, \\
B &= 5.45, \\
D_X &= 0.008, \\
D_Y &= 0.004.
\end{aligned}$$

Branches of periodic solutions bifurcate from the trivial steady state branch ($X = A, Y = B/A$), as can be seen in the bifurcation diagram in Figure 8.2 [32].

### 8.5.3 Snapshots

To create a low-dimensional model of our system, we started with four sets of snapshots, calculated for four values of the continuation parameter $L$, using time integration with a time step $\Delta t = 0.2$, as shown in Table 8.1.

Each snapshot is a vector with 256 elements, containing the concentration $X$ of the first reagent. Figure 8.3 gives a graphical representation of the concentration $X$ for the different parameter values. White indicates a high concentration, black a low concentration. The $z$ (spatial) axis runs horizontally, while the $t$ (time) axis runs vertically (time runs from the top to the bottom).

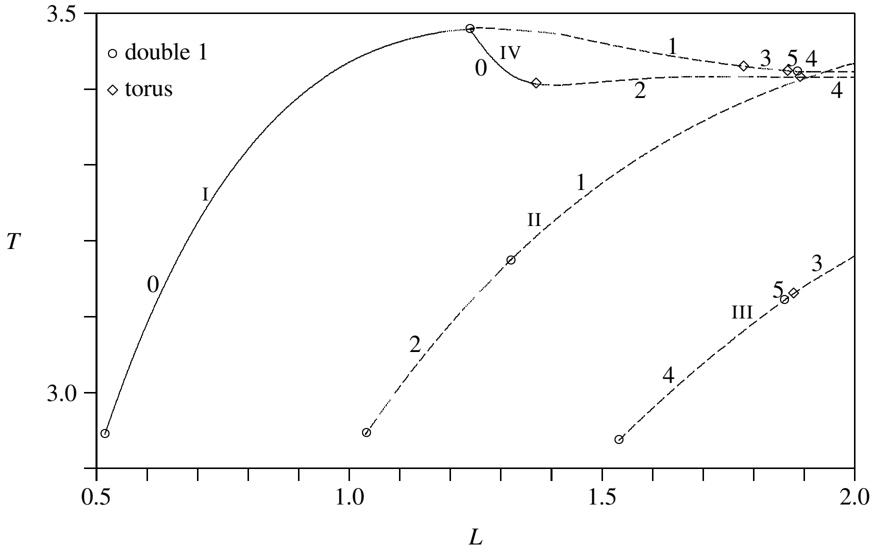



Figure 8.2: Bifurcation diagram of periodic (branches I–IV) and quasi-periodic (branch V) solutions for the Brusselator model, period $T$ versus the reactor length $L$ [32]. Note that this diagram is calculated using a 31-point discretization and that it is not complete.

| Set | $L$ | Branch | # snapshots | |
|---|---|---|---|---|
| $A$ | 0.75 | I | 20 | Periodic solution |
| $B$ | 1.25 | I | 20 | Periodic solution (before period-doubling) |
| $C$ | 1.43 | V | 192 | Quasi-periodic solution (torus) |
| $D$ | 1.50 | IV | 20 | Periodic solution |
| **Total** | | | **252** | |

Table 8.1: The snapshots used for testing and their bifurcation parameter values.

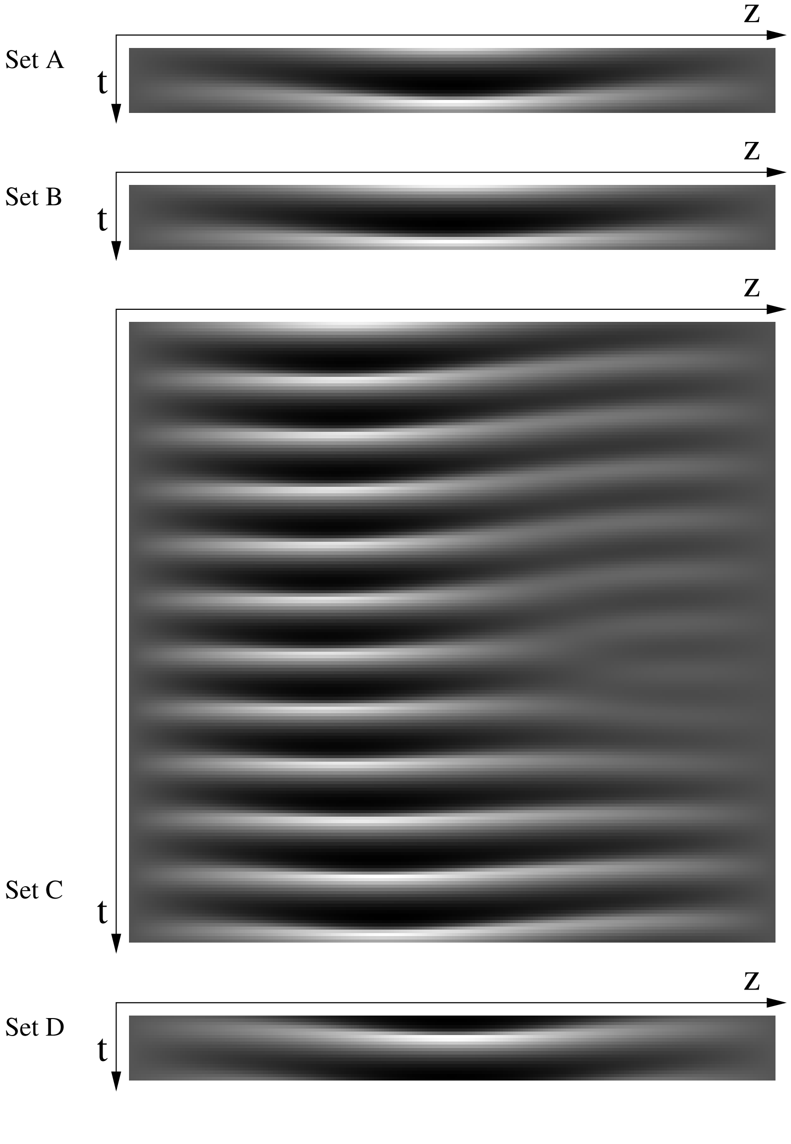



Figure 8.3: The four sets of input snapshots calculated for four values of the continuation parameter $L$ (cfr. Table 8.1), using time integration. Each snapshot is a vector with 256 elements, containing the concentration $X$ of the first reagent. White indicates a high concentration, black a low concentration.

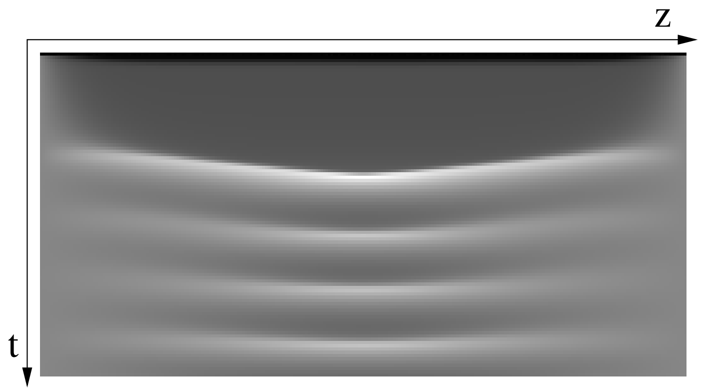



Figure 8.4: The set of 100 “strange” input snapshots, calculated for $L = 1.40$ using time integration. Each snapshot is a vector with 256 elements, containing the concentration $X$ of the first reagent. White indicates a high concentration, black a low concentration.

### 8.5.4 “Strange” Snapshots

To test how well this low-dimensional model can represent the dynamical behavior of the parameter dependent Brusselator model, we decomposed a set of “strange” snapshots — snapshots not belonging to the set of snapshots we used to construct the APOD model — in the APOD basis. The set consists of 100 snapshots for $L = 1.40$, obtained by time integration with a time step $\Delta t = 0.2$ starting from the perturbed trivial solution. The “strange” snapshots show a periodic solution including the transient behavior.

### 8.5.5 Eigenvalue Distribution

Since the snapshots were normalized by subtracting their mean and by rescaling them (see 8.1), the eigenvalues are equal to the fraction of captured energy.

To approximate the snapshots by using only a few eigenmodes, we need to keep the eigenmodes which capture the most energy and hence have the largest eigenvalues.

In Figure 8.5 the sum of the largest $i$ eigenvalues is plotted for different compression rates. The plot reveals that e.g. the 4 most important eigenmodes capture more than 99% of the energy present in the set of snapshots.

Observe that for a fixed number of eigenmodes the captured energy is increasing in function of the compression ratio. Due to the lossy compression some small non-correlated information is lost, and the resulting data is thus more correlated, resulting in the effect that the same number of eigenvalues capture more energy.

