Hypercomplex Scaling and Wavelet Filters: Their Discovery and Their Application to Colour Vector Image Processing

Peter Fletcher, BSc Lanc, BA BSc MMath Open, MSc Sheff, FCA, AMIMA

A thesis submitted for the degree of

Doctor of Philosophy

School of Computer Science and Electronic Engineering

University of Essex

October 2018

Abstract

The aim of this thesis is to extend existing work and find new discrete scaling and wavelet filters with quaternion coefficients and the first with Clifford $Cl(1,1)$ and $Cl(2,0)$ coefficients; and then to demonstrate the use of these filters by finding hypercomplex wavelet transforms of colour vector images. We solve certain symbolic matrix equations simultaneously to find our scaling filter coefficients and then use a numerical method involving paraunitary completion of the polyphase matrix to find related wavelet filter coefficients. We find that our symbolic solutions include full and 'partial' transposes of each other. Now complex numbers are isomorphic to each of the three two-dimensional subalgebras of the quaternions and some two-dimensional subalgebras of $Cl(1,1)$ and $Cl(2,0)$, $Cl(1,0)$ being isomorphic to the rest: thus, we may use the values of the coefficients from complex and $Cl(1,0)$ scaling and wavelet filters in the appropriate places of further quaternion, $Cl(1,1)$ and $Cl(2,0)$ scaling and wavelet filters. We use the cascade algorithm on all our filters and illustrate the resulting hypercomplex scaling and wavelet functions with plots of all possible projections onto two and three dimensions. We then use our filters to find hypercomplex wavelet transforms of some colour test images represented as arrays of pure hypercomplex numbers, ones with no scalars. To do this, for each one we place copies of one pair of filters down the leading diagonal of a zero matrix to produce a banded matrix. We pre-multiply a colour vector test image by this banded matrix and post-multiply by its conjugate transpose. This results in an array of full hypercomplex numbers. We then extract the approximation plus horizontal, vertical and diagonal detail images from the scalar (black and white) and vector (colour) parts of the result separately and illustrate them side by side, each arranged in the conventional format.

.

Contents

List of Figures

List of Tables

Acknowledgements

I should like to express my gratitude to my supervisor, Dr Stephen Sangwine, for his constant encouragement and enthusiasm over the past four years. I know that I would not have got this far if I had had a different supervisor.

My thanks also to Dr Jane Pearson of the History Department for mentoring me since 2015, when I was diagnosed with Asperger's Syndrome. I shall miss our weekly chats.

There is a large body of work, without which this PhD would not have been possible. I am referring to the Quaternion Toolbox for MATLAB® and the Clifford Multivector Toolbox for MATLAB®. Both were written by Stephen Sangwine, the former with Nicolas Le Bihan at Le laboratoire GIPSA-lab in Grenoble, France, and the latter with Eckhard Hitzer at the International Christian University in Tokyo.

Declaration

The work in this thesis is based on research carried out in the School of Computer Science and Electronic Engineering at the University of Essex in the United Kingdom. I certify that this is all my own work, except where stated in the text, and that no part of this thesis has been submitted elsewhere for any other degree or qualification.

List of Publications

P. Fletcher and S. J. Sangwine. The development of the quaternion wavelet transform. *Signal Processing*, 136:2-15, 2017. https://doi.org/10.1016/j.sigpro.2016.12.025

The purpose of this article is to review what has been written on what other authors have called quaternion wavelet transforms (QWTs): there is no consensus about what these should look like and what their properties should be. We briefly explain what real continuous and discrete wavelet transforms and multiresolution analysis are and why complex wavelet transforms were introduced; we then go on to detail published approaches to QWTs and to analyse them. We conclude with our own analysis of what it is that should define a QWT as being truly quaternionic and why all but a few of the 'QWTs' we have described do not fit our definition.

P. Fletcher. Quaternion wavelet transforms of colour vector images. In *Proceedings of the 9th Computer Science and Electronic Engineering Conference (CEEC 2017), held in Wivenhoe, UK*, pages 168–171. IEEE, 2017. https://doi.org/10.1109/CEEC.2017.8101619

The most common existing quaternion wavelet transforms (QWTs) do not make use of any quaternion properties and can only be used to analyse monochrome images. However, one recent QWT has been found in the form of a pair of filters with quaternion coefficients. Our aim is to extend this work by finding more such QWTs. We solve equations that the filter coefficients must satisfy using algebraic and numerical methods and find a new QWT. We plot the functions associated with the pre-existing QWT and our new one and then use the two pairs of filters on the same colour vector image. This is the first time that this form of QWT has been demonstrated. Finally, we consider how we can extend our work and find more QWTs.

P. Fletcher. Discrete Wavelets with Quaternion and Clifford Coefficients. *Advances in Applied Clifford Algebras*, 25(3):59, 2018. http://dx.doi.org/10.1007/s00006-018-0876-5

In this paper we use matrix representations of quaternions and Clifford algebras and solve the same matrix equations in each case to find Daubechies quaternion and Clifford scaling filters. We use paraunitary completion of the polyphase matrix to find corresponding quaternion and Clifford wavelet filters. We then use the cascade algorithm on our filters to find quaternion and Clifford scaling and wavelet functions, which we illustrate using all possible projections onto two and three dimensions: to our knowledge, this is the first time that this has been done. We discuss the shapes of these functions and conclude with a consideration of what we could actually do with our filters.

P. Fletcher and S. J. Sangwine. Hypercomplex Wavelet Transforms of Colour Vector Images. In development and will be submitted to *IEEE Transactions on Image Processing*.

Discrete real-valued scaling and wavelet filters have existed for some years as a tool for signal and image processing. They have been generalised to complex- and matrix-valued filters and others but, as we found in our review of the literature, only one particular scaling filter and its associated wavelet filter can be said to be truly quaternion-valued. The method to find this quaternion-valued scaling filter was to represent quaternions as 4×4 real matrices and then to solve certain matrix equations simultaneously. Paraunitary completion of the polyphase matrix then gave the quaternion-valued wavelet filter. We extended the method to find other quaternion-valued scaling and wavelet filters, as well as Clifford-valued ones, and presented our results in a previous article. The purpose of this article is to demonstrate the use of our filters by finding hypercomplex wavelet transforms of colour vector images. We arrange our filters in a certain pattern down the leading diagonal of square matrices and represent a colour image as an array of either pure quaternions or the vector/bivector part of a four-dimensional Clifford algebra. We then pre-multiply our array by our filter matrix and post-multiply by the Hermitian conjugate of our filter matrix. This product forms the two-dimensional wavelet transform. We extract from this product, an approximation to our original image plus horizontal, vertical and diagonal detail, and display these. We do the same with the scalar part of the wavelet transform. This is the first time that this form of image analysis has been presented.

P. Fletcher. The Left and Right Inverses of a 2×2 Matrix of Octonions. In development and will be submitted to *Advances in Applied Clifford Algebras.*
A numerical algorithm exists for inverting square matrices of octonions. In practice, it does not give very accurate results due to its requiring a large number of matrix multiplications: these magnify small errors. We provide a new numerical algorithm for the 2×2 case that gives much improved accuracy and use MATLAB® to show this by comparing the results of the old algorithm with those of our new one.

Chapter 1

Introduction

This thesis is concerned with the theory and numerical implementation of discrete hypercomplex wavelets, *i.e.* wavelet filters with hypercomplex coefficients, applied to the processing of digital images, in particular, colour images. These require three components per pixel: each pixel may be represented as a four dimensional hypercomplex number with the coefficients of the three imaginary parts being the values of red, green and blue and the coefficient of the scalar as zero.

We first introduce the origin and concept of hypercomplex numbers, which are an extension of the complex numbers to 2^n dimensions where $n \geq 1$ (a complex number being a particular hypercomplex number), and we then introduce the concept of wavelets as localised basis functions for representing a signal or image. We list our main contributions and finish this chapter by giving the structure of the rest of the thesis.

1.1 Background

We begin by introducing the first component of this thesis.

Rotations of points in the plane about some origin may be performed by multiplying certain 2×2 matrices by the co-ordinates of the points represented as length two column vectors. It is well known that an alternative way of doing the same rotations is to write the set of points, together with the rotations, as complex numbers. Multiplication of 2×2 rotation matrices then becomes multiplication of complex numbers.

In the same way, rotations of points in three dimensions about some axis may be performed by multiplying certain 3×3 matrices by the co-ordinates of the points represented as length three column vectors. The three basic 3×3 rotation matrices actually have 2×2 rotation matrices embedded. Each one is designed so as to rotate points in all planes parallel to one of: the x-y, x-z and y-z planes, *i.e.* around the z, y or x axis respectively. Multiplying these matrices together then gives a compound rotation in parallel planes perpendicular to, in general, a fourth axis at some angle $\neq \pi/2$ to each of the other three. The quaternions were the first hypercomplex algebra and were introduced specifically to be the analogue of complex numbers in the representation of rotations, but in three dimensions instead of two: multiplication of 3×3 rotation matrices then becomes multiplication of quaternions. Of course, complex numbers have many other applications and quaternions have found many other uses as well. The quaternions are actually a four-dimensional algebra; no three-dimensional algebra exists that can be used to model rotations in three dimensions.

The Clifford algebras are a collection of hypercomplex algebras. The simplest way of thinking of them is as a generalisation of real and complex numbers and quaternions, these three algebras becoming, or rather being isomorphic to, particular Clifford algebras. Each lower dimensional Clifford algebra is isomorphic to several subsets of the higher dimensional ones.

We now introduce the second component of this thesis.

Wavelets, literally 'little waves', localised periodic disturbances, were originally developed to analyse seismic waves. They were later developed further and introduced to the wider signal processing community. They have found many applications, *e.g.* data and image compression (the JPEG 2000 standard uses wavelets), solving partial differential equations, transient detection, pattern recognition, texture analysis, noise reduction and others. In the same way that sines and cosines are the usual basis functions for the Fourier transform, wavelets are the basis functions for the wavelet transform: a Fourier transform decomposes a signal into its component frequencies and a wavelet transform does likewise as the wavelets are stretched and compressed; but thanks to the localisation of the wavelets, the wavelet transform also encodes when or where those frequencies occurred. Wavelet transforms exist in both continuous and discrete forms but we shall only be interested in the latter. Briefly, a wavelet filter extracts the high frequency part of a signal at a certain resolution and a scaling filter extracts the low frequency part; the latter may then be analysed at a finer resolution by being put through further wavelet and scaling filters; and so on. This is multiresolution analysis.

Discrete wavelets have been generalised to complex, analytic, monogenic, matrix-valued and so-called quaternion wavelets. We have added 'so-called' to the last of these because we would argue that, since apparently all but one, or maybe two, of the practical quaternion wavelets that have been developed so far are actually the basis functions for two-dimensional short-time Fourier transforms, these should be called something else. One exception, which was certainly the only one when we began research for this thesis, was presented in chapter five of Paul Ginzberg's PhD thesis [61, p. 169] (repeated in condensed form slightly earlier in [62]). This was the first one to truly marry quaternions to wavelets. However, Ginzberg did not do anything with his filters, a single quaternion scaling filter and a single quaternion wavelet filter: this omission provided the main motivation for this thesis.

1.2 Contributions

The contributions of this thesis are:

1. extensions of Ginzberg's work to find more quaternion and the first Clifford scaling and wavelet filters;
2. a way of visualising complex and hypercomplex scaling and wavelet functions by multiple projections onto two and three dimensions;
3. the first ever 'true' quaternion and Clifford wavelet transforms of colour vector images;
4. re-use of the coefficients of existing complex-valued scaling and wavelet filters in certain quaternion and Clifford scaling and wavelet filters.

We define colour vector images fully in Section 2.6.

1.3 Structure of this Thesis

In Chapter 2, we expand on what we said in Section 1.1 about hypercomplex algebras, giving a short history of them, full definitions and an example of colour vector image processing not involving wavelets. In Chapter 3, we expand on what we said in Section 1.1 about real wavelets. Chapter 4 is based on slightly expanded versions of Sections 4 and 5 of our review article [44], wherein we review the literature on quaternion wavelet transforms. Chapter 5 is taken from our article on the subject of hypercomplex scaling and wavelet filters [43]. We explain how Ginzberg found his quaternion scaling and wavelet filters and adapt his method to find more quaternion and the first Clifford scaling and wavelet filters. We list all the filter coefficients we found and find and illustrate all the corresponding scaling and wavelet functions. In Chapter 6 we use most of our filters, as well as hypercomplex filters using complex filter coefficients, to find hypercomplex wavelet transforms of a few colour vector images. This chapter will form the basis of a future article on the subject. In Chapter 7 we present our conclusions and consider how our work could be developed further.