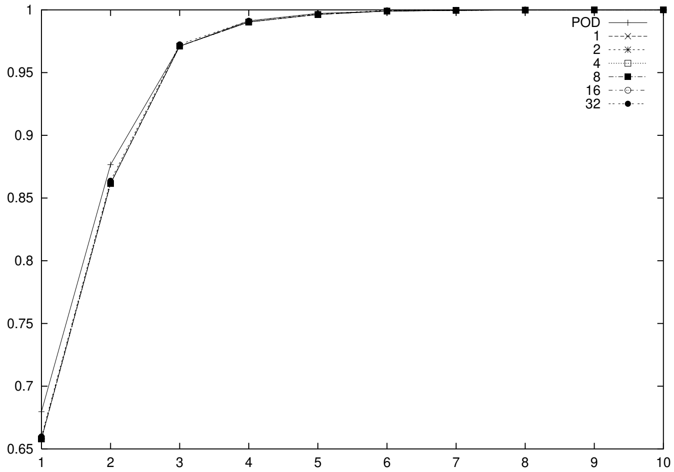



Figure 8.5: Sum of the eigenvalues vs. number of eigenvalues, both for the POD and the APOD with different compression rates.

Note that in the case of compression rate 32, we retain only 8 eigenmodes because the dimension of the compressed system can not be higher due to the large compression ratio ($256/32 = 8$).

The distribution of the eigenvalues is a measure for the decorrelation achieved by the (approximate) POD. If many eigenvalues are very small, the original set of snapshots was highly correlated and can be represented well by only a few eigenmodes. In contrast, if all eigenvalues have about the same magnitude, the set of snapshots was not correlated at all.

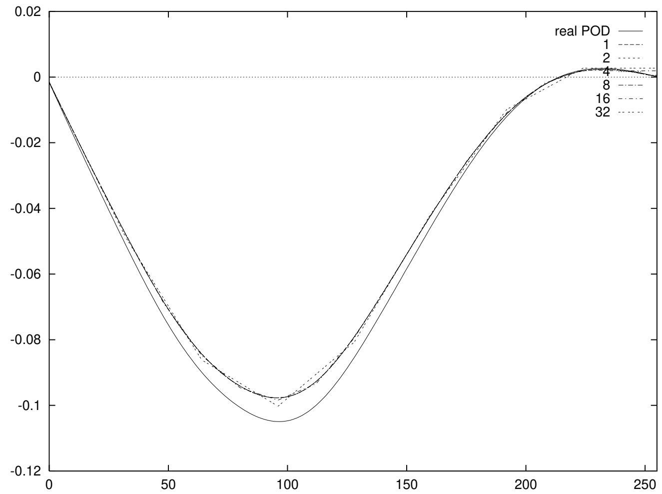
### 8.5.6 Eigenmodes

The 4 most important eigenmodes, both for the POD and the Approximate POD with different compression rates, are shown in Figures 8.6 and 8.7.

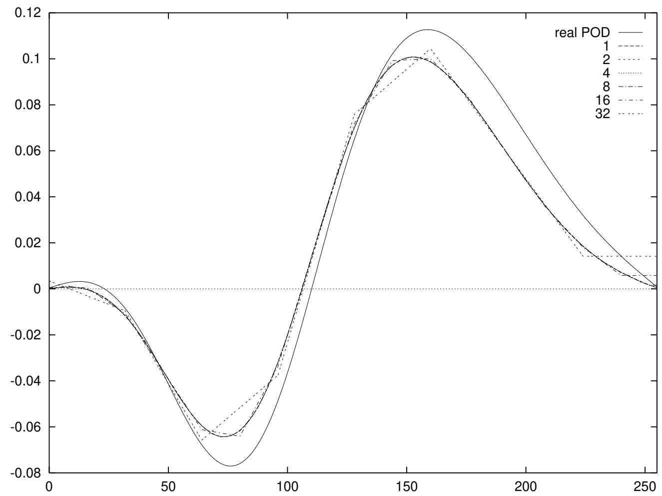
The difference between the results for the POD and the Approximate POD using compression rate 1 (i.e. no compression) is due to the biorthogonality of the wavelet packet basis. If we would have used an orthogonal wavelet packet basis, there would be no difference.

### 8.5.7 Reconstruction of the Snapshots

If we reconstruct the snapshots (both the set used for the construction of the APOD model and the “strange” set) using only the most important eigenmodes, we get

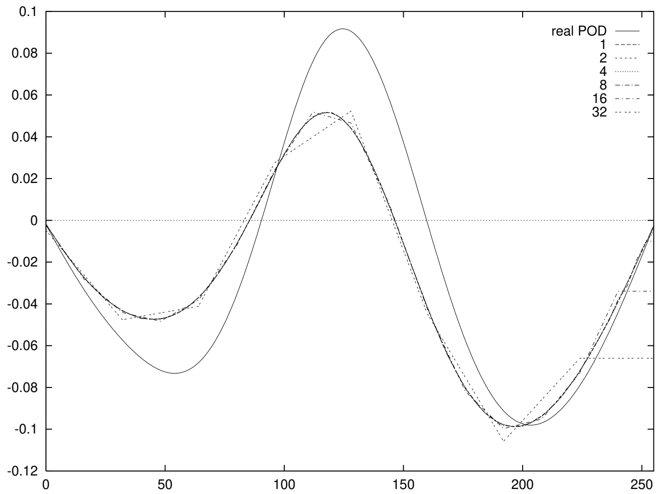



Eigenmode 1

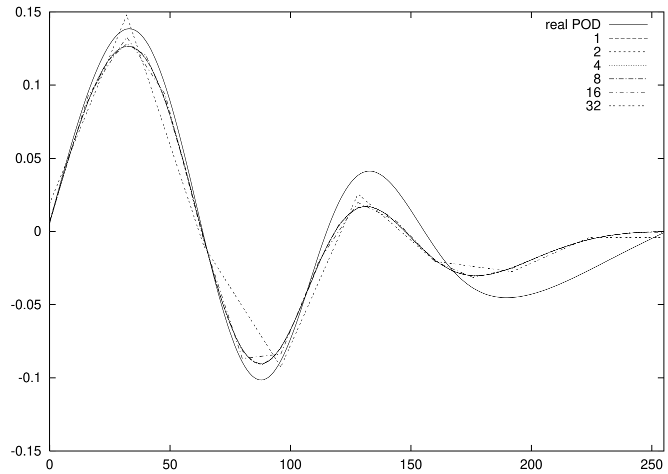



Eigenmode 2

Figure 8.6: The 4 most important eigenmodes for the POD and for the APOD with different compression rates.




Eigenmode 3




Eigenmode 4

Figure 8.7: The 4 most important eigenmodes for the POD and for the APOD with different compression rates. (cont'd)

| Number of eigenmodes | POD | APOD with compression rate | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 4 | 8 | 16 | 32 |
| 1 | 67.97 | 68.11 | 68.11 | 68.11 | 68.11 | 68.10 | 67.97 |
| 2 | 87.66 | 87.69 | 87.69 | 87.69 | 87.69 | 87.64 | 87.04 |
| 3 | 97.10 | 97.17 | 97.17 | 97.17 | 97.17 | 97.17 | 96.83 |
| 4 | 99.13 | 99.10 | 99.10 | 99.10 | 99.09 | 99.07 | 98.50 |
| 5 | 99.72 | 99.68 | 99.68 | 99.68 | 99.68 | 99.65 | 99.04 |
| 6 | 99.89 | 99.90 | 99.90 | 99.90 | 99.89 | 99.86 | 99.23 |
| 7 | 99.95 | 99.94 | 99.94 | 99.94 | 99.93 | 99.90 | 99.24 |
| 8 | 99.99 | 99.98 | 99.98 | 99.98 | 99.97 | 99.93 | 99.25 |
| 9 | 99.99 | 99.99 | 99.99 | 99.99 | 99.98 | 99.94 | |
| 10 | 100.00 | 99.99 | 99.99 | 99.99 | 99.99 | 99.95 | |

Table 8.2: Mean percentage of energy after reconstruction of the snapshots vs. number of used eigenmodes for different compression rates.

the following results. Tables 8.2 and 8.3 show the mean percentage of captured energy (over all reconstructed snapshots) after reconstruction by using the most important eigenmodes for different compression ratios.

### 8.5.8 Coefficients of the Snapshots in the APOD Basis

We can analyze the behavior of the dynamical system by analyzing the coefficients of the projection of the snapshots onto the APOD basis. Figure 8.8 shows these coefficients for the two most important eigenmodes. The first 20 snapshots are from set $A$, the next 20 from set $B$ and so on (cfr. Table 8.1). The periodicity of the solutions is clearly visible. There is no visible difference among the coefficients for the APOD bases with different compression rates.