Chapter 2

Hypercomplex Algebras and Colour Vector Images

In this chapter we introduce quaternions and Clifford algebras, which we use in the rest of the thesis. In Section 2.1, we give a brief history of complex numbers and quaternions; the quaternions were the first hypercomplex numbers and in Section 2.2 we give their main properties. In Section 2.3 we give a brief history of Clifford algebras and in Section 2.4 we give their main properties. In Section 2.5 we give matrix representations of complex numbers, quaternions and Clifford algebras up to four dimensions. In Section 2.6 we introduce representations of colour pixels in hypercomplex algebras. We mention quaternion Fourier transforms in Section 2.7 and give an example of colour vector image processing not involving wavelets in Section 2.8. We give a summary of the chapter in Section 2.9.

2.1 A Brief History of Complex Numbers and Quaternions

We start by briefly outlining the history of the original sources of complex numbers, namely solutions of cubic and, only later, quadratic equations. We then discuss how complex numbers were discovered and gradually came to be accepted. Finally, we explain how hypercomplex numbers came from generalisations of complex numbers and not from solutions of particular equations. Until the introduction of abbreviations for 'things' (*i.e.* independent variables), squares, cubes, roots, *etc.*, everything had to be written out in words.

The earliest quadratic equation for which we have evidence was solved by an Ancient Egyptian sometime between 1900BC and 1800BC. This sole example is on Berlin Papyrus 6619 [45, §2.3a, p. 81]. In modern notation, the problem is $x^2 + \left(\frac{3}{4}x\right)^2 = 100$. The Babylonians, strictly the Mesopotamians (Babylon was a city in Mesopotamia), co-existed with the Egyptians but used clay tablets instead of papyri to record their writings and mathematical problems. Most tablets that have been excavated date from the time of Hammurabi (*c.* 1792BC to 1750BC) [157]. The mathematical ones, five hundred or so out of a total of several hundred thousand, fall broadly into two categories, table texts and problem texts [40, §1.4]. The table texts list, *e.g.*, squares, reciprocals, cubes, cube roots and sums of squares and cubes. The Babylonians' quadratics arose from geometry and in modern notation may generally be written as the simultaneous equations $x - y = b$ and $x^2 + y^2 = c$. They could also solve a limited number of cubics, ones of the form $x^3 + x^2 = a$ in modern notation. In all cases, solutions were found by reference to the tables. The Greeks found a way to solve quadratics of the form $a(x - b) = x^2$, examples of which also appeared on some of the Babylonian problem texts. They also studied cubics and some higher powers, but left no general methods of solution. In all cases, only positive roots were sought or considered in any way real.

Brahmagupta's *Brahmasphutasiddhunta* of *c.* AD628 included, for the first time, a general method to solve all quadratic equations, the one used today, and also for the first time, negative roots of quadratics were accepted as real solutions. Those whose roots involved the square roots of negative numbers were still discarded as insoluble.

During the next 800 years, solutions to some special cases of cubics were found and Umar ibn Ibrahim al-Khayyami (1048–1131) proved that any third-order equation can be solved geometrically by finding the intersection of two conic sections [67, p. 9], but he could not translate his solution into words.

In sixteenth century Italy, academic appointments were temporary, often for only one year at a time [42, p. 24]. Professorships could be won and lost at public problem-solving contests, so if a professor or an aspiring professor found a way to solve forms of equations that had not been solved before, he kept them secret until such time as they felt it right to publish. Sometime before 1515, Scipione del Ferro (1465–1526) found the general solution of $x^3 + cx = d$: one solution is given by

$$x = \sqrt[3]{\sqrt{\left(\frac{c}{3}\right)^3 + \left(\frac{d}{2}\right)^2} + \frac{d}{2}} - \sqrt[3]{\sqrt{\left(\frac{c}{3}\right)^3 + \left(\frac{d}{2}\right)^2} - \frac{d}{2}} \tag{2.1}$$

and factorisation then reduces the problem to solving a quadratic. This cubic is now known as the depressed cubic and all other cubics may be transformed into this form by certain substitutions, but that was not known at the time. Before his death, he passed his method onto his student Antonio Maria del Fiore and his son-in-law Annibale della Nave. The latter became his successor as professor at Bologna. With del Ferro's death, del Fiore felt able to exploit del Ferro's solution and when Niccolò Tartaglia (1499–1557), a Venetian mathematician, engineer, surveyor and bookkeeper, announced he could solve $x^3 + ax^2 = b$, del Fiore challenged him to a public contest, the loser to pay for 30 dinners for the winner and his friends. Two months before the day of the contest in 1535, each gave the other 30 problems to solve. Del Fiore was not a good mathematician and could only solve equations of the form $x^3 + cx = d$; all of the problems he gave to Tartaglia were of this form. Tartaglia independently rediscovered del Ferro's method and del Fiore could not solve the more general mathematical problems that Tartaglia had set: Tartaglia won.

Gerolamo Cardano (1501–1576) found fame as a physician, but was also interested in physics, mathematics and gambling. He heard of the contest between del Fiore and Tartaglia and in return for introducing Tartaglia to the Spanish Governor of Milan, a possible source of finance for Tartaglia's research in military science, Tartaglia revealed his method of solving $x^3 + cx = d$, conditional on Cardano's promising not to show it to anyone else. When Cardano and his secretary and pupil, Ludovico Ferrari (1522–1565), prepared to publish a major work on algebra, they were going to omit the solution of $x^3 + cx = d$ because of Cardano's promise. However, della Nave showed del Ferro's papers to them. Cardno felt able to forget his promise to Tartaglia and included the solution. The *Ars Magna* appeared in 1545; this deals with 13 separate cases in great detail, the depressed cubic being one, as well as quartics. The *Ars Magna* was the most comprehensive text on algebra for the next century. However, Equation (2.1) can give some, to Cardano and his contemporaries, strange results. For example, given $x^3 = 15x + 4$, so that $c = -15$ and $d = 4$ (this would have been a special case separate from Equation (2.1), but we get the same result), we find $x = \sqrt[3]{2 + \sqrt{-121}} + \sqrt[3]{2 - \sqrt{-121}}$. Cardano could not make sense of this. We can recognise the complex numbers (this name for them came later) and this is actually equal to $2 + \mathrm{i} + 2 - \mathrm{i} = 4$. Factorising using this root then gives a quadratic, which is easily solved to give $2 \pm \sqrt{3}$; thus, complex numbers have appeared in an intermediate step, but the three roots are in fact all real. The *Ars Magna* did however contain the first ever calculation with complex numbers: $(5+\sqrt{-15}) \times (5-\sqrt{-15}) = 25-(-15) = 40$. No previous mathematician would have contemplated doing anything involving the square root of a negative number. *L'Algebra* of 1572, by Rafael Bombelli (1526–1572), gave a thorough account of all algebra then known and was the first book to give complete and correct rules for the arithmetic manipulation of complex numbers [80, p. 222].

Mathematicians gradually came to accept that $\sqrt{-1}$ had its uses in the solution of some polynomials, but it was still regarded as something rather strange; René Descartes (1596–1650) described the square roots of negative numbers as 'imaginary' in 1637 and wrote "For any equation one can imagine as many roots (as its degree would suggest), but in many cases no quantity exists which corresponds to what one imagines" [13], so it would appear that around this time, mathematicians were close to accepting the role of complex numbers in the roots of those quadratic equations not having real roots. A few mathematicians eventually started to study complex numbers in their own right, without reference to polynomials, and Caspar Wessel (1745–1818) gave $\sqrt{-1}$ a geometric interpretation in 1799. His paper was generally unknown until 1895, but in the meantime, Jean-Robert Argand (1768–1822) had introduced his diagrams of the complex plane in 1806 and Carl Friedrich Gauss (1777–1855) had proposed using the symbol i ('imaginary') to represent $\sqrt{-1}$ in 1831, defining $\mathrm{i}^2 = -1$ [116]. He also coined the term 'complex number' in 1832 for numbers like $a + b\mathrm{i}$.

Sir William Rowan Hamilton (1805–1865), who became Andrews Professor of Astronomy at the University of Dublin and Royal Astronomer of Ireland, both in 1827 at the age of 22, was the first to consider complex numbers as ordered pairs of real numbers or 'algebraic couples', as he called them in 1833, with special rules of multiplication [70]: $(a, b) \cdot (c, d) = (ac - bd, ad + bc)$. He spent many years from around 1835 trying unsuccessfully to generalise his ordered pairs to ordered triplets, the aim being to allow spatial rotations to be represented algebraically, just as planar ones can with complex numbers. He could add triplets, but he could not devise a way of multiplying them to give another triplet. It became an obsession and it reached the stage where his children each morning would enquire, "Well, Papa can you multiply triplets?" to which the answer was always no. Then on Monday 16th October 1843, Hamilton and his wife were walking along the Royal Canal in Dublin when, in his words, "...there dawned on me the notion that we must admit, in some sense, a fourth dimension of space for the purpose of calculating with triples [*sic*]. [...] An electric circuit seemed to close, and a spark flashed forth" [115]. He could not resist carving Equation (2.2) into the stone of Broome Bridge as they passed it and in 1958, the Royal Irish Academy unveiled a plaque on the bridge to commemorate this act of vandalism. Hamilton named his new objects quaternions from the Latin *quaternio*, meaning a set of four.

Apart from the pure mathematical interest, they came to be used widely in physics, *e.g.*, James Clerk Maxwell (1831–1879) used them when developing his eponymous equations. Vectors started to replace quaternions from the 1880s onwards and interest in quaternions waned: vectors are conceptually easier and their notation is clearer. However, in the twentieth century quaternions found new uses in physics, for example in: quantum mechanics, relativity, string theory, super-symmetry and quantum gravity [38, p. 1]. In computer animation, quaternions are used for rotations because they take up less space in memory than would the equivalent rotation matrices.

Nowadays the algebra of quaternions is denoted $\mathbb{H}$ in Hamilton's honour. In modern terminology, quaternions are hypercomplex numbers and they form a four-dimensional, non-commutative, normed division algebra over the real numbers.

2.2 Properties of Quaternions

These definitions can be found in a number of different places, *e.g.*, Ell *et al.* [37, pp. 1-6] or Ginzberg [61, pp. 21-23].

Definition 2.2.1. The Cartesian form of a quaternion $\mathbf{q} \in \mathbb{H}$ is given by

$$\mathbf{q} = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k},$$

where $a, b, c, d \in \mathbb{R}$ are its *components* and

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1, \tag{2.2}$$
$$\mathbf{ij} = -\mathbf{ji} = \mathbf{k}, \quad \mathbf{jk} = -\mathbf{kj} = \mathbf{i}, \quad \mathbf{ki} = -\mathbf{ik} = \mathbf{j}.$$

Remark 2.2.2. If we had $a, b, c, d \in \mathbb{C}$, then $\mathbf{q}$ would be a *biquaternion.*

Definition 2.2.3. The conjugate of $\mathbf{q}$ is

$$\overline{\mathbf{q}} = a - b\mathbf{i} - c\mathbf{j} - d\mathbf{k}.$$

Definition 2.2.4. The modulus of $\mathbf{q}$ is

$$|\mathbf{q}| = \sqrt{\mathbf{q}\overline{\mathbf{q}}} = \sqrt{a^2 + b^2 + c^2 + d^2}.$$

Remark 2.2.5. Different authors have different definitions of the norm $\|\cdot\|$: some would write $\|\mathbf{q}\| = |\mathbf{q}|^2$ and others would write $\|\mathbf{q}\| = |\mathbf{q}|$[1]. To avoid any ambiguity, we shall only refer to the modulus or modulus squared as appropriate.

Definition 2.2.6. Provided $\mathbf{q} \neq 0$, the inverse of $\mathbf{q}$ is

$$\mathbf{q}^{-1} = \frac{\overline{\mathbf{q}}}{|\mathbf{q}|^2},$$

[1] This is actually the definition of the 2-norm.

Definition 2.2.7. The real and imaginary parts of $\mathbf{q}$ are

$$\Re(\mathbf{q}) = a \qquad \Im_{\mathbf{i}}(\mathbf{q}) = b, \qquad \Im_{\mathbf{j}}(\mathbf{q}) = c, \qquad \Im_{\mathbf{k}}(\mathbf{q}) = d.$$

Definition 2.2.8. The scalar and vector parts of $\mathbf{q}$ are respectively

$$\mathrm{S}(\mathbf{q}) = \Re(\mathbf{q}) \qquad \text{and} \qquad \mathbf{V}(\mathbf{q}) = b\mathbf{i} + c\mathbf{j} + d\mathbf{k}.$$

Remark 2.2.9. The conjugate can therefore be written $\overline{\mathbf{q}} = \mathrm{S}(\mathbf{q}) - \mathbf{V}(\mathbf{q})$.

Definition 2.2.10. The set of pure quaternions is

$$\mathbb{P} = \left\{\mathbf{q} \in \mathbb{H} : \mathbf{q} = \mathbf{V}(\mathbf{q})\right\}.$$

Definition 2.2.11. The set of unit quaternions is

$$\mathbb{S} = \left\{\mathbf{q} \in \mathbb{H} : |\mathbf{q}| = 1\right\}.$$

Remark 2.2.12. If $\boldsymbol{\mu} \in \mathbb{P} \cap \mathbb{S}$, then $\boldsymbol{\mu}^2 = -1$. In the representation of a quaternion as

$$\mathbf{q} = \mathrm{S}(\mathbf{q}) + \mathbf{V}(\mathbf{q}) = \mathrm{S}(\mathbf{q}) + |\mathbf{V}(\mathbf{q})|\boldsymbol{\mu},$$

$\boldsymbol{\mu}$ is usually referred to as the *axis*. Note that for $\boldsymbol{\mu} \in \mathbb{P} \cap \mathbb{S}$, it is true that $\boldsymbol{\mu} = -\boldsymbol{\mu}^{-1}$.

Definition 2.2.13. An involution $f : \mathbb{H} \to \mathbb{H}$ is such that for all $\mathbf{q}, \mathbf{p} \in \mathbb{H}$,

$$\begin{aligned} f(f(\mathbf{q})) &= \mathbf{q}, \\ f(\mathbf{p} + \mathbf{q}) &= f(\mathbf{p}) + f(\mathbf{q}), \\ f(\mathbf{pq}) &= f(\mathbf{p})f(\mathbf{q}). \end{aligned}$$

Definition 2.2.14. Given a quaternion $\mathbf{q}$, its three canonical involutions are

$$\overline{\mathbf{q}}^{\mathbf{i}} = a + b\mathbf{i} - c\mathbf{j} - d\mathbf{k} = -\mathbf{iqi},$$

$$\overline{\mathbf{q}}^{\mathbf{j}} = a - b\mathbf{i} + c\mathbf{j} - d\mathbf{k} = -\mathbf{jqj},$$

$$\overline{\mathbf{q}}^{\mathbf{k}} = a - b\mathbf{i} - c\mathbf{j} + d\mathbf{k} = -\mathbf{kqk}.$$

Remark 2.2.15. The most general definition of involution is

$$\overline{\mathbf{q}}^{\boldsymbol{\mu}} = -\boldsymbol{\mu}\mathbf{q}\boldsymbol{\mu},$$

where $\boldsymbol{\mu} \in \mathbb{P} \cap \mathbb{S}$.

Remark 2.2.16. Note that in Definition 2.2.14, we effectively find ‘partial conjugates’, negating two imaginary parts at a time. We can also just negate one imaginary part at a time.

Definition 2.2.17. The partial conjugates of a quaternion, negating one imaginary part at a time, are

$$-\text{conj}(\mathbf{iqi}) = a - b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$$

$$-\text{conj}(\mathbf{jqj}) = a + b\mathbf{i} - c\mathbf{j} + d\mathbf{k}$$

$$-\text{conj}(\mathbf{kqk}) = a + b\mathbf{i} + c\mathbf{j} - d\mathbf{k}.$$

Definition 2.2.18. The exponential of a pure quaternion $\boldsymbol{p} \in \mathbb{P}$ is

$$\mathrm{e}^{\boldsymbol{p}} = \cos|\boldsymbol{p}| + \frac{\boldsymbol{p}}{|\boldsymbol{p}|}\sin|\boldsymbol{p}|.$$

Definition 2.2.19. The exponential of a full quaternion $\boldsymbol{q} \in \mathbb{H}$ is, using Remark 2.2.12,

$$\mathrm{e}^{\boldsymbol{q}} = \mathrm{e}^{\mathrm{S}(\boldsymbol{q})}\mathrm{e}^{\mathbf{V}(\mathbf{q})},$$

where $\mathrm{e}^{\mathbf{V}(\mathbf{q})}$ is given by Definition 2.2.18.

Remark 2.2.20. Note that with pure quaternions $\boldsymbol{p}, \boldsymbol{r} \in \mathbb{P}$, in general

$$\mathrm{e}^{\boldsymbol{p}}\mathrm{e}^{\boldsymbol{r}} \neq \mathrm{e}^{\boldsymbol{p}+\boldsymbol{r}}$$

unless $\boldsymbol{p} = \boldsymbol{r}$.

Remark 2.2.21. Note that with full quaternions $\boldsymbol{q}, \boldsymbol{s} \in \mathbb{H}$, in general

$$\mathrm{e}^{\boldsymbol{q}}\mathrm{e}^{\boldsymbol{s}} \neq \mathrm{e}^{\boldsymbol{q}+\boldsymbol{s}}$$

unless $\boldsymbol{V}(\boldsymbol{q}) = \boldsymbol{V}(\boldsymbol{s})$.

2.3 A Brief History of Clifford Algebras

Quaternions were the first hypercomplex numbers, but they were not the last.

In 1844, Hermann Grassmann (1809–1877) published his *Die lineale Ausdehnungslehre* or *The Theory of Linear Extension*; this was re-released in revised form in 1862 [65]. What Grassmann did was to invent linear algebra [41]: he defined free linear spaces, subspaces, independence, span, dimension, *etc.* In an article of 1855 [64], which he incorporated into the second edition of *Die lineale Ausdehnungslehre*, he introduced the exterior or outer product and from that derived the inner product. In modern terminology, the exterior product of two vectors $\mathbf{u}$ and $\mathbf{v}$ is called the wedge product and is written $\mathbf{u} \wedge \mathbf{v}$ and the well-known inner product of modern vector analysis, also called the scalar product or dot product, is written $\mathbf{u} \cdot \mathbf{v}$. Grassmann called $\mathbf{u}$ and $\mathbf{v}$, 'extensive quantities'. In three dimensions, $|\mathbf{u} \wedge \mathbf{v}| = |\mathbf{u} \times \mathbf{v}|$, but $\mathbf{u} \wedge \mathbf{v}$ is a directed area called a bivector, 2-vector or 2-blade, and $\mathbf{u} \times \mathbf{v}$ is just another vector perpendicular to $\mathbf{u}$ and $\mathbf{v}$. In the same way that $\mathbf{u} \times \mathbf{v} = -\mathbf{v} \times \mathbf{u}$ and $\mathbf{u} \times \mathbf{u} = 0$, the wedge product satisfies $\mathbf{u} \wedge \mathbf{v} = -\mathbf{v} \wedge \mathbf{u}$ and $\mathbf{u} \wedge \mathbf{u} = 0$, but the wedge product is more general: $\mathbf{u} \wedge \mathbf{v} \wedge \mathbf{w}$ is well defined and is, assuming $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ are linearly independent of each other, the directed volume of the parallelepiped formed by the three vectors. This is called a trivector, 3-vector or 3-blade and the modulus is the normal volume. The cross product of three vectors is undefined, unless we write either $(\mathbf{u} \times \mathbf{v}) \times \mathbf{w}$ or $\mathbf{u} \times (\mathbf{v} \times \mathbf{w})$, and both of these are zero if the three vectors are mutually perpendicular or if the pair in the brackets are linearly dependent; otherwise we just have another vector. Unlike the cross product, the wedge product is also well defined in space of dimension greater than three.

We now come to the contribution of William Kingdon Clifford (1845–1879).

Relevant details of an unfinished article by Clifford [27] of 1876 are given in [33, §6] (published in 1882). Clifford uses Grassmann's general definition of a product: "A product of two extensive quantities $\mathbf{u}$ and $\mathbf{v}$ is either defined as an extensive quantity or a scalar." He considers n basis elements $\iota_1, \iota_2, \ldots, \iota_n$ that each satisfy $\iota_j^2 = \pm 1$, a scalar using Grassmann's inner product, and where $\iota_i \iota_j = -\iota_j \iota_i$, an extensive quantity using Grassmann's outer product, provided $i \neq j$. All possible products must either be of odd order, (products of $2^{2m-1}, m \geq 1$, distinct basis elements) or of even order (products of $2^{2m}, m \geq 1$, distinct basis elements, or of none, with $m = 0$). With n basis elements, there is ${}^nC_0 = 1$ term of order 0, ${}^nC_1 = n$ of order 1, ${}^nC_2 = \frac{1}{2}n(n-1)$ of order 2, …, ${}^nC_n = 1$ of order n, so 2^n terms in total. He uses the symbol V_r for the subspace containing all terms of order $r = 0, 1, \ldots, n$. He also writes $\omega = \iota_1 \iota_2 \cdots \iota_n$ and finds the value of

$$\omega^2 = \iota_1 \iota_2 \cdots \iota_n \iota_1 \iota_2 \cdots \iota_n.$$

Changing the order of multiplication of any two adjacent unequal elements changes the sign of the product, so if after r interchanges the result is

$$\omega^2 = (-1)^r \iota_1^2 \iota_2^2 \cdots \iota_n^2 = \begin{cases} (-1)^r & \text{if } \iota_j^2 = +1 \\ (-1)^{r+n} & \text{if } \iota_j^2 = -1, \end{cases}$$

then

$$\begin{aligned} r = 1 + 2 + \cdots + (n-1) \qquad &= \frac{1}{2}n(n-1) \\ r + n = 1 + 2 + \cdots + (n-1) + n &= \frac{1}{2}n(n+1). \end{aligned}$$

Both are odd if $n \pmod 4 = 2$, both are even if $n \pmod 4 = 0$, the first is even and the second is odd if $n \pmod 4 = 1$ and the first is odd and the second is even if $n \pmod 4 = 3$.

Clifford then considers products of the form $\iota_j\omega$ and $\omega\iota_j$ and discovers that $\omega\iota_j = +\iota_j\omega$ if n is odd and $\omega\iota_j = -\iota_j\omega$ if n is even. The sign of ι_j^2 is irrelevant.

In a later article of 1878 [26], which was published in its own right, Clifford joins quaternions with Grassmann's algebra. In three dimensions, he takes four points ι_0, ι_1, ι_2 and ι_3 with ι_1, ι_2 and ι_3 at infinite distances from ι_0 in three mutually perpendicular directions. He notes that the quaternion **i**, **j** and **k** can be thought of as operators that turn a figure through 90°in the y-z, x-z and x-y planes respectively. Then considering the lines $\iota_0\iota_1$, $\iota_0\iota_2$ and $\iota_0\iota_3$, we have

$$\begin{aligned} \mathbf{i}\iota_0\iota_2 &= \iota_0\iota_3 \\ \mathbf{j}\iota_0\iota_3 &= \iota_0\iota_1 \\ \mathbf{k}\iota_0\iota_1 &= \iota_0\iota_2. \end{aligned}$$

In a sense, the operators effectively translate the points at infinity $\iota_2 \to \iota_3$, $\iota_3 \to \iota_1$ and

$\iota_1 \to \iota_2$ along the lines $\iota_2\iota_3$, $\iota_3\iota_1$ and $\iota_1\iota_2$ respectively. If we use bivector representations and write $\mathbf{i} = \iota_2\iota_3$, $\mathbf{j} = \iota_3\iota_1$ and $\mathbf{k} = \iota_1\iota_2$ and use Clifford's versions of Grassmann's inner and outer product defined earlier, we also have

$$\begin{aligned}
\iota_0\iota_3 &= \iota_2\iota_3\iota_0\iota_2 = -\iota_2^2\iota_0\iota_3 \\
\iota_0\iota_1 &= \iota_3\iota_1\iota_0\iota_3 = -\iota_3^2\iota_0\iota_1 \\
\iota_0\iota_2 &= \iota_1\iota_2\iota_0\iota_1 = -\iota_1^2\iota_0\iota_2,
\end{aligned}$$

so we must have $\iota_1^2 = \iota_2^2 = \iota_3^2 = -1$. Clifford found that 1, $\iota_2\iota_3$, $\iota_3\iota_1$ and $\iota_1\iota_2$ satisfied exactly the rules of multiplication of the quaternion 1, $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$ respectively. Thus the quaternions are isomorphic to an even subalgebra, where the three basis elements each square to -1.