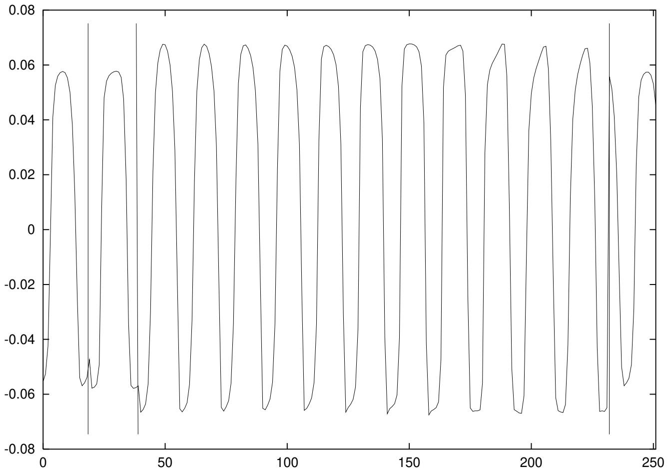
Figure 8.9 shows how the maximum error in the coefficients of the 10 most important eigenmodes depends on the compression rate.

The same results for the decomposition of the "strange" snapshots in the APOD basis are shown in the Figures 8.10 and 8.11.

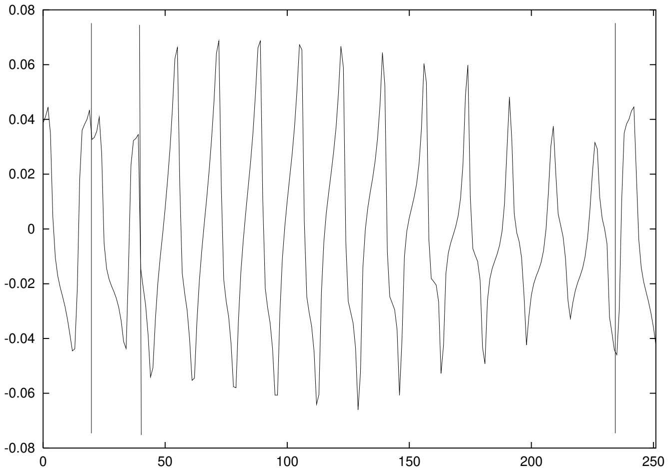
### 8.5.9 Timings

All timings were obtained on one node of an IBM SP2 cluster of RS/6000 POWER2 thin nodes, with an implementation in C++ using LAPACK++.

The normalization of the snapshots took 0.32 seconds. The decomposition timings for the APOD are shown in Table 8.4. As a comparison, the real POD required 17.40 seconds. Note that the wavelet packet transform routines used — taking ca. 25 seconds — were very generic and were not optimized at all. A

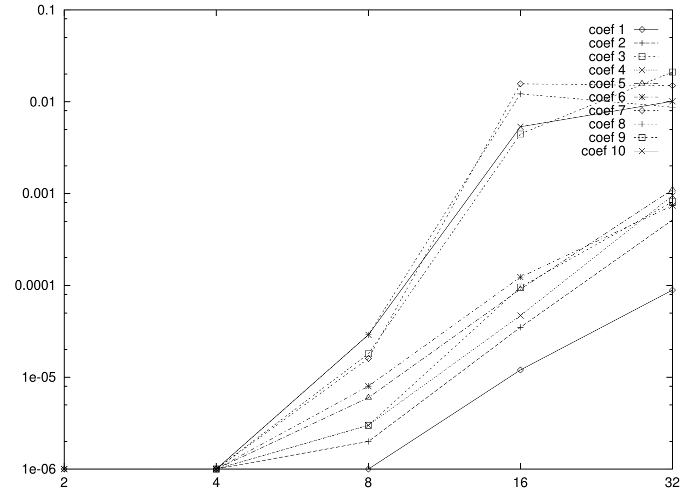



Coefficient 1

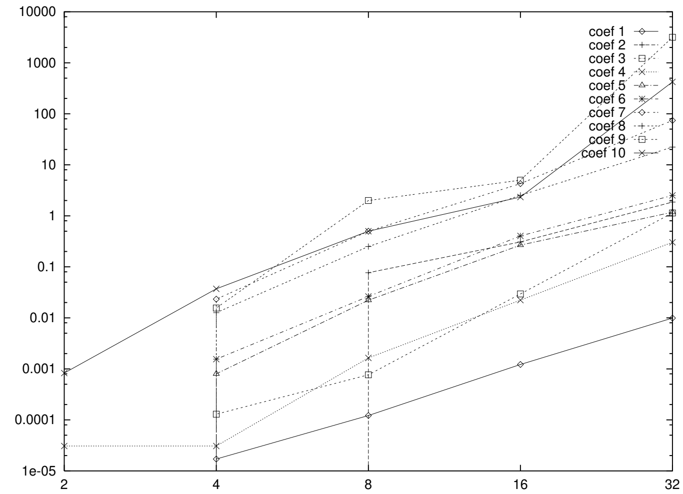



Coefficient 2

Figure 8.8: First two coefficients of the decomposition of the snapshots into the best basis.




Absolute errors




Relative errors

Figure 8.9: Maximum error in the coefficients (compared to the case without compression) for different compression rates.

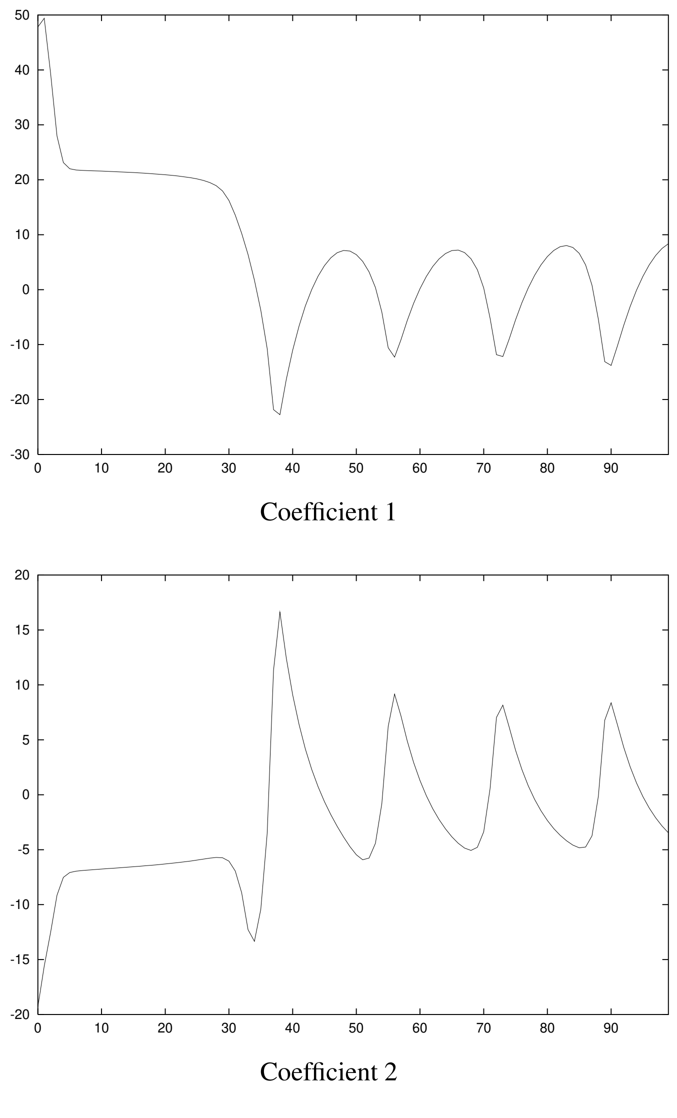



Figure 8.10: First two coefficients of the decomposition of the "strange" snapshots into the best basis.

| Number of eigenmodes | APOD with compression rate | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 4 | 8 | 16 | 32 |
| 1 | 52.87 | 52.87 | 52.87 | 52.87 | 52.87 | 52.81 |
| 2 | 71.62 | 71.62 | 71.62 | 71.62 | 71.60 | 71.34 |
| 3 | 91.75 | 91.75 | 91.75 | 91.74 | 91.79 | 91.90 |
| 4 | 96.87 | 96.87 | 96.87 | 96.87 | 96.90 | 96.84 |
| 5 | 97.40 | 97.40 | 97.40 | 97.39 | 97.43 | 97.52 |
| 6 | 98.22 | 98.22 | 98.22 | 98.21 | 98.23 | 98.32 |
| 7 | 99.12 | 99.12 | 99.12 | 99.10 | 99.16 | 99.90 |
| 8 | 99.09 | 99.09 | 99.09 | 99.07 | 99.11 | 99.88 |
| 9 | 99.32 | 99.32 | 99.32 | 99.30 | 99.37 | |
| 10 | 99.43 | 99.43 | 99.43 | 99.40 | 99.47 | |

Table 8.3: Mean percentage of energy after reconstruction of the "strange" snapshots vs. number of used eigenmodes for different compression rates.