Clifford went on to consider quaternions with complex coefficients and then with quaternion coefficients.

It is not clear exactly when or where the geometric product of two vectors first appeared, but Clifford is credited with introducing it: if $\mathbf{a}$ and $\mathbf{b}$ are two vectors in n-dimensional space, then the geometric product is defined as

$$\mathbf{ab} = \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \wedge \mathbf{b},$$

so we also have

$$\mathbf{ba} = \mathbf{b} \cdot \mathbf{a} + \mathbf{b} \wedge \mathbf{a} = \mathbf{a} \cdot \mathbf{b} - \mathbf{a} \wedge \mathbf{b},$$

which leads to

$$\mathbf{a} \cdot \mathbf{b} = \frac{1}{2}(\mathbf{ab} + \mathbf{ba}) \qquad \text{and} \qquad \mathbf{a} \wedge \mathbf{b} = \frac{1}{2}(\mathbf{ab} - \mathbf{ba}).$$

As an example of the usefulness of Clifford's geometric algebra, it is used in [25] to greatly simplify Maxwell's equations. In this article, the authors show how the equations

$$\begin{aligned}
\nabla \cdot \mathbf{E} &= \frac{\rho}{\epsilon_0} \quad \text{(Gauss's law)}\\
\nabla \times \mathbf{B} - \frac{1}{c^2}\frac{\partial \mathbf{E}}{\partial t} &= \mu_0 \mathbf{J} \quad \text{(Ampère's circuital law with Maxwell's addition)}\\
\nabla \times \mathbf{E} + \frac{\partial \mathbf{B}}{\partial t} &= \mathbf{0} \quad \text{(Faraday's law of induction)}\\
\nabla \cdot \mathbf{B} &= 0 \quad \text{(Gauss's law of magnetism)}
\end{aligned}$$

may be reduced to

$$\partial F = J,$$

where

$$\begin{aligned}
\partial &= \frac{1}{c}\frac{\partial}{\partial t} + \nabla\\
F &= \mathbf{E} + jc\mathbf{B}\\
J &= \frac{\rho}{\epsilon_0} - c\mu_0 \mathbf{J}.
\end{aligned}$$

Here, $j = \mathbf{e}_1\mathbf{e}_2\mathbf{e}_3$ where the $\mathbf{e}_j$ is modern notation for Clifford's ι_j; $\mathbf{E}$ is the electric field vector $E_1\mathbf{e}_1 + E_2\mathbf{e}_2 + E_3\mathbf{e}_3$, $\mathbf{B}$ is the magnetic field vector $B_1\mathbf{e}_1 + B_2\mathbf{e}_2 + B_3\mathbf{e}_3$ so that $j\mathbf{B}$ is $\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3(B_1\mathbf{e}_1 + B_2\mathbf{e}_2 + B_3\mathbf{e}_3)$, which is the bivector $B_1\mathbf{e}_2\mathbf{e}_3 + B_2\mathbf{e}_3\mathbf{e}_1 + B_3\mathbf{e}_1\mathbf{e}_2$, ρ is the charge density scalar and $\mathbf{J}$ is the current density vector $J_1\mathbf{e}_1 + J_2\mathbf{e}_2 + J_3\mathbf{e}_3$.

Apart from the obvious simplicity of $\partial F = J$, the geometric algebra formulation has some other advantages over the conventional vector notation. The electric and magnetic fields are two quantities with different physical properties and above, $\mathbf{E}$ is a vector and $j\mathbf{B}$ is a bivector: the conventional notation represents both with the same type of mathematical object and we have to remember that $\mathbf{E}$ is a polar vector and $\mathbf{B}$ is an axial vector. Complex numbers are used in electrical engineering theory to allow a simple representation of sinusoidal alternating current; they also have applications to complex permittivity and permeability, *etc.* but they lack physical justification. Lastly, a single equation for electromagnetic theory matches the single equation for Newton's law of universal gravitation and the single equation that Einstein's field equations may be reduced to in general relativity.

2.4 Properties of Clifford Algebras

A good source for Clifford algebras is [151]; another, which better shows the relationships between Clifford algebras and other areas of maths is [105].

Different authors introduce Clifford algebras in different ways. We shall use an approach which is appropriate for our purposes, mirroring as far as possible Section 2.2.

Definition 2.4.1. The basis elements of a Clifford algebra are written $\mathbf{e}_1, \mathbf{e}_2, \ldots$ and may be called vectors, 1-vectors or 1-blades.

Definition 2.4.2. The (geometric) product of any two basis vectors is written $\mathbf{e}_\alpha \mathbf{e}_\beta = \mathbf{e}_{\alpha\beta}$ and is called a bivector, 2-vector or 2-blade. A product of any three basis vectors is written $\mathbf{e}_\alpha \mathbf{e}_\beta \mathbf{e}_\gamma = \mathbf{e}_{\alpha\beta\gamma}$ and is called a trivector, 3-vector or 3-blade. Similarly, a product of $n > 3$ basis vectors is written $\mathbf{e}_\alpha \mathbf{e}_\beta \mathbf{e}_\gamma \mathbf{e}_\delta \cdots = \mathbf{e}_{\alpha\beta\gamma\delta\cdots}$ and is called an n-vector or n-blade.

Definition 2.4.3. The multiplicative identity is $\mathbf{e}_0$ and is called the scalar: this is isomorphic to 1 and commutes with all vectors, bivectors, *etc.*

Definition 2.4.4. The signature of a Clifford algebra may be written $Cl(p, q, r)$, where

$$\mathbf{e}_\alpha^2 = \begin{cases} +\mathbf{e}_0 & \text{for } \alpha = 1, 2, \ldots, p \\ -\mathbf{e}_0 & \text{for } \alpha = p+1, p+2, \ldots, p+q \\ \quad 0 & \text{for } \alpha = p+q+1, p+q+2, \ldots, p+q+r, \end{cases}$$

so p is the number of basis vectors that square to $+\mathbf{e}_0$, q is the number that square to $-\mathbf{e}_0$ and r is the number that square to 0.

Remark 2.4.5. If $r = 0$, we write $Cl(p, q)$.

Definition 2.4.6. Basis vectors anticommute, so $\mathbf{e}_{\alpha\beta} = -\mathbf{e}_{\beta\alpha}$. This property can be used to simplify products of vectors, bivectors, *etc.*

Example 2.4.7. In $Cl(4, 3)$, we can write

$$\begin{aligned}
\mathbf{e}_7\mathbf{e}_{156}\mathbf{e}_{26}\mathbf{e}_{235} &= \mathbf{e}_{715626235} \\
&= -\mathbf{e}_{715(66)(22)35} \\
&= -\mathbf{e}_{715(66)35} && \text{because } \mathbf{e}_2^2 = +\mathbf{e}_0 \\
&= \mathbf{e}_{71535} && \text{because } \mathbf{e}_6^2 = -\mathbf{e}_0 \\
&= -\mathbf{e}_{71(55)3} \\
&= \mathbf{e}_{713} && \text{because } e_5^2 = -\mathbf{e}_0 \\
&= -\mathbf{e}_{173} \\
&= \mathbf{e}_{137}.
\end{aligned}$$

It is usual to write bivectors, trivectors *etc.* with the basis vector factors in numerical order.

Definition 2.4.8. The last element in a clifford algebra, the $(p+q+r)$-vector, is called the pseudoscalar. This commutes with all vectors, bivectors, *etc.*

Definition 2.4.9. For each Clifford algebra, the sum of two or more out of the set {scalar, vectors, bivectors, . . . , pseudoscalar}, is called a multivector.

Example 2.4.10. In a four-dimensional algebra, a multivector may be written

$$\mathbf{m} = a\mathbf{e}_0 + b\mathbf{e}_1 + c\mathbf{e}_2 + d\mathbf{e}_{12},$$

where $a, b, c, d \in \mathbb{R}$ are its components.

Definition 2.4.11. A grade-r multivector is one formed solely of r-blades.

Definition 2.4.12. The conjugate of $\mathbf{m}$ is

$$\overline{\mathbf{m}} = a\mathbf{e}_0 - b\mathbf{e}_1 - c\mathbf{e}_2 - d\mathbf{e}_{12}.$$

Definition 2.4.13. The modulus of $\mathbf{m}$ is

$$|\mathbf{m}| = \sqrt{\mathbf{m}\overline{\mathbf{m}}} = \sqrt{a^2 + b^2 + c^2 + d^2}.$$

Definition 2.4.14. The reverse of a multivector, denoted by a superscript $\dagger$, reverses the order of every product of vectors, so

$$\mathbf{e}_\alpha^\dagger = \mathbf{e}_\alpha, \quad \mathbf{e}_{\alpha\beta}^\dagger = \mathbf{e}_{\beta\alpha} = -\mathbf{e}_{\alpha\beta}, \quad \mathbf{e}_{\alpha\beta\gamma}^\dagger = \mathbf{e}_{\gamma\beta\alpha} = -\mathbf{e}_{\gamma\alpha\beta} = \mathbf{e}_{\alpha\gamma\beta} = -\mathbf{e}_{\alpha\beta\gamma}, \quad \textit{etc.}$$

Remark 2.4.15. Some products are negated as above and some do not change sign, *e.g.* $\mathbf{e}_{\alpha\beta\gamma\delta}^\dagger = \mathbf{e}_{\alpha\beta\gamma\delta}$ and $\mathbf{e}_{\alpha\beta\gamma\delta\epsilon}^\dagger = \mathbf{e}_{\alpha\beta\gamma\delta\epsilon}$.

Definition 2.4.16. Provided $\mathbf{m}\mathbf{m}^\dagger \neq 0$, the inverse of $\mathbf{m}$ is

$$\mathbf{m}^{-1} = \frac{\mathbf{m}^\dagger}{\mathbf{m}\mathbf{m}^\dagger}.$$

The component-wise representation of this is dependent upon the particular algebra in question. This definition is sufficient for $Cl(1,1)$ and $Cl(2,0)$: higher dimensional algebras have more complicated inverses [74].

Remark 2.4.17. We shall only be interested in $Cl(1,0)$, $Cl(1,1)$ and $Cl(2,0)$. We describe the dimensions of these algebras as two, four and four respectively, but a mathematician would only count the number of basis vectors and call the dimensions of these one, two and two. Note that $\mathbb{R} \cong Cl(0,0)$, $\mathbb{C} \cong Cl(0,1)$ and $\mathbb{H} \cong Cl(0,2)$, where $\cong$ means 'is isomorphic to'.

It is convenient to record products of the vectors and bivectors in multiplication tables, as per Tables 2.1(a) and (b) and 2.2(a), (b) and (c). We include the complex numbers and quaternions for completeness.

Table 2.1: Multiplication tables for the two-dimensional Clifford algebras

(a) $Cl(0,1) \cong \mathbb{C}$

	$\mathbf{e}_0$	$\mathbf{e}_1$
$\mathbf{e}_0$	$\mathbf{e}_0$	$\mathbf{e}_1$
$\mathbf{e}_1$	$\mathbf{e}_1$	$-\mathbf{e}_0$

(b) $Cl(1,0)$

	$\mathbf{e}_0$	$\mathbf{e}_1$
$\mathbf{e}_0$	$\mathbf{e}_0$	$\mathbf{e}_1$
$\mathbf{e}_1$	$\mathbf{e}_1$	$\mathbf{e}_0$

Table 2.2: Multiplication tables for the four-dimensional Clifford algebras

(a) $Cl(0,2) \cong \mathbb{H}$

	$\mathbf{e}_0$	$\mathbf{e}_1$	$\mathbf{e}_2$	$\mathbf{e}_{12}$
$\mathbf{e}_0$	$\mathbf{e}_0$	$\mathbf{e}_1$	$\mathbf{e}_2$	$\mathbf{e}_{12}$
$\mathbf{e}_1$	$\mathbf{e}_1$	$-\mathbf{e}_0$	$\mathbf{e}_{12}$	$-\mathbf{e}_2$
$\mathbf{e}_2$	$\mathbf{e}_2$	$-\mathbf{e}_{12}$	$-\mathbf{e}_0$	$\mathbf{e}_1$
$\mathbf{e}_{12}$	$\mathbf{e}_{12}$	$\mathbf{e}_2$	$-\mathbf{e}_1$	$-\mathbf{e}_0$

(b) $Cl(1,1)$

	$\mathbf{e}_0$	$\mathbf{e}_1$	$\mathbf{e}_2$	$\mathbf{e}_{12}$
$\mathbf{e}_0$	$\mathbf{e}_0$	$\mathbf{e}_1$	$\mathbf{e}_2$	$\mathbf{e}_{12}$
$\mathbf{e}_1$	$\mathbf{e}_1$	$\mathbf{e}_0$	$\mathbf{e}_{12}$	$\mathbf{e}_2$
$\mathbf{e}_2$	$\mathbf{e}_2$	$-\mathbf{e}_{12}$	$-\mathbf{e}_0$	$\mathbf{e}_1$
$\mathbf{e}_{12}$	$\mathbf{e}_{12}$	$-\mathbf{e}_2$	$-\mathbf{e}_1$	$\mathbf{e}_0$

(c) $Cl(2,0)$

	$\mathbf{e}_0$	$\mathbf{e}_1$	$\mathbf{e}_2$	$\mathbf{e}_{12}$
$\mathbf{e}_0$	$\mathbf{e}_0$	$\mathbf{e}_1$	$\mathbf{e}_2$	$\mathbf{e}_{12}$
$\mathbf{e}_1$	$\mathbf{e}_1$	$\mathbf{e}_0$	$\mathbf{e}_{12}$	$\mathbf{e}_2$
$\mathbf{e}_2$	$\mathbf{e}_2$	$-\mathbf{e}_{12}$	$\mathbf{e}_0$	$-\mathbf{e}_1$
$\mathbf{e}_{12}$	$\mathbf{e}_{12}$	$-\mathbf{e}_2$	$\mathbf{e}_1$	$-\mathbf{e}_0$

2.5 Real Matrix Representations of Complex Numbers, Quaternions and Clifford Algebras

Recall that complex numbers may be represented as real 2×2 matrices with

$$1 \mapsto \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \text{and} \qquad \mathrm{i} \mapsto \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}. \tag{2.3}$$

In a similar fashion, the basis elements of $Cl(1,0)$ may be represented as

$$\mathbf{e}0 \mapsto \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \text{and} \qquad \mathbf{e}_1 \mapsto \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{2.4}$$

Both quaternions and matrices are non-commutative and it turns out that a quaternion can be represented as a 4×4 real matrix in a number of different ways: Farebrother *et al.* [39] found 48 distinct ordered sets of three 4×4 matrices which would serve as the imaginary basis elements of a quaternion, the 4×4 identity matrix in each case being the first basis element, representing the scalar part. The matrices we shall use are

$$\begin{gathered} 1 \mapsto \mathbf{I}_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad \mathbf{i} \mapsto \mathbf{M}_{\mathbf{i}} = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \\ \mathbf{j} \mapsto \mathbf{M}_{\mathbf{j}} = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix}, \qquad \mathbf{k} \mapsto \mathbf{M}_{\mathbf{k}} = \begin{pmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}. \end{gathered} \tag{2.5}$$

By analogy with the relations between the basis elements of the quaternions in Equation (2.2), we have

$$\mathbf{Q} = a\mathbf{I}_4 + b\mathbf{M}_\mathbf{i} + c\mathbf{M}_\mathbf{j} + d\mathbf{M}_\mathbf{k}$$

and

$$\mathbf{M}_\mathbf{i}^2 = \mathbf{M}_\mathbf{j}^2 = \mathbf{M}_\mathbf{k}^2 = \mathbf{M}_\mathbf{i}\mathbf{M}_\mathbf{j}\mathbf{M}_\mathbf{k} = -\mathbf{I}_4,$$
$$\mathbf{M}_\mathbf{i}\mathbf{M}_\mathbf{j} = -\mathbf{M}_\mathbf{j}\mathbf{M}_\mathbf{i} = \mathbf{M}_\mathbf{k}, \quad \mathbf{M}_\mathbf{j}\mathbf{M}_\mathbf{k} = -\mathbf{M}_\mathbf{k}\mathbf{M}_\mathbf{j} = \mathbf{M}_\mathbf{i}, \quad \mathbf{M}_\mathbf{k}\mathbf{M}_\mathbf{i} = -\mathbf{M}_\mathbf{i}\mathbf{M}_\mathbf{k} = \mathbf{M}_\mathbf{j}.$$

We then have, for the **q** in Definition 2.2.1,

$$\mathbf{q} \mapsto \mathbf{Q} = \begin{pmatrix} a & -b & -c & -d \\ b & a & -d & c \\ c & d & a & -b \\ d & -c & b & a \end{pmatrix},$$

so that the first column gives the scalar and coefficients of the imaginary parts of **q** in order.

Definitions 2.2.3 to 2.2.14 have their matrix equivalents, with transpose replacing conjugate, $\mathbf{I}_4$ replacing the 1, The modulus becoming the fourth root of the determinant times $\mathbf{I}_4$ and the inverse, either the normal matrix inverse, or simply the transpose divided element-wise by the square of the determinant. Any quaternion computation is equivalent to one with the above 4×4 matrices: the first column of the result would give the coefficients of the equivalent quaternion result.

Quaternions also have representations as a group of four 2×2 complex matrices and as a group of four 2×2 real matrices, but we shall not discuss either of these.

All Clifford algebras where $r = 0$ have real matrix representations. We would expect the basis elements of $Cl(1,1)$ and $Cl(2,0)$ to have many possible real 4×4 matrix representations, but the ones we shall use are, for $Cl(1,1)$,

$$\mathbf{e}_0 \mapsto \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad \mathbf{e}_1 \mapsto \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$
$$\mathbf{e}_2 \mapsto \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix}, \qquad \mathbf{e}_{12} \mapsto \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}; \tag{2.6}$$

and for $Cl(2,0)$,

$$\mathbf{e}_0 \mapsto \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad \mathbf{e}_1 \mapsto \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$
$$\mathbf{e}_2 \mapsto \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix}, \qquad \mathbf{e}_{12} \mapsto \begin{pmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}. \tag{2.7}$$

It is straightforward to check that products of pairs of the matrices in each of Equations (2.6) and (2.7) agree with the appropriate multiplication tables in Table 2.2.

We shall make use of these matrix representations of hypercomplex numbers in Chapter 5.

2.6 Colour Vector Images

Colour images are stored as a two-dimensional (2-D) array of integer triplets, each integer telling the computer how much red, green or blue to display in each pixel. However, colour images are conventionally processed as three separate monochrome arrays. The Fourier transform of a colour image is thus performed as three separate Fourier transforms of the three separate monochrome images. Processing a colour image in this way does not take into account correlations between primary colours that exist in secondary colours.

In 1993, Todd Ell [35] developed a continuous 2-D quaternion Fourier transform (QFT),

$$H[\mathbf{j}\omega,\mathbf{k}v] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \exp(-\mathbf{j}\omega t)h(t,\tau)\exp(-\mathbf{k}v\tau)\mathrm{d}t\mathrm{d}\tau,$$

with inverse

$$h(t,\tau) = \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \exp(\mathbf{j}\omega t)H[\mathbf{j}\omega,\mathbf{k}v]\exp(\mathbf{k}v\tau)\mathrm{d}v\mathrm{d}\omega.$$

Stephen Sangwine realised that using this on a colour image, one treated as a single array of quaternion-valued pixels, would result in a single frequency-domain representation instead of the usual three. It should then be possible to find new types of filter, specifically for colour images. The transformation to an array of unit pure quaternions can be done by using the normalised values of red, green and blue in each pixel as the coefficients multiplying **i**, **j** and **k**, *i.e.*,

$$\mathbf{f}(n,m) = r(n,m)\mathbf{i} + g(n,m)\mathbf{j} + b(n,m)\mathbf{k},$$

where $\mathbf{f}(n,m)$ is the pure quaternion representation of the pixel at (n,m) and $r(n,m)$, $g(n,m)$ and $b(n,m)$ are the normalised numbers taken from the red, green and blue monochrome arrays at the pixel. An image represented in this way, as an array of pure quaternions, *i.e.*

an array of the vector parts of quaternions, is called a *colour vector image* or just *vector image*, each pixel effectively being represented as a vector in the RGB colour cube of Figure 2.1. In going to a vector representation, the problem of processing a colour image becomes a geometric one, something for which quaternions were originally developed.

For all practical purposes, any colour can be represented as a point inside or on the surface of this cube. This includes grey, as the vector joining $(0, 0, 0)$ to $(1, 1, 1)$ goes through all shades from black to white and is known as the grey axis.

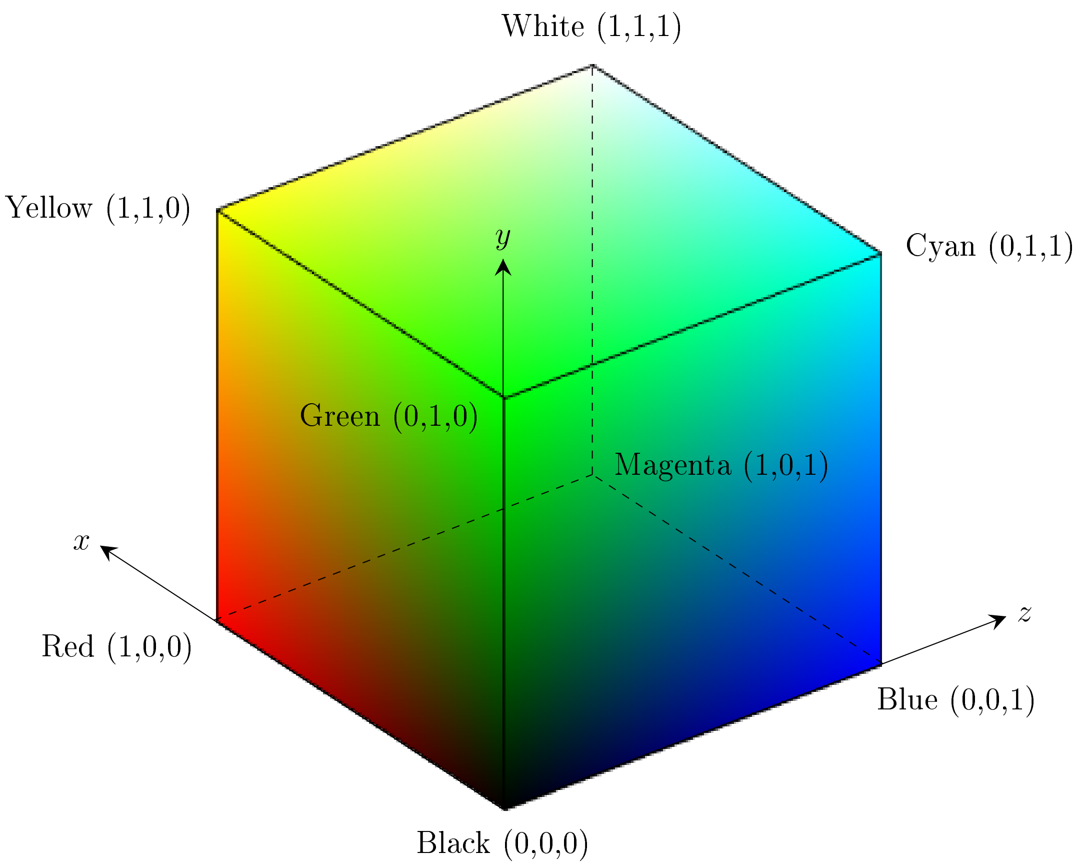


Figure 2.1: The RGB colour cube.[2]

[2]Corrected with right-handed axes from the one at https://stackoverflow.com/questions/29953652/drawing-3-d-rgb-cube-model-with-matlab Accessed 20th June 2018.