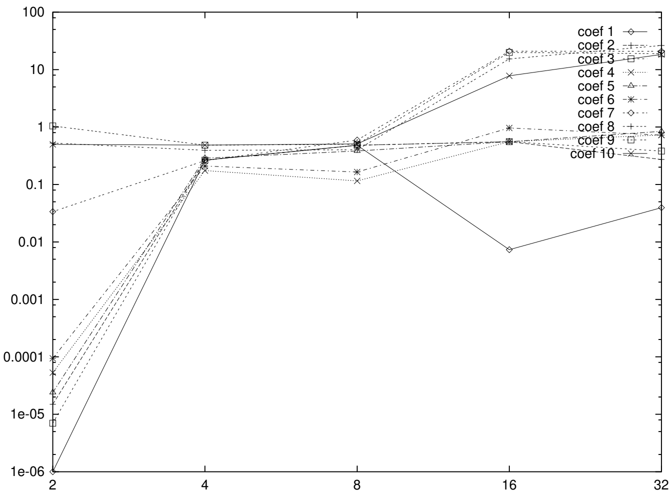



Figure 8.11: Maximum absolute error in the coefficients (compared to the case without compression) for different compression rates.

| Compression rate | Calculation time | |
|---|---|---|
| 1 | 45.30 | $\approx$ 25+17.4 |
| 2 | 37.30 | $\approx$ 25+9 |
| 4 | 31.15 | |
| 8 | 27.45 | |
| 16 | 25.62 | |
| 32 | 25.40 | $\approx$ 25+0.5 |

Table 8.4: Computation times (in seconds) for the Approximate Proper Orthogonal Decomposition for the various compression ratios. The real POD required ca. 17.40 seconds. The wavelet packet transform took ca. 25 seconds. The third column shows that the total cost of the APOD is approximately equal to the sum of the cost of the wavelet packet transform (ca. 25 seconds) and the cost of the construction of the matrix $C^*$ for the POD of the reduced system (proportional to the reciprocal of the compression ratio).

| Method | Complexity |
|---|---|
| POD | $O(NS^2) + O(S^3)$ |
| WPT | $O(SN \log N)$ |
| APOD | $O(SN \log N) + O(S^2 N/C) + O(S^3)$ |

Table 8.5: Comparison of the complexity of the various methods. $S$ is the number of snapshots, $N$ is the size of a snapshot, and $C$ is the compression rate.

cautious implementation using the lifting scheme algorithm will speed it up an order of magnitude. But at the time this research was done, the lifting scheme had just emerged and was not mature yet.

A comparison of the complexity of the various methods is shown in Table 8.5. However, the factor $\log N$ is only valid for the one-dimensional case we considered here. For higher dimensions it becomes

$$\log \sqrt[d]{N},$$

with $d$ the dimension of the snapshots and thus $\sqrt[d]{N}$ the number of data points per dimension. This factor decreases quickly for increasing dimensions, so for higher dimensions we eventually win using the APOD.

The selected wavelet packet basis can also be fixed in advance, instead of being determined adaptively by a best basis search. In that case the factor $\log N$ must be replaced by a number between 1 and $\log N$, depending on the chosen basis. E.g. for an ordinary wavelet transform, that factor is 1.

## 8.6 Conclusions

The Approximate Proper Orthogonal Decomposition using biorthogonal wavelet packets allows a good reconstruction of the original data, as is shown by the amount of captured energy. We only need very few eigenmodes (basis functions), while methods based on other approximations like the Discrete Cosine Transform (DCT) and the Fast Wavelet Transform (FWT) require much more basis functions. One can analyze the behavior of a dynamical system by interpreting the most important coefficients of the decomposition in the APOD basis.

The decomposition can be calculated in both the original domain and in the wavelet packet domain. The eigenmodes are not localized, but the underlying wavelet packets are, and thus the calculation of inner products in the wavelet packet domain is a cheaper operation.

This is useful for the construction of low-dimensional approximate models for PDEs (reaction-diffusion, Navier-Stokes, ...). When assuming the solution of a PDE can be written as

$$u(x,t) = \sum_{i=0}^{K-1} a_i(t) E_i^*(x),$$

the PDE can be transformed into the form

$$\begin{aligned} \frac{da_i(t)}{dt} &= f(a_0, \cdots, a_{K-1}) \\ &= c_i + c_{ij}a_j + c_{ijk}a_j a_k + \cdots \end{aligned}$$

by means of a Galerkin projection. The Galerkin projection requires $O(K^3)$ inner products of vectors of size $N/C$, which can be sped up by using the properties of wavelet packets (localization, biorthogonality).

Another application is the compressed storage of related data. In [45] Sirovich et al. suggest to use the POD to store a library of pictures of human faces by storing the most important POD eigenmodes and, for each face (snapshot), the coefficients in this basis. The APOD allows a further storage reduction by storing the compressed eigenmodes.

# Chapter 9

# Conclusions and Suggestions for Future Research

## 9.1 Wavelet Transforms

The wavelet transform allows to decorrelate data because wavelets are localized in both space/time and scale/frequency, and because they are based on a multi-resolution analysis. The lifting scheme is an efficient implementation of a wavelet transform algorithm. Furthermore it can easily be converted to operate on integer data. The inverse transform is trivial to find and always provides perfect reconstruction of the input data. We have worked out the details of integer wavelet transforms for a class of biorthogonal wavelets (Cohen-Daubechies-Feauveau).

Not only can the lifting scheme be used to calculate all classical biorthogonal wavelet transforms, it also can be used as a paradigm to design so-called second-generation wavelets: wavelets that are not based on simple translates and dilates of a one-dimensional function. An example of this are the new wavelets we created on a rectangular grid, based on a Red-Black splitting scheme. These Red-Black wavelets are non-separable two-dimensional wavelets, unlike classical tensor-product wavelets. They are less anisotropic and do not favor the horizontal and vertical directions that strongly, as we have shown in a denoising application. Furthermore the idea behind Red-Black wavelets can easily be applied to higher dimensions and more complex geometries.

## 9.2 A Software Library for Wavelet-Based Image Processing

We developed the C++ software library WAILI (Wavelets with Integer Lifting), which implements wavelet transforms and wavelet-based image processing operations. WAILI is built on the framework of the integer version of the lifting scheme.

The library implements several wavelet transforms belonging to the biorthogonal family of Cohen-Daubechies-Feauveau, on images of arbitrary dimensions. Other wavelet transforms can easily be added, given a decomposition in lifting steps.

WAILI provides image processing operations on wavelet-transformed images. Soft-thresholding using generalized cross validation is an efficient and well-performing method for noise reduction. Other important operations are crop and merge: due to the spatial locality of wavelets, subimages can be cut out or pasted in a larger wavelet-transformed image without retransforming them, while affecting only a limited set of wavelet coefficients.

We also extended WAILI to handle very large images. By tiling the individual wavelet subbands, in a way transparent to the user, large images can be transformed and processed. This is thus done in a block-based manner under supervision of a block manager. The division in blocks also has a positive influence on the performance of the denoising algorithm: since denoising works on the independent blocks, it adapts better to the local properties of the image.

Because most operations require communication of only a limited strip of wavelet coefficients (e.g. lifting steps), or even no communication at all (e.g. denoising), this scheme has potential for parallelization.

## 9.3 Approximate POD using Wavelet Packets

Due to its excellent decorrelation capabilities, a wavelet packet transform can serve as a good approximation for the Proper Orthogonal Decomposition (POD), which is the theoretical ideal decorrelation method. By compressing the data in the wavelet packet domain, we can dramatically reduce the complexity of the system that is being analyzed.

We used the Approximate Proper Orthogonal Decomposition (APOD) to analyze precalculated solution profiles for a complex dynamical system. We have done experiments with a reaction-diffusion system, described by a partial differential equation (PDE). The APOD allowed a good reconstruction of the original data, requiring only very few eigenmodes, and thus revealed the low-dimensional intrinsics of the dynamical system we are interested in. The APOD can be used to construct a low-dimensional model for the original problem, allowing fast and easy analysis of the dynamical system.

## 9.4 Suggestions for Future Research

We just suggest two directions for future research:

1. Modern microprocessors have additional instructions for the efficient parallel execution of operations that are heavily used in multi-media applications. Examples are:

   - Altivec from Motorola on the PowerPC,
   - MAX-2 (Multimedia Acceleration eXtensions 2) from Hewlett-Packard on the PA-RISC,
   - MMX (MultiMedia eXtension) from Intel on the Pentium MMX, Pentium II, and Pentium III,
   - MVI (Motion Video Instruction) from Compaq (formerly Digital) on the Alpha AXP,
   - VIS (Visual Instruction Set) from Sun Microsystems on the SPARC V9 architecture.

   Using these instructions, it should be possible to implement a highly optimized version of the integer wavelet transform and plug this into WAILI, yielding a serious performance boost.

2. Currently the extension of WAILI for handling very large images divided in blocks has not been parallelized yet. Modifying the library for parallel execution would be an interesting topic.

# Bibliography

[1] E. H. Adelson, E. Simoncelli, and R. Hingorani. Orthogonal pyramid transforms for image coding. In *Visual Communications and Image Processing* II, number 845 in SPIE, pages 50–58, 1987.

[2] Ph. Bekaert, G. Uytterhoeven, and Y.D. Willems. An experiment with wavelet image coding. In Martin Sperka, editor, *Proceedings of the Eleventh Spring Conference on Computer Graphics, Bratislava, Slovakia*, pages 1–6, June 1995.

[3] J. Benedetto and M. Frazier, editors. *Wavelets: Mathematics and Applications*. CRC Press, Boca Raton, Florida, 1993.

[4] C. M. Brislawn. Classification of nonexpansive symmetric extension transforms for multirate filter banks. *Appl. Comput. Harmon. Anal.*, 3:337–357, 1996.

[5] S. Cabrera, V. Kreinovich, and O. Sirisaengtaksin. Wavelets compress better than all other methods: A 1-dimensional theorem. Technical Report 25, University of Texas at El Paso, El Paso, TX 79968, 1992.

[6] R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo. Wavelet transforms that map integers to integers. *Appl. Comput. Harmon. Anal.*, 5(3):332–369, 1998.

[7] C.K. Chui. *An Introduction to Wavelets*, volume 1 of *Wavelet Analysis and its Applications*. Academic Press, Boston, 1992.

[8] C.K. Chui, editor. *Wavelets: A Tutorial in Theory and Applications*, volume 2 of *Wavelet Analysis and its Applications*. Academic Press, Boston, 1992.

[9] A. Cohen, I. Daubechies, and J. Feauveau. Bi-orthogonal bases of compactly supported wavelets. *Comm. Pure Appl. Math.*, 45:485–560, 1992.

[10] R.R. Coifman and V. Wickerhauser. Wavelets and adapted waveform analysis. In J.J. Benedetto and M. Frazier, editors, *Wavelets: mathematics and applications*, chapter 10. CRC Press, 1993.

[11] J. M. Combes, A. Grossmann, and Ph. Tchamitchian, editors. *Wavelets: Time-Frequency Methods and Phase Space*. Inverse problems and theoretical imaging. Springer-Verlag, New York, 1989.

[12] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Comm. Pure Appl. Math.*, 41:909–996, 1988.

[13] I. Daubechies. *Ten Lectures on Wavelets*. Number 61 in CBMS-NSF Series in Applied Mathematics. SIAM, Philadelphia, 1992.

[14] I. Daubechies and W. Sweldens. Factoring wavelet transforms into lifting steps. *J. Fourier Anal. Appl.*, 4(3):245–267, 1998.

[15] R. A. DeVore, B. Jawerth, and B. J. Lucier. Image compression through wavelet transform coding. *IEEE Trans. Inform. Theory*, 38(2):719–746, 1992.

[16] D. L. Donoho and I. M. Johnstone. Ideal spatial adaptation via wavelet shrinkage. *Biometrika*, 81:425–455, 1994.

[17] D. L. Donoho and I. M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *J. Amer. Statist. Assoc.*, 90:1200–1224, 1995.

[18] D. L. Donoho, I. M. Johnstone, G. Kerkyacharian, and D. Picard. Wavelet shrinkage: Asymptopia? *J. Roy. Statist. Soc. Ser. B*, 57(2):301–369, 1995.

[19] G. Fernández, S. Periaswamy, and Wim Sweldens. LIFTPACK: A software package for wavelet transforms using lifting. In M. Unser, A. Aldroubi, and A. F. Laine, editors, *Wavelet Applications in Signal and Image Processing IV*, pages 396–408. Proc. SPIE 2825, 1996.

[20] R. Fletcher. *Practical methods of optimization*. Wiley-interscience publications. Wiley, Chichester, 1981.

[21] A. Grossmann, R. Kronland-Martinet, and J. Morlet. Reading and understanding continuous wavelet transforms. In [11], pages 2–20, 1989.

[22] M. Holodniok, P. Knedlik, and M. Kubiček. Continuation of periodic solutions in parabolic differential equations. In T. Küpper, R. Seydel, and H. Troger, editors, *Bifurcation: Analysis, Algorithms, Applications*, ISNM 79, pages 122–130. Birkhäuser, Basel, 1987.

[23] M. Jansen and A. Bultheel. Multiple wavelet threshold estimation by generalised cross validation for images with correlated noise. *IEEE Transactions on Image Processing*, 1998. Accepted.

[24] M. Jansen, M. Malfait, and A. Bultheel. Generalized cross validation for wavelet thresholding. *Signal Processing*, 56(1):33–44, January 1997.

[25] M. Jansen, G. Uytterhoeven, and A. Bultheel. Image de-noising by integer wavelet transforms and generalized cross validation. *Medical Physics*, 1999. to appear.

[26] B. Jawerth and W. Sweldens. An overview of wavelet based multiresolution analyses. *SIAM Rev.*, 36(3):377–412, 1994.

[27] I. M. Johnstone and B. W. Silverman. Wavelet threshold estimators for data with correlated noise. *J. Roy. Statist. Soc. Ser. B*, 59:319–351, 1997.

[28] M. Kobayashi. *Applications of Wavelets. Case studies*. SIAM, 1998.

[29] T.H. Koornwinder, editor. *Wavelets: an elementary treatment of theory and applications*, volume 1 of *Series in Approximations & Decompositions*. World Scientific, 1993.

[30] J. Kovačević and M. Vetterli. Nonseparable multidimensional perfect reconstruction filter banks and wavelet bases for $\mathbf{R}^n$. *IEEE Trans. Inform. Theory*, 38(2):533–555, March 1992.

[31] J. Kovačević and W. Sweldens. Wavelet families of increasing order in arbitrary dimensions. Technical report, Bell Laboratories, Lucent Technologies, 1997.

[32] K. Lust, D. Roose, A. Spence, and A.R. Champneys. An adaptive Newton-Picard algorithm with subspace iteration for computing periodic solutions. *SIAM J. Sci. Comput.*, 19(4):1188–1209, 1998.

[33] S. Mallat, editor. *A wavelet tour of signal processing*. Academic Press, 1998.

[34] S. G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. Patt. Anal. Mach. Intell.*, 11(7):674–693, 1989.

[35] S. G. Mallat. Multifrequency channel decompositions of images and wavelet models. *IEEE Trans. Acoust. Speech Signal Process.*, 37(12):2091–2110, 1989.

[36] Y. Meyer. *Wavelets: Algorithms and Applications*. SIAM, Philadelphia, 1993. translated and revised by R.D. Ryan.

[37] G. P. Nason and B. W. Silverman. The stationary wavelet transform and some statistical applications. In A. Antoniadis and G. Oppenheim, editors, *Wavelets and Statistics*, Lecture Notes in Statistics, pages 281–299, 1995.

[38] C. Poynton. Poynton's Color FAQ. February 1997.

[39] A. Said and W. A. Pearlman. Image compression using the spatial-orientation tree. In *Proc. IEEE Int. Symp. Circuits and Syst., Chicago, IL*, pages 279–282, May 1993.

[40] P. Schröder and W. Sweldens. Spherical wavelets: Efficiently representing functions on the sphere. *Computer Graphics, (SIGGRAPH '95 Proceedings)*, pages 161–172, 1995.

[41] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, July and October 1948.

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

[43] M. J. Shensa. The discrete wavelet transform: Wedding the à trous and Mallat algorithms. *IEEE Trans. Signal Process.*, 40(10):2464–2482, 1992.

[44] E.P. Simoncelli and E.H. Adelson. Non-separable extensions of quadrature mirror filters to multiple dimensions. *Proceedings of the IEEE*, 78:652–664, April 1990.

[45] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterisation of human faces. *J. Opt. Soc. Amer.*, 4(3):519–524, March 1987.

[46] G. Strang and T. Nguyen. *Wavelets and filter banks*. Wellesley-Cambridge Press, Box 812060, Wellesley MA 02181, fax 617-253-4358, 1996.

[47] W. Sweldens. The lifting scheme: A new philosophy in biorthogonal wavelet constructions. In A. F. Laine and M. Unser, editors, *Wavelet Applications in Signal and Image Processing III*, pages 68–79. Proc. SPIE 2569, 1995.