By analogy with the pure quaternion representation of a colour pixel, we shall do the same with the four-dimensional Clifford algebras $Cl(1,1)$ and $Cl(2,0)$ and write

$$\mathbf{f}(n,m) = r(n,m)\mathbf{e}_1 + g(n,m)\mathbf{e}_2 + b(n,m)\mathbf{e}_{12}.$$

A mathematician would object to the equal treatment of two different types of object, *viz.* vectors and bivectors, but we justify the above by the fact that the quaternion $\mathbf{i}$ and $\mathbf{j}$ are effectively Clifford vectors and the $\mathbf{k}$, a Clifford bivector.

2.7 The Quaternion Fourier Transform

In 1996, Sangwine rewrote Ell's equations in discrete form [124], making it possible to turn them into code. In 1998 he presented a new type of colour image edge detector for colour images [125], using convolution with quaternion mask coefficients. This was primarily to show the potential of the new representation rather than an end in itself.

Sangwine's long-term aim for a number of years has been to develop linear methods for colour image filtering [126, p. 284]. Very simply, the output from a linear filter is a linear function of its input: halve the input and the output is halved. Such filters obey the *principle of superposition*. Suppose a signal s_{in} is decomposed into two parts, each part is filtered by a copy of the same filter F and the filtered outputs are added together to give a signal $s_{out,1}$. Suppose also that s_{in} is not decomposed, but filtered by a single filter F to give an output signal $s_{out,2}$. Only if F is linear, would it be found that $s_{out,1} = s_{out,2}$. Sangwine did not realise it at the time, but the 1998 edge detector was a linear filter [126, p. 289].

In 1998, Sangwine and Ell [129] used a non-specific quaternion $\sqrt{-1}$, $\boldsymbol{\mu} = \boldsymbol{V}(\mathbf{q})/|\boldsymbol{V}(\mathbf{q})|$, to rewrite Ell's QFT in a more general form, with a single exponential:

$$\mathcal{F}[u,v] = \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \exp\big(-\boldsymbol{\mu} 2\pi(mv/M + nu/N)\big) f(n,m).$$

Recently, Ell *et al.* [37, p. 48] have given eight possible forms of the 2-D QFT.

2.8 An Example of Colour Vector Image Processing

In 2005, Sangwine and Le Bihan released their toolbox for MATLAB® [128], which has been updated regularly since then. As an example of the use of this toolbox, we replicate the results of Denis *et al.* [32], who devised a colour gradient edge detector based on Sangwine's 1998 one [125]. The method they used was to filter the chosen image using four separate pairs of masks to detect horizontal, vertical, diagonal and antidiagonal edges. These masks were:

$$\boldsymbol{l} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ \boldsymbol{R} & \boldsymbol{R} & \boldsymbol{R} \end{pmatrix} \quad \text{and} \quad \boldsymbol{r} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ \overline{\boldsymbol{R}} & \overline{\boldsymbol{R}} & \overline{\boldsymbol{R}} \end{pmatrix},$$

$$\boldsymbol{l}^T \quad \text{and} \quad \boldsymbol{r}^T,$$

$$\boldsymbol{l}_{diag} = \begin{pmatrix} 0 & 1 & 1 \\ \boldsymbol{R} & 0 & 1 \\ \boldsymbol{R} & \boldsymbol{R} & 0 \end{pmatrix} \quad \text{and} \quad \boldsymbol{r}_{diag} = \begin{pmatrix} 0 & 1 & 1 \\ \overline{\boldsymbol{R}} & 0 & 1 \\ \overline{\boldsymbol{R}} & \overline{\boldsymbol{R}} & 0 \end{pmatrix},$$

$$\boldsymbol{l}_{diag}^T \quad \text{and} \quad \boldsymbol{r}_{diag}^T,$$

where $\boldsymbol{R} = \exp(\boldsymbol{\mu}\pi/2)$ and $\boldsymbol{\mu} = (\mathbf{i}+\mathbf{j}+\mathbf{k})/\sqrt{3}$, the greyscale axis.

For each pixel $\mathbf{p}_{\text{unfilt}}(s,t)$ in Figure 2.2a and $\mathbf{p}_{\text{filt}}(s,t)$ in Figure 2.2b and each pair of masks above, with $*$ indicating convolution,

$$\mathbf{p}_{\text{filt}}(s,t) = ([\text{Left mask}] * \mathbf{p}_{\text{unfilt}} * [\text{Right mask}])\,(s,t),$$

giving four potential values $\mathbf{p}_{\text{filt}}(s,t)$ for each pixel.

The RGB colour system is not the only one, HSV being another, where H is hue, S is saturation and V is value. In terms of a pure quaternion, $\mathrm{S} = 1/2|\mathbf{q} + \boldsymbol{\mu}\mathbf{q}\boldsymbol{\mu}|$ and each pixel making up the image in Figure 2.2b is the one with the maximum saturation of the four potential potential values $\mathbf{p}_{\text{filt}}(s,t)$. Where the colour changes only gradually the saturation is small, but where there is a colour edge and there is a sudden change in colour, the saturation is large such that, as can be seen, the edges become clear. Our code is in Appendix A.

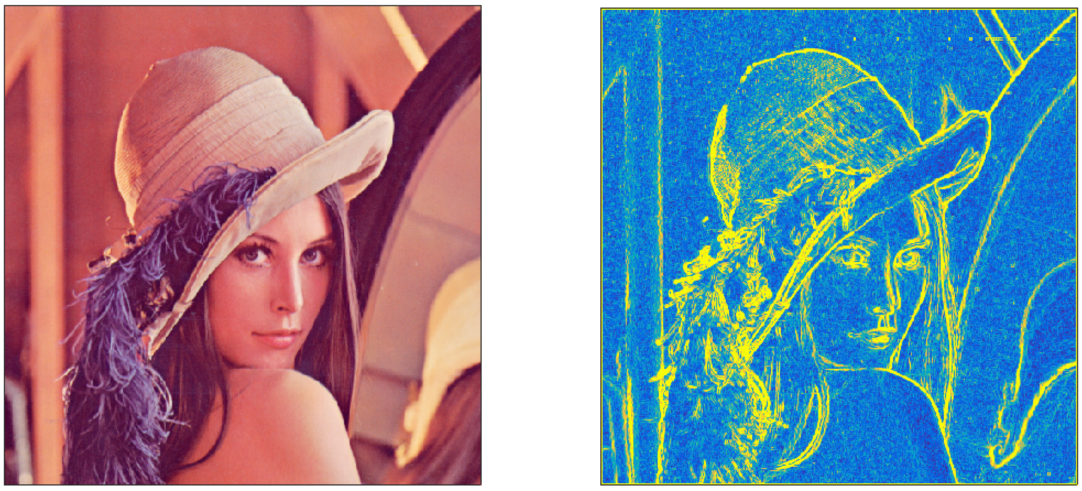

(a) Lena (b) Edge-filtered Lena

Figure 2.2: Lena before and after filtering using Denis *et al.*'s method.

2.9 Chapter Summary

We started with a brief history of complex numbers, followed by quaternions and some of their properties. We followed this with a brief history of Clifford algebras, followed by some of their properties. We showed that complex numbers, quaternions and the Clifford algebras we are mainly interested in are isomorphic to particular groups of real matrices. We then introduced the colour cube and colour vector images and briefly mentioned quaternion Fourier transforms, before finishing with an example of colour vector image processing not involving wavelets.

Chapter 3

Real Wavelets and Discrete Wavelet Transforms

In this chapter we introduce real wavelets and discrete wavelet transforms. In Section 3.1 we briefly cover the background to real wavelets and in Section 3.2 we give their basic properties. In Section 3.3 we introduce multiresolution analysis and in Section 3.4, we show how each level of analysis may be done in practice. In Section 3.5 we explain how real Daubechies filters can be calculated and in Section 3.6 we show how they may be used to find associated continuous functions.

3.1 Background

The Fourier transform (FT) gives information about the frequency content of a signal, but says nothing about where in time or space its different constituent frequencies occur. In some applications, for example with non-stationary signals, it might be desirable to know the distribution of frequencies over time or space, which led to the introduction of the short-time Fourier transform (STFT) by Gabor [47]. A ‘sliding window’ is introduced into the FT, initially centred at time t, say; the signal is assumed to be approximately stationary in the window and its FT is found. The window is then shifted by t and the FT of the new section of signal is found, and so on until the whole signal has been covered. In continuous time and frequency, the STFT in one dimension centred on time t can be expressed as

$$\mathcal{F}_{\text{STFT}} f(\omega, t) = \int_{-\infty}^{\infty} f(s) g(s-t) \mathrm{e}^{-j\omega s} \mathrm{d}s, \tag{3.1}$$

where $g(\cdot)$ is the window function. In discrete time the STFT becomes

$$F_{\text{STFT}}(\omega, t) = \sum_{m=-\infty}^{\infty} f[m] g[m-t] \mathrm{e}^{-j\omega m}. \tag{3.2}$$

The function $g(\cdot)$ or $g[\cdot]$ will always be even in practice and some authors would write $g(t-s)$ and $g[t-m]$ above. Gabor experimented with a number of different functions for the window $g(\cdot)$ and found the best he could do was to use a Gaussian; the STFT with this window function is now called the Gabor transform. We cannot know the exact frequency at a given time and Heisenberg's Uncertainty Principle applies per Gabor [48]:

$$\Delta t \Delta \omega \geqslant \frac{1}{2}, \tag{3.3}$$

where Δt is the uncertainty in time and $\Delta \omega$ is the uncertainty in angular frequency. The Gabor transform is optimal in the sense that this inequality theoretically becomes an equality when $g(\cdot)$ is a Gaussian.

A major drawback of the STFT is that once the window is chosen its resolution is fixed, whereas a wavelet transform has good time resolution at high frequencies and good frequency resolution at low frequencies.

Per Gao and Yan [58], Haar [69] developed a set of orthogonal rectangular basis functions, with each basis function consisting of a short positive pulse followed immediately by a short negative pulse:

$$\psi(t) := \begin{cases} 1 & 0 \leqslant t \leqslant \frac{1}{2} \\ -1 & \frac{1}{2} < t \leqslant 1 \\ 0 & \text{otherwise.} \end{cases} \tag{3.4}$$

Later, after the development of wavelets, this rectangular pulse function was named the Haar wavelet and an associated so-called scaling function was introduced:

$$\phi(t) := \begin{cases} 1 & 0 \leqslant t \leqslant 1 \\ 0 & \text{otherwise.} \end{cases} \tag{3.5}$$

These two pulses form orthogonal basis functions, from which further basis functions are derived: $\phi_{\alpha,\beta}(t) := 2^{-\frac{\alpha}{2}}\phi(2^{-\alpha}t - \beta)$ and $\psi_{\alpha,\beta}(t) := 2^{-\frac{\alpha}{2}}\psi(2^{-\alpha}t - \beta)$, where $\alpha, \beta \in \{0\} \cup \mathbb{Z}^+$. The terms multiplying $\phi_{\alpha,\beta}(\cdot)$ and $\psi_{\alpha,\beta}(\cdot)$ are to ensure that $|\phi_{\alpha,\beta}(\cdot)| = 1$ and $|\psi_{\alpha,\beta}(\cdot)| = 1$. The scaling function $\phi(t)$ is sometimes called the father wavelet and $\psi(t)$, the mother wavelet with the $\psi_{\alpha,\beta}(t)$ functions, daughter wavelets; α is a scale parameter and β is a translation parameter.

There was little interest in Haar's rectangular pulse until it was picked up by Lévy [92] as an improvement on the Fourier basis functions for studying the fine detail of Brownian motion: as demonstrated by Pinsky [119], Brownian motion can be expressed as a sum of Haar wavelets.

A square pulse is not the only wavelet that can be used. Strömberg [142] started the development of discrete wavelets beyond what Haar had done and Daubechies [28] introduced families of orthogonal wavelets with compact support.

3.2 Properties of Wavelets

Many wavelets have been developed, each with properties suited to particular applications, but all have certain features in common: finite support and localisation in space and time. A wavelet having finite support simply means that there is a finite set of points at which it is non-zero. A wavelet transform should be capable of analysing an image at different scales and so the underlying wavelets need to be localised in spatial frequency. It also needs to encode where in two dimensional space these frequencies occur and so the wavelets need to be localised in two dimensions as well. A one dimensional example of space and time localisation would be a music score, which shows what sound frequencies need to occur and at what times. Wavelet analysis of the music would theoretically allow the music score to be reconstructed, but Fourier analysis of the same music would not.

The wavelet functions are chosen from $L^1(\mathbb{R}) \cap L^2(\mathbb{R})$, the space of measurable functions that are absolutely and square integrable:

$$\int_{-\infty}^{\infty} |\psi(t)| \mathrm{d}t < \infty, \qquad \text{and} \qquad \int_{-\infty}^{\infty} |\psi(t)|^2 \mathrm{d}t < \infty.$$

In addition, a wavelet must have mean zero and squared norm one, so:

$$\int_{-\infty}^{\infty} \psi(t) \mathrm{d}t = 0 \qquad \text{and} \qquad \int_{-\infty}^{\infty} |\psi(t)|^2 \mathrm{d}t = 1.$$

We define a wavelet as

$$\psi_{a,b} : \mathbb{R} \to \mathbb{C}, \qquad t \mapsto \frac{1}{\sqrt{a}} \psi\left(\frac{t-b}{a}\right),$$

where $(a, b) \in \mathbb{R}^+ \times \mathbb{R}$. Note that this definition of $\psi_{a,b}$ is slightly different from the one we used for the $\psi_{\alpha,\beta}$ of the Haar wavelet above: if $\psi_{a,b}$ were that Haar wavelet, we would have $a = 2^\alpha$ and $b = 2^\alpha\beta$. Both definitions are in use and we also call a and b the scale and translation parameters respectively.

3.3 Multiresolution Analysis

Multiresolution analysis (MRA) was developed by Mallat [107] and Meyer [109] as a design method for scaling functions and wavelets. We do not use MRA as a design tool, but rather as a means of analysing images in Chapter 6. The idea is that a function (in our case image) is viewed at the finest resolution and decomposed into the detail at that resolution plus a complementary approximation; this approximation is then viewed at a lower resolution and decomposed into the detail at the new resolution plus a new approximation; and so on.

With MRA, we consider the spaces that the detail and approximation functions resulting from a multilevel analysis have to belong to. We can then develop conditions that these functions must obey. Let $\boldsymbol{V}_s$ be generated by the approximation bases $\{\phi_{k,s}\colon 2^{-s/2}\phi(2^{-s}t - k); k \in \mathbb{Z}\}$ and $\boldsymbol{W}_s$ by the detail bases $\{\psi_{k,s}\colon 2^{-s/2}\psi(2^{-s}t - k); k \in \mathbb{Z}\}$ so that any function $x_s(t) \in \boldsymbol{V}_s$, say, can be represented as a linear combination of $\phi_{k,s}$ for a finite set of different k's and similarly for any function $y_s(t) \in \boldsymbol{W}_s$, say, and $\psi_{k,s}$. The approximation $x_s(t)$ and detail $y_s(t)$ both come from the approximation $x_{s-1}(t) \in \boldsymbol{V}_{s-1}$.

In the following, we define for $a \in \mathbb{R}^+$,

$$D_a: \quad \psi \mapsto D_a\psi, \qquad D_a\psi(t) := \psi\left(\frac{t}{a}\right).$$

Per Blatter [14, pp. 121/2] (and many other authors), a multiresolution analysis has the following ingredients.

1. A bilateral sequence $(\boldsymbol{V}_j, j \in \mathbb{Z})$ of closed subspaces of $L^2(\mathbb{R})$. These $\boldsymbol{V}_j$ are ordered

by inclusion,

$$\{0\} \ldots \subset \boldsymbol{V}_2 \subset \boldsymbol{V}_1 \subset \boldsymbol{V}_0 \subset \boldsymbol{V}_{-1} \subset \ldots \boldsymbol{V}_j \subset \boldsymbol{V}_{j-1} \subset \ldots \subset L^2(\mathbb{R}) \tag{3.6}$$

(smaller values of j correspond to larger spaces $\boldsymbol{V}_j$), and one has

$$\bigcap_j \boldsymbol{V}_j = \{0\} \qquad \text{(separation axiom)}$$

$$\overline{\bigcup_j \boldsymbol{V}_j} = L^2(\mathbb{R}) \quad \text{(completeness axiom)}$$

The time signals $x \in \boldsymbol{V}_j$ only comprise features (details) exhibiting a spread of size $\geqslant 2^j$ on the time axis. The more negative j is, the finer are the details that may occur in an $x \in \boldsymbol{V}_j$, and 'in the limit' every single $x \in L^2(\mathbb{R})$ can be attained by functions $x_j \in \boldsymbol{V}_j$.

2. The $\boldsymbol{V}_j$ are connected to each other by a rigid scaling property:

$$\boldsymbol{V}_{j+1} = D_2(\boldsymbol{V}_j) \qquad \forall j \in \mathbb{Z}.$$

Referring to time signals x this can be expressed as follows:

$$x \in \boldsymbol{V}_j \Leftrightarrow x(2^j \cdot) \in \boldsymbol{V}_0.$$

3. $\boldsymbol{V}_0$ contains one basis vector per base step 1. To be precise, there is a function $\phi \in L^2 \cap L^1$ such that its translates $(\phi(\cdot - k), k \in \mathbb{Z})$ form an orthonormal basis of $\boldsymbol{V}_0$. This function ϕ is commonly called the *scaling function* of the MRA under consideration; it is the determining element of the whole set-up.

Some authors reverse the ordering of the $\boldsymbol{V}_j$ in Equation (3.6).

Each $\boldsymbol{V}_s$ is a proper subspace of $\boldsymbol{V}_{s-1}$ and the remaining space in $\boldsymbol{V}_{s-1}$ is called $\boldsymbol{W}_s$, the wavelet subspace, satisfying

$$\begin{aligned} \boldsymbol{V}_s \cap \boldsymbol{W}_s &= \{0\} \quad s \in \mathbb{Z} \\ \boldsymbol{V}_s \oplus \boldsymbol{W}_s &= \boldsymbol{V}_{s-1}, \end{aligned}$$

which leads to

$$V_s = \bigoplus_{\ell=s+1}^{\infty} \boldsymbol{W}_\ell.$$

The $\{\boldsymbol{V}_s\}$ are nested and the $\{\boldsymbol{W}_s\}$ are mutually orthogonal and together satisfy

$$\begin{aligned} \boldsymbol{V}_\ell \cap \boldsymbol{V}_m &= \boldsymbol{V}_\ell, \quad && \ell > m \\ \boldsymbol{W}_\ell \cap \boldsymbol{W}_m &= \{0\}, && \ell \neq m \\ \boldsymbol{V}_\ell \cap \boldsymbol{W}_m &= \{0\}, && \ell \geqslant m. \end{aligned}$$

The way $\boldsymbol{V}_s$ and $\boldsymbol{W}_s$ are related as s changes is illustrated in Figure 3.1.

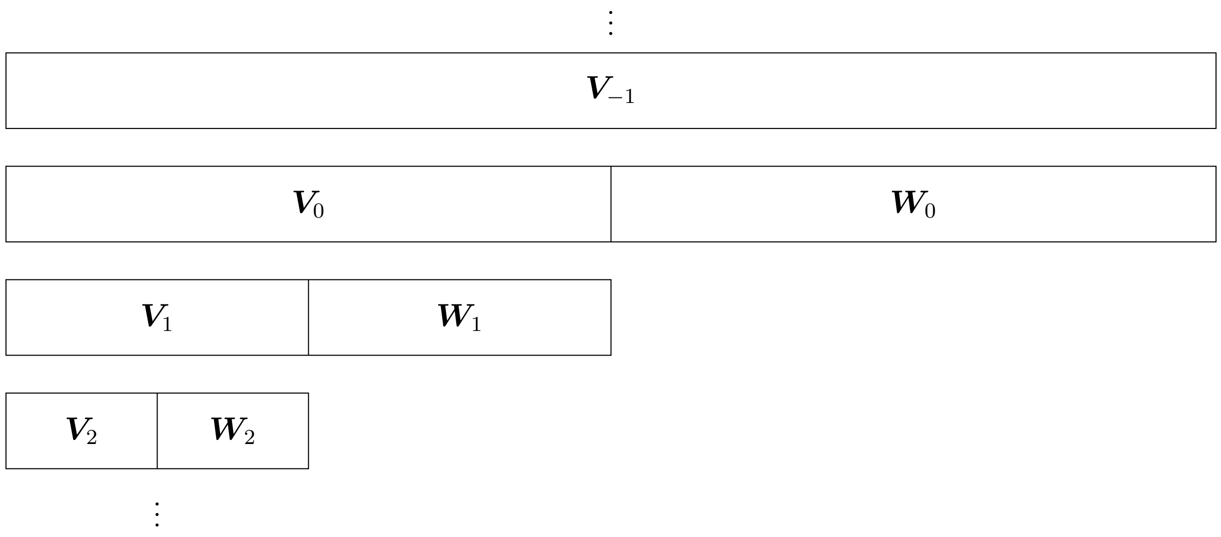


Figure 3.1: Illustration of splitting of MRA subspaces, adapted from Goswami and Chan [63, p. 97 Fig. 5.2].

Referring to Figure 3.1, if our function is in $\boldsymbol{V}_{-1}$, then the first approximation is in $\boldsymbol{V}_0$ with the detail removed at the finest resolution in $\boldsymbol{W}_0$. The next approximation is in $\boldsymbol{V}_1$ with the detail removed at the new, coarser resolution in $\boldsymbol{W}_1$. And so on. We see how the resolution is made coarser at each level in the next subsection.

3.4 The Quadrature Mirror Filter

A wavelet can be viewed as a bandpass filter and a discrete wavelet transform (DWT) and its inverse can be implemented as a two-channel filter bank with finite impulse response filters (FIRs), as illustrated in Figure 3.2. The processing block in practice might perform *e.g.*, signal compression by suppression of small components, but we assume here that it has been removed and the outputs of the downsamplers (decimators) go straight into the upsamplers (expanders). The H's are low-pass filters, the G's are high-pass filters and the tilde indicates that the filter's coefficients have been reversed. Together the four filters form a quadrature mirror filter (QMF). The passband of a filter can never have a perfect cut-off and where filter responses overlap, aliasing can occur. In a QMF, the aliasing that must occur in the analysis bank is cancelled by the equal and opposite aliasing in the synthesis bank, leading to a perfect reconstruction of the input signal, apart from a delay. The aliasing is actually augmented by the decimators, but this makes it easier to remove.

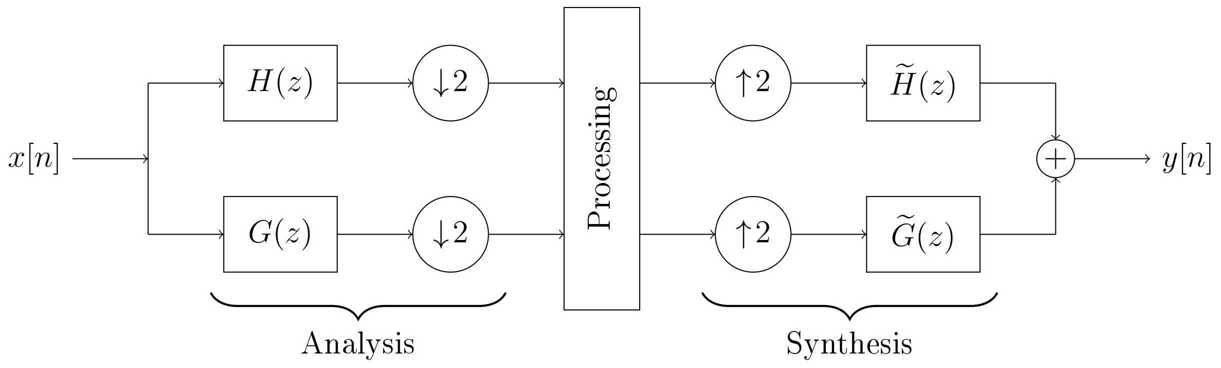


Figure 3.2: The basic layout of a simple discrete wavelet transform.

In terms of z-transforms, with no additional processing between analysis and synthesis, we find that

$$Y(z) = \frac{1}{2}X(z)[H(z)\widetilde{H}(z) + G(z)\widetilde{G}(z)] + \frac{1}{2}X(-z)[H(-z)\widetilde{H}(z) + G(-z)\widetilde{G}(-z)].$$

The second term is due to the decimation and expansion plus aliasing and we can make it zero by choosing $\widetilde{H}(z) = G(-z)$ and $\widetilde{G}(z) = -H(-z)$.