[48] W. Sweldens. The lifting scheme: A custom-design construction of biorthogonal wavelets. *Appl. Comput. Harmon. Anal.*, 3(2):186–200, 1996.

[49] W. Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal.*, 29(2):511–546, 1997.

[50] W. Sweldens and P. Schröder. Building your own wavelets at home. In *Wavelets in Computer Graphics, ACM SIGGRAPH 1996 Course Notes*. ACM Press, 1996.

[51] G. Uytterhoeven. Ontwerp en implementatie van compressiealgoritmen met multi-wavelets. Master's thesis, Departement Computerwetenschappen, Katholieke Universiteit Leuven, Belgium, 1994. (in Dutch).

[52] G. Uytterhoeven and A. Bultheel. The Red-Black wavelet transform. In *Proceedings of the IEEE Benelux Signal Processing Symposium, Leuven, Belgium*, pages 191–194, March 1998.

[53] G. Uytterhoeven and D. Roose. Experiments with a wavelet-based approximate proper orthogonal decomposition. TW Report 263, Department of Computer Science, Katholieke Universiteit Leuven, Belgium, September 1997.

[54] G. Uytterhoeven, D. Roose, and A. Bultheel. Applying integer wavelet transforms. In D. Roose and M. Moonen, editors, *Wavelet analysis: a new tool in signal and image processing*. Technologisch Instituut, KVIV, Antwerpen, December 1996.

[55] G. Uytterhoeven, D. Roose, and A. Bultheel. A wavelet toolbox for large scale image processing. In P. Zinterhof, M. Vajteršic, and A. Uhl, editors, *Parallel Computation*, volume 1557 of *Lecture Notes in Computer Science*, pages 337–346. Austrian Center for Parallel Computation, Springer-Verlag, February 1999.

[56] G. Uytterhoeven, F. Van Wulpen, M. Jansen, D. Roose, and A. Bultheel. WAILI: Wavelets with integer lifting. Software library, `http://www.cs.kuleuven.ac.be/~wavelets/`, 1997.

[57] G. Uytterhoeven, F. Van Wulpen, M. Jansen, D. Roose, and A. Bultheel. WAILI: A software library for image processing using integer wavelet transforms. In K.M. Hanson, editor, *Medical Imaging 1998: Image Processing*, volume 3338 of *SPIE Proceedings*, pages 1490–1501. The International Society for Optical Engineering, February 1998.

[58] M. Vetterli and J. Kovacevic. *Wavelets and subband coding*. Applied Mathematics and Mathematical Computation Series. Prentice Hall, Englewood Cliffs, 1995.

[59] G. K. Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34(4):30–44, 1991.

[60] N. Weyrich and G. T. Warhola. De-noising using wavelets and cross validation. In S.P. Singh, editor, *Approximation Theory, Wavelets and Applications*, volume 454 of *NATO ASI Series C*, pages 523–532, 1995.

[61] M.V. Wickerhauser. *Adapted Wavelet Analysis from Theory to Software*. A.K. Peters, 289 Linden Street, Wellesley, MA 02181, 1994.

[62] F. Van Wulpen. Implementatie van een interface voor het scaleren van beelden met wavelets. Master's thesis, Departement Computerwetenschappen, Katholieke Universiteit Leuven, Belgium, 1996. (in Dutch).

# Appendix A

# Lifting Decompositions

Sample decompositions for some analyzing filter pairs of the family of biorthogonal Cohen-Daubechies-Feauveau wavelets with $n$ ($\widetilde{n}$) vanishing moments for the primal (dual) wavelet [9]. Note that $n = 1, \widetilde{n} = 1$ is the Haar wavelet.

These decompositions are not unique due to the non-uniqueness of the division of Laurent polynomials. We have chosen the factorizations such to preserve symmetry in the lifting steps as much as possible.

| $n$ | $\widetilde{n}$ | |
|---|---|---|
| 1 | | $\widetilde{g}(z) = \frac{1}{2}z^{-1} - \frac{1}{2}$ |
| | | $t_1(z) = -1$ <br> $K_1 = 1$ |
| | 1 | $\widetilde{h}(z) = \frac{1}{2}z^{-1} + \frac{1}{2}$ |
| | | $s_1(z) = \frac{1}{2}$ <br> $K_2 = \frac{1}{2}$ |
| | 3 | $\widetilde{h}(z) = -\frac{1}{16}z^{-3} + \frac{1}{16}z^{-2} + \frac{1}{2}z^{-1} + \frac{1}{2} + \frac{1}{16}z - \frac{1}{16}z^2$ |
| | | $s_1(z) = -\frac{1}{16}z^{-1} + \frac{1}{2} + \frac{1}{16}z$ <br> $K_2 = \frac{1}{2}$ |
| | 5 | $\widetilde{h}(z) = \frac{3}{256}z^{-5} - \frac{3}{256}z^{-4} - \frac{11}{128}z^{-3} + \frac{11}{128}z^{-2} + \frac{1}{2}z^{-1} + \frac{1}{2} + \frac{11}{128}z - \frac{11}{128}z^2 - \frac{3}{256}z^3 + \frac{3}{256}z^4$ |
| | | $s_1(z) = \frac{3}{256}z^{-2} - \frac{11}{128}z^{-1} + \frac{1}{2} + \frac{11}{128}z - \frac{3}{256}z^2$ <br> $K_2 = \frac{1}{2}$ |

| $n$ | $\widetilde{n}$ | |
|---|---|---|
| 2 | | $\widetilde{g}(z) = \frac{1}{4}z^{-2} - \frac{1}{2}z^{-1} + \frac{1}{4}$ |
| | | $t_1(z) = -\frac{1}{2}z^{-1} - \frac{1}{2}$ <br> $K_1 = 1$ |
| | 2 | $\widetilde{h}(z) = -\frac{1}{8}z^{-2} + \frac{1}{4}z^{-1} + \frac{3}{4} + \frac{1}{4}z - \frac{1}{8}z^2$ |
| | | $s_1(z) = \frac{1}{4} + \frac{1}{4}z$ <br> $K_2 = -\frac{1}{2}$ |
| | 4 | $\widetilde{h}(z) = \frac{3}{128}z^{-4} - \frac{3}{64}z^{-3} - \frac{1}{8}z^{-2} + \frac{19}{64}z^{-1} + \frac{45}{64} + \frac{19}{64}z - \frac{1}{8}z^2 - \frac{3}{64}z^3 + \frac{3}{128}z^4$ |
| | | $s_1(z) = -\frac{3}{64}z^{-1} + \frac{19}{64} + \frac{19}{64}z - \frac{3}{64}z^2$ <br> $K_2 = -\frac{1}{2}$ |
| | 6 | $\widetilde{h}(z) = -\frac{5}{1024}z^{-6} + \frac{5}{512}z^{-5} + \frac{17}{512}z^{-4} - \frac{39}{512}z^{-3} - \frac{123}{1024}z^{-2} + \frac{81}{256}z^{-1} + \frac{175}{256}$ <br> $+ \frac{81}{256}z - \frac{123}{1024}z^2 - \frac{39}{512}z^3 + \frac{17}{512}z^4 + \frac{5}{512}z^5 - \frac{5}{1024}z^6$ |
| | | $s_1(z) = \frac{5}{512}z^{-2} - \frac{39}{512}z^{-1} + \frac{81}{256} + \frac{81}{256}z - \frac{39}{512}z^2 + \frac{5}{512}z^3$ <br> $K_2 = -\frac{1}{2}$ |

| $n$ | $\widetilde{n}$ | |
|---|---|---|
| 3 | | $\widetilde{g}(z) = \frac{-1}{8}z^{-2} + \frac{3}{8}z^{-1} - \frac{3}{8} + \frac{1}{8}z$ |
| | | $t_1(z) = 0$ <br> $s_1(z) = -3z$ <br> $t_2(z) = \frac{1}{8}z^{-2} + \frac{3}{8}z^{-1}$ <br> $K_1 = \frac{-1}{2}z^{-1}$ |
| | 1 | $\widetilde{h}(z) = \frac{-1}{4}z^{-2} + \frac{3}{4}z^{-1} + \frac{3}{4} - \frac{1}{4}z$ |
| | | $s_2(z) = -4z^2$ <br> $K_2 = -z$ |
| | 3 | $\widetilde{h}(z) = \frac{3}{64}z^{-4} - \frac{9}{64}z^{-3} - \frac{7}{64}z^{-2} + \frac{45}{64}z^{-1} + \frac{45}{64} - \frac{7}{64}z - \frac{9}{64}z^2 + \frac{3}{64}z^3$ |
| | | $s_2(z) = \frac{-3}{4}z - 4z^2 + \frac{3}{4}z^3$ <br> $K_2 = -z$ |
| | 5 | $\widetilde{h}(z) = \frac{-5}{512}z^{-6} + \frac{15}{512}z^{-5} + \frac{19}{512}z^{-4} - \frac{97}{512}z^{-3} - \frac{13}{256}z^{-2} + \frac{175}{256}z^{-1} + \frac{175}{256} - \frac{13}{256}z - \frac{97}{512}z^2 + \frac{19}{512}z^3 + \frac{15}{512}z^4 - \frac{5}{512}z^5$ |
| | | $s_2(z) = \frac{5}{32} - \frac{17}{16}z - 4z^2 + \frac{17}{16}z^3 - \frac{5}{32}z^4$ <br> $K_2 = -z$ |

| $n$ | $\widetilde{n}$ | |
|---|---|---|
| 4 | | $\widetilde{g}(z) = -\frac{1}{16}z^{-3}+\frac{1}{4}z^{-2}-\frac{3}{8}z^{-1}+\frac{1}{4}-\frac{1}{16}z$ |
| | | $t_1(z) = 0$ <br> $s_1(z) = -\frac{1}{4}-\frac{1}{4}z$ <br> $t_2(z) = -z^{-1}-1$ <br> $K_1 = 2$ |
| | 2 | $\widetilde{h}(z) = \frac{3}{32}z^{-3}-\frac{3}{8}z^{-2}+\frac{5}{32}z^{-1}+\frac{5}{4}+\frac{5}{32}z-\frac{3}{8}z^{2}+\frac{3}{32}z^{3}$ |
| | | $s_2(z) = \frac{3}{16}+\frac{3}{16}z$ <br> $K_2 = -\frac{1}{4}$ |
| | 4 | $\widetilde{h}(z) = -\frac{5}{256}z^{-5}+\frac{5}{64}z^{-4}-\frac{1}{256}z^{-3}-\frac{3}{8}z^{-2}+\frac{35}{128}z^{-1}+\frac{35}{32}+\frac{35}{128}z-\frac{3}{8}z^{2}-\frac{1}{256}z^{3}$ <br> $+\frac{5}{64}z^{4}-\frac{5}{256}z^{5}$ |
| | | $s_2(z) = -\frac{5}{128}z^{-1}+\frac{29}{128}+\frac{29}{128}z-\frac{5}{128}z^{2}$ <br> $K_2 = -\frac{1}{4}$ |
| | 6 | $\widetilde{h}(z) = \frac{35}{8192}z^{-7}-\frac{35}{2048}z^{-6}-\frac{55}{8192}z^{-5}+\frac{115}{1024}z^{-4}-\frac{557}{8192}z^{-3}-\frac{733}{2048}z^{-2}+\frac{2625}{8192}z^{-1}$ <br> $+\frac{525}{512}+\frac{2625}{8192}z-\frac{733}{2048}z^{2}-\frac{557}{8192}z^{3}+\frac{115}{1024}z^{4}-\frac{55}{8192}z^{5}-\frac{35}{2048}z^{6}+\frac{35}{8192}z^{7}$ |
| | | $s_2(z) = \frac{35}{4096}z^{-2}-\frac{265}{4096}z^{-1}+\frac{499}{2048}+\frac{499}{2048}z-\frac{265}{4096}z^{2}+\frac{35}{4096}z^{3}$ <br> $K_2 = -\frac{1}{4}$ |

| $n$ | $\tilde{n}$ | |
|---|---|---|
| 5 | | $\tilde{g}(z) = \frac{1}{32}z^{-3} - \frac{5}{32}z^{-2} + \frac{5}{16}z^{-1} - \frac{5}{16} + \frac{5}{32}z - \frac{1}{32}z^{2}$ |
| | | $t_1(z) = -5$<br>$s_1(z) = \frac{1}{40}z^{-1} + \frac{5}{24}$<br>$t_2(z) = \frac{75}{2} + \frac{45}{2}z$<br>$K_1 = 15$ |
| | 1 | $\tilde{h}(z) = \frac{3}{16}z^{-3} - \frac{15}{16}z^{-2} + \frac{5}{4}z^{-1} + \frac{5}{4} - \frac{15}{16}z + \frac{3}{16}z^{2}$ |
| | | $s_2(z) = \frac{-1}{75}$<br>$K_2 = \frac{1}{30}$ |
| | 3 | $\tilde{h}(z) = \frac{-5}{128}z^{-5} + \frac{25}{128}z^{-4} - \frac{13}{64}z^{-3} - \frac{35}{64}z^{-2} + \frac{35}{32}z^{-1} + \frac{35}{32} - \frac{35}{64}z - \frac{13}{64}z^{2} + \frac{25}{128}z^{3} - \frac{5}{128}z^{4}$ |
| | | $s_2(z) = \frac{-1}{360}z^{-1} - \frac{1}{75} + \frac{1}{360}z$<br>$K_2 = \frac{1}{30}$ |
| | 5 | $\tilde{h}(z) = \frac{35}{4096}z^{-7} - \frac{175}{4096}z^{-6} + \frac{15}{512}z^{-5} + \frac{25}{128}z^{-4} - \frac{1357}{4096}z^{-3} - \frac{1575}{4096}z^{-2} + \frac{525}{512}z^{-1} + \frac{525}{512} - \frac{1575}{4096}z - \frac{1357}{4096}z^{2} + \frac{25}{128}z^{3} + \frac{15}{512}z^{4} - \frac{175}{4096}z^{5} + \frac{35}{4096}z^{6}$ |
| | | $s_2(z) = \frac{7}{11520}z^{-2} - \frac{23}{5760}z^{-1} - \frac{1}{75} + \frac{23}{5760}z - \frac{7}{11520}z^{2}$<br>$K_2 = \frac{1}{30}$ |

| $n$ | $\widetilde{n}$ | |
|---|---|---|
| 6 | | $\widetilde{g}(z) = \frac{1}{64}z^{-4} - \frac{3}{32}z^{-3} + \frac{15}{64}z^{-2} - \frac{5}{16}z^{-1} + \frac{15}{64} - \frac{3}{32}z + \frac{1}{64}z^2$ |
| | | $t_1(z) = \frac{-1}{6}z^{-1} - \frac{1}{6}$ |
| | | $s_1(z) = \frac{-9}{16} - \frac{9}{16}z$ |
| | | $t_2(z) = \frac{-4}{3}z^{-1} - \frac{4}{3}$ |
| | | $K_1 = 4$ |
| | 2 | $\widetilde{h}(z) = \frac{-5}{64}z^{-4} + \frac{15}{32}z^{-3} - \frac{7}{8}z^{-2} - \frac{7}{32}z^{-1} + \frac{77}{32} - \frac{7}{32}z - \frac{7}{8}z^2 + \frac{15}{32}z^3 - \frac{5}{64}z^4$ |
| | | $s_2(z) = \frac{5}{32} + \frac{5}{32}z$ |
| | | $K_2 = \frac{-1}{8}$ |
| | 4 | $\widetilde{h}(z) = \frac{35}{2048}z^{-6} - \frac{105}{1024}z^{-5} + \frac{165}{1024}z^{-4} + \frac{235}{1024}z^{-3} - \frac{1827}{2048}z^{-2} + \frac{63}{512}z^{-1} + \frac{987}{512} + \frac{63}{512}z - \frac{1827}{2048}z^2 + \frac{235}{1024}z^3 + \frac{165}{1024}z^4 - \frac{105}{1024}z^5 + \frac{35}{2048}z^6$ |
| | | $s_2(z) = \frac{-35}{1024}z^{-1} + \frac{195}{1024} + \frac{195}{1024}z - \frac{35}{1024}z^2$ |
| | | $K_2 = \frac{-1}{8}$ |
| | 6 | $\widetilde{h}(z) = \frac{-63}{16384}z^{-8} + \frac{189}{8192}z^{-7} - \frac{119}{4096}z^{-6} - \frac{777}{8192}z^{-5} + \frac{1101}{4096}z^{-4} + \frac{557}{8192}z^{-3} - \frac{3465}{4096}z^{-2} + \frac{2079}{8192}z^{-1} + \frac{14091}{8192} + \frac{2079}{8192}z - \frac{3465}{4096}z^2 + \frac{557}{8192}z^3 + \frac{1101}{4096}z^4 - \frac{777}{8192}z^5 - \frac{119}{4096}z^6 + \frac{189}{8192}z^7 - \frac{63}{16384}z^8$ |
| | | $s_2(z) = \frac{63}{8192}z^{-2} - \frac{469}{8192}z^{-1} + \frac{843}{4096} + \frac{843}{4096}z - \frac{469}{8192}z^2 + \frac{63}{8192}z^3$ |
| | | $K_2 = \frac{-1}{8}$ |

# Appendix B

# A Simple Image Compression Example using WAILI

```
//
//      Simple image compression example
//

#include <waili/Image.h>

int main(void)
{
    const char infile[] = "image.pgm";
    const char outfile[] = "result.pgm";
    double threshold = 20.0;
    Image image;

    // Read the image
    image.Import(infile);

    // Transform the image using the Cohen-Daubechies-Feauveau
    // (2, 2) biorthogonal wavelets
    Wavelet *wavelet = Wavelet::CreateCDF(2, 2);
    TransformDescriptor transform[] = {
        { TT_ColsRows, wavelet },
        { TT_ColsRows, wavelet },
        { TT_ColsRows, wavelet },
        { TT_ColsRows, wavelet },
        { TT_ColsRows, wavelet },
        { TT_ColsRows, wavelet },
        { TT_ColsRows, wavelet },
        { TT_ColsRows, wavelet }
```

```
    };
    image.Fwt(transform, sizeof(transform)/sizeof(*transform));

    // Zero all entries smaller than the threshold
    for (u_int ch = 0; ch < image.GetChannels(); ch++)
        image[ch]->Threshold(threshold);

    // Inverse wavelet transform
    image.IFwt();

    // Write the reconstructed image to a file
    image.Export(outfile);
    return(0);
}
```

After the extension of WAILI to support very large images that are divided in blocks, the above example had to be changed:

```
//
//      Simple image compression example
//

#include <waili/Image.h>

int main(void)
{
    const char infile[] = "image.pgm";
    const char outfile[] = "result.pgm";
    double threshold = 20.0;
    BlockMgrM mgr;

    // Read the image
    Image *image = new Image(&mgr, infile);

    // Transform the image using the Cohen-Daubechies-Feauveau
    // (2, 2) biorthogonal wavelets
    Wavelet *wavelet = Wavelet::CreateCDF(2, 2);
    u8 id = wavelet->GetID();
    delete wavelet;
    TransformDescriptor transform[] = {
        { TT_ColsRows, id },
        { TT_ColsRows, id },
        { TT_ColsRows, id },
        { TT_ColsRows, id },
        { TT_ColsRows, id },
        { TT_ColsRows, id },
        { TT_ColsRows, id },
```

```
        { TT_ColsRows, id }
    };
    image->Fwt(transform, sizeof(transform)/sizeof(*transform));

    // Zero all entries smaller than the threshold
    for (u_int ch = 0; ch < image->GetChannels(); ch++)
        (*image)[ch]->Threshold(threshold);

    // Inverse wavelet transform
    image->IFwt();

    // Write the reconstructed image to a file
    image->Export(outfile);
    return 0;
}
```

The main difference between the two examples is the addition of the BlockMgr object when creating a new image. The *BlockMgrM* class manages blocks that are stored in main memory. The addition of the *GetID()* function is related to other evolutions of the internals of WAILI.

# Index


anisotropy, 79
APOD, *see* Approximate Proper Orthogonal Decomposition
Approximate Proper Orthogonal Decomposition, 96

best basis, 15
biorthogonal wavelets, *see* wavelets, biorthogonal
block manager, 72
blocks, 70
Brusselator, 104

CDF $(2,2)$, 22, 27, 32
chrominance, 40
coarse representation, 22
Cohen-Daubechies-Feauveau, 11, 22, 54, 87
color component, 40
communication, 71, 72, 76
complementary filter pair, 24
compression, 48, 74, 96
  lossless, 48
  lossy, 48, 65
Continuous Wavelet Transform, *see* wavelet transform, continuous
coordinates, 59
covariance matrix, 94
crop, 47, 58, 74
CWT, *see* wavelet transform, continuous

Daubechies wavelets, 9
DCT, *see* Discrete Cosine Transform
decomposition into lifting steps, 26
denoising, 41, 59, 73
dilation, 7
Discrete Cosine Transform, 48, 117
Discrete Wavelet Transform, *see* wavelet transform, discrete
dual filter pair, 11
dual lifting, 20, 21, 25
DWT, *see* wavelet transform, discrete

eigenmode, 94, 97, 108
  dual, 100, 103
  primal, 100, 103
eigenvalue, 94, 107
eigenvector, 94
embedded coder, 51
encoding, 49
energy, 17, 103, 107
entropy
  Shannon, *see* Shannon entropy
Euclidean algorithm, 26

Fast Fourier Transform, 12, 48
Fast Wavelet Transform, *see* wavelet transform, fast
father function, *see* wavelet function, father function
FFT, *see* Fast Fourier Transform
filter bank, 8, 22
filter coefficients, 8
  dyadic, 55
floating point number, 29, 62
FWT, *see* wavelet transform, fast

Galerkin projection, 117
GCV, *see* Generalized Cross Validation
Generalized Cross Validation, 43, 89
Geographical Information Systems, 69, 74
GIS, *see* Geographical Information Systems

Haar wavelet, 32
high-frequency information, 22
high-pass filter, 8, 22

image
  color, 40, 49
  grey scale, 39
in-place transform, 60, 61
integer wavelet transform, *see* wavelet transform, integer
interleaving, 60
inverse wavelet transform, *see* wavelet transform, inverse

Joint Photographic Experts Group, 50
JPEG, *see* Joint Photographic Experts Group

Karhunen-Loève Transform, 94
KLT, *see* Karhunen-Loève Transform

Laurent polynomial, 23
  degree of, 23
lazy wavelet transform, *see* wavelet transform, lazy
lifting scheme, 19
  integer, 84
linear interpolation, 21
localization, 6, 58, 70
long division, 26
low-dimensional model, 97
low-pass filter, 8, 22
luminance, 40

Mallat ordering, 60, 83
merge, 47, 58, 74
minimization, 43
mother function, *see* wavelet function, mother function
multi-resolution, 75
multi-resolution analysis, 6

noise reduction, 89
non-decimated wavelet transform, *see* wavelet transform, redundant
non-separable wavelets, 79
normalization, 33, 62

objects
  BlockMgr, 73
  Channel, 56
    LChannel, 56
    LChannelC, 56
    LChannelCR, 56
    LChannelR, 56
    NTChannel, 56
    NTChannelB, 73
    NTChannelR, 73
  Image, 56
  Lifting, 57
  Wavelet, 57
orthogonal wavelets, *see* wavelets, orthogonal
overlap regions, 71

pan, 75
parallelization, 71, 74, 75
Parseval's identity, 9, 49, 99
Partial Differential Equation, 103
PDE, *see* Partial Differential Equation
Peak Signal to Noise Ratio, 65
perfect reconstruction, 30
performance, 75
PIEFPAK, *see* WAILI
POD, *see* Approximate Proper Orthogonal Decomposition

polyphase matrix, 24
prediction, 20, 21, 80–82, 87
primal filter pair, 11
primal lifting, 20, 21, 25
Proper Orthogonal Decomposition, 94
PSNR, *see* Peak Signal to Noise Ratio

quantization, 49
quincunx lattice, 80
quotient, 26

reaction-diffusion system, 103
Red-Black wavelet transform, 80, *see* wavelet transform, Red-Black
redundant wavelet transform, *see* wavelet transform, redundant
regular rectangular grid, 80
remainder, 26
resolution, 39
Riesz basis, 10, 49
rounding, 30, 83

scaling, 44, 59
    down, 44
    up, 44
scaling function, 7
second-generation wavelets, 19
separable, 79
Shannon entropy, 17, 65
shift, 63
signal extension, 33
    lifting, 36
    periodic, 33
    symmetric, 34, 36, 83
snapshots, 94, 104, 107
splitting, 20, 21, 80–82
stationary wavelet transform, *see* wavelet transform, redundant
subimage, 59, 75
subsampling, 23

tensor product, 12, 79, 87
threshold
    optimal, 42
    universal, 43
thresholding, 41
    hard-thresholding, 41, 65
    soft-thresholding, 41
tiling, 70
timings, 75, 111
translation, 7
translation invariance, 13
two-dimensional, 12

update, 20, 21, 80–82, 87

vanishing moments, 6

WAILI, 53, 70
    demo, 64
    example, 137
    functionality, 64
wavelet decomposition tree, 11
wavelet function, 8
    father function, 7
    mother function, 8
Wavelet Packet Transform, 14, 97
wavelet transform, 23
    continuous, 1
    discrete, 1, 22
    fast, 8, 117
    integer, 29, 83
    inverse, 8, 21
    lazy, 20, 21, 24, 71
    Red-Black, 80
    redundant, 13, 87
wavelets
    biorthogonal, 10
    orthogonal, 9
WPT, *see* Wavelet Packet Transform

$z$-transform, 23, 30
zerotree encoding, 50, 65, 74
zoom, 75