As an example, let the coefficients of H be $\boldsymbol{h} = (a, b, c, d)$ so that:

$$\begin{aligned} H(z) &= a + bz^{-1} + cz^{-2} + dz^{-3}, & \widetilde{H}(z) &= \quad d + cz^{-1} + bz^{-2} + az^{-3}, \\ G(z) &= d - cz^{-1} + bz^{-2} - az^{-3}, & \widetilde{G}(z) &= -a - bz^{-1} + cz^{-2} - dz^{-3}. \end{aligned}$$

Then after some calculation we find that

$$Y(z) = X(z)[(ac + bd)z^{-1} + (a^2 + b^2 + c^2 + d^2)z^{-3} + (ac + bd)z^{-5}].$$

To four decimal places, if $a = -0.1294$, $b = 0.2241$, $c = 0.8365$ and $d = 0.4830$ then $ac + bd = 0.0000$ and $a^2 + b^2 + c^2 + d^2 = 1.0000$, so that $Y(z) = X(z)z^{-3}$, exactly the same as the input but with a delay. These numbers were chosen from the Daubechies db2 wavelet (see Example 3.5.1), but we could have chosen db3, …, the Mexican hat wavelet, the Meyer wavelet, *etc.* The coefficient vectors are always even-length and $\boldsymbol{h}$ and $\boldsymbol{g}$, which is $(d, -c, b, -a)$ in this case, may be different lengths.

For more an MRA, the first analysis stage's low-pass output after downsampling would form the input to another analysis stage and the high-pass output would be saved; and so on to whatever level of analysis is desired. The downsampling coarsens the resolution at each level of analysis, hence 'multi-resolution'.

For reconstruction, the two final analysis outputs would form the two inputs to the first synthesis stage. The output of this would form the low-pass input to the second synthesis stage, with the high-pass input being the last but one analysis high-pass output; and so on until all analysis high-pass outputs have been added back and the original signal reconstructed.

Thus, at each stage of analysis, the detail at that scale is stripped off and at each stage of synthesis, the detail at that scale is added back in.

As an experiment to see how far it is possible to go with the two dimensional version of this procedure, we decomposed a 512×512 pixel monochrome image with wavelets and reduced it to a single pixel through successive analysis and downsampling; we were then able to reconstruct the original image perfectly through successive upsampling and synthesis.

In recent years, wavelet transforms have found applications in video and internet communications compression, object recognition and numerical analysis. As an example of image compression, The JPEG2000 standard uses CDF 5/7 and CDF 9/7 wavelets, which are two versions of the Cohen-Daubechies-Feauveau wavelet.

3.5 Construction of Real Daubechies Scaling Filters

The Daubechies wavelets are perhaps the most popular family of wavelets. In the following, we summarise Strang and Nguyen's approach [141, pp.164–9] to their derivation and construction.

The two key properties of Daubechies filters are:

1. *orthogonality*;
2. *maximum flatness* at $\omega = 0$ and $\omega = \pi$.

We let the length of the filters be $2N$ and the coefficients in the lowpass filter be $c(0), c(1), \ldots,$ $c(2N-1)$, with frequency response $C(\omega)$; we also let $p(0), p(1), \ldots, p(2N-1)$ be the coefficients of the centred even polynomial $P(\omega) = |C(\omega)|^2$ with $p(n) = p(-n)$. The 2N numbers required come from N conditions for orthogonality and N conditions for flatness:

Orthogonality:

$$p(0) = 1, \; p(2) = p(4) = \cdots = p(2N-2) = 0$$

Flatness:

$$C(\pi) = C'(\pi) = \cdots = C^{(N-1)}(\pi) = 0.$$

As Strang and Nguyen explain, these conditions have a number of consequences.

Because $C(\pi) = 0$, this means that $\sum (-1)^n c(n) = 0$ so that

$$\sum_{\text{odd } n} c(n) = \sum_{\text{even } n} c(n)$$

and from the other flatness conditions,

$$\sum_{n=0}^{2N-1}(-1)^n n^k c(n) = 0 \qquad \text{for } k = 0, 1, \ldots, N-1.$$

We also have

$$\sum_{\text{odd } n} p(n) = \sum_{\text{even } n} p(n) = p(0) = 1$$

so that

$$\sum_{\text{all } n} p(n) = P(0) = 2.$$

Then from $P(\omega) = |C(\omega)|^2$, we get $C(0) = \pm\sqrt{2}$ and

$$\sum_{\text{all } n} c(n) = C(0) = \sqrt{P(0)} = \sqrt{2}.$$

The p zeros at π mean that $C(\omega)$ has a factor $(1 + \mathrm{e}^{-\mathrm{i}\omega})$ and

$$C(\omega) = \left(\frac{1+\mathrm{e}^{-\mathrm{i}\omega}}{2}\right)^N R(\omega),$$

where $R(\omega)$ has degree $N-1$, to bring the degree of $C(\omega)$ up to $2N-1$. The N coefficients of $R(\omega)$ satisfy the N orthogonality equations above.

Since $P(\omega) = |C(\omega)|^2$ and $|1 + \mathrm{e}^{-\mathrm{i}\omega}|^2/2 = \big(1 + \cos(\omega)\big)$, we know that $P(\omega)$ has a factor

$$\left(\frac{1+\cos(\omega)}{2}\right)^N.$$

We now find an equation for $P(\omega)$. The starting point suggested by Strang and Nguyen is the polynomial $B_N(y)$, the binomial series for $(1-y)^{-N}$ truncated after N terms:

$$B_N(y) = 1 + Ny + \frac{N(N+1)}{2}y^2 + \cdots + \binom{2N-2}{N-1}y^{N-1} = (1-y)^{-N} + O(y^N).$$

The coefficient of y^k is $\binom{N+k-1}{k}$.

Strang and Nguyen then write

$$\widetilde{P}(y) = 2(1-y)^N B_N(y) = 2(1-y)^N \Big[(1-y)^{-N} + O(y^N)\Big] = 2 + O(y^N),$$

a polynomial of degree $2N-1$. It is the only polynomial with 2N coefficients that satisfies N conditions at each endpoint, *i.e.* $\widetilde{P}(y)$ and its first $p-1$ derivatives are zero at $y=0$ and $y=1$, except that $\widetilde{P}(0)=2$.

We now change to trigonometrical polynomials, taking $0 \le y \le 1$ to $0 \le \omega \le \pi$ by defining

$$y = \frac{1-\cos(\omega)}{2} \qquad \text{or} \qquad 1-y = \frac{1+\cos(\omega)}{2}.$$

The polynomial $\widetilde{P}(y)$ then becomes $P(\omega)$:

$$P(\omega) = 2\left(\frac{1+\cos(\omega)}{2}\right)^N \sum_{k=0}^{N-1} \binom{N+k-1}{k} \left(\frac{1-\cos(\omega)}{2}\right)^k.$$

We now make a second change of variables, to complex z:

$$\frac{z+z^{-1}}{2} = \cos(\omega) = 1-2y$$

or $y = (2-z-z^{-1})/4$. We find that

$$P(z) = 2\left(\frac{1+z}{2}\right)^N \left(\frac{1+z^{-1}}{2}\right)^N \sum_{k=0}^{N-1} \binom{N+k-1}{k} \left(\frac{1-z}{2}\right)^k \left(\frac{1-z^{-1}}{2}\right)^k. \tag{3.7}$$

This factors into $P(z) = C(z)C(z^{-1})$. The N zeros at $y=1$ and $\omega=\pi$ are now $2N$ zeros at $z=-1$; half of them go into $C(z)$. The $N-1$ complex zeros of $B_p(y)$ become $2N-2$ zeros of $P(z)$; again, half of them (the $N-1$ zeros inside the circle $|z|=1$ if we want minimum phase) go into $C(z)$. We can calculate $C(z)$ as follows.

1. Find the $N-1$ roots of $B_N(y) = 0$.
2. Equate each root in turn to $(2 - z - z^{-1})/4$.
3. Solve the resulting quadratics and take the $N-1$ values with $|z| < 1$.
4. Include N zeros at $z = -1$.
5. Expand the resulting polynomial, which will have $2N-1$ zeros and $2N$ coefficients.
6. Divide each coefficient of z by the square root of the sum of the squares of all the coefficients.

The resulting $2N$ numbers will be the required (real) filter coefficients, the squares of which will sum to 1.

Example 3.5.1. We can find the scaling filter coefficients of db2 by setting $N = 2$. Then $B_2 = 1 + 2y$, the zero of which is at $y = -\frac{1}{2}$, and

$$\frac{2 - z - z^{-1}}{4} = -\frac{1}{2}$$

or

$$\frac{4 - z - z^{-1}}{4} = 0,$$

giving $z^2 - 4z - 1$ and $z = 2 \pm \sqrt{3}$. Therefore

$$\begin{aligned} f(z) &= (z+1)^2(z - [2 - \sqrt{3}]) \\ &= z^3 + \sqrt{3}z^2 + (2\sqrt{3} - 3)z - (2 - \sqrt{3}). \end{aligned}$$

The sum of the squares of the coefficients is $32 - 16\sqrt{3}$, the square root of which may be found in, for example, Maple™ to be $2\sqrt{2}(\sqrt{3} - 1)$. Dividing $f(z)$ by this and simplifying gives

$$\frac{f(z)}{2\sqrt{2}(\sqrt{3}-1)} = \frac{1}{4\sqrt{2}}\Big[(1+\sqrt{3})z^3 + (3+\sqrt{3})z^2 + (3-\sqrt{3})z - (\sqrt{3}-1)\Big].$$

Evaluating the coefficients gives the four values 0.4830, 0.8365, 0.2241, -0.1294 respectively, to four decimal places. These are the filter coefficients of the db2 scaling filter we mentioned in Section 3.4 in connection with the QMF.

3.6 The Cascade Algorithm

For each scaling and wavelet filter there exists an associated basis function, akin to the sines and cosines of Fourier analysis. Unlike sines and cosines, these functions have compact support and are not analytic. We may still however, for each function, generate a vector of points that lie on it by using the *cascade* or *two-scale dilation* equation. If we generate a sufficient number of points, we may plot them and obtain what is, to all intents and purposes, a continuous curve. In this section we introduce the cascade algorithm and go on to demonstrate its use on the real Daubechies db2 filters.

Scaling and wavelet functions are *refinable* functions (see *e.g.* [81, p. 3]), defined by

$$\phi(t) = \sqrt{2}\sum_i h[i]\phi(2t-i) \tag{3.8}$$

and

$$\psi(t) = \sqrt{2}\sum_i g[i]\phi(2t-i)$$

respectively, where $h[i]$ and $g[i]$ are called the recursion coefficients.

Daubechies and Lagarias [30, 31] made an extensive study of Eqn. (3.8), which they called a *lattice two-scale difference equation*. They proved the existence and uniqueness of L^1

solutions and that these are continuous and have compact support. They also found an iterative method of approximating solutions, versions of which we meet in the next two equations.

We can use the cascade algorithm (see *e.g.* [81, p. 231]) to find approximate point values of $\phi(t)$ and $\psi(t)$:

$$\phi^{k+1}(t) = \sqrt{2}\sum_{i=1}^{2n} s[i]\,\phi^{k}(2t-i) \tag{3.9}$$

and

$$\psi^{k+1}(t) = \sqrt{2}\sum_{i=1}^{2n} w[i]\,\phi^{k}(2t-i), \tag{3.10}$$

where $s[\cdot]$ and $w[\cdot]$ are the filter coefficients and $2n$ is the length of the filters in question. The results of iterating these functions are vectors of samples of $\Phi(t)$ and $\Psi(t)$ whose lengths depend on the number of iterations.

An example of an implementation of this algorithm in MATLAB® is at `http://people.sc.fsu.edu/~jburkardt/m_src/wavelet/cascade.m` (accessed 17th June 2019), although MATLAB® provides functions to generate vectors of real samples automatically.

Example 3.6.1. If we run the following code, we obtain the top left plot in Figure 3.3; changing the 2 in 'db2' appropriately gives the other plots. Note that we have adjusted the vertical scale in each case. In 'db$N$', N is the number of vanishing moments and $2N$ is the length of the filters.

```
[phi,psi,xval] = wavefun('db2');
plot(xval,phi,'LineWidth',1.5);
hold on
plot(xval,psi,'LineWidth',1.5,'Color','r');
hold on
yline(0);
hold off
axis square
```

```
ylim([-1.6,2])
```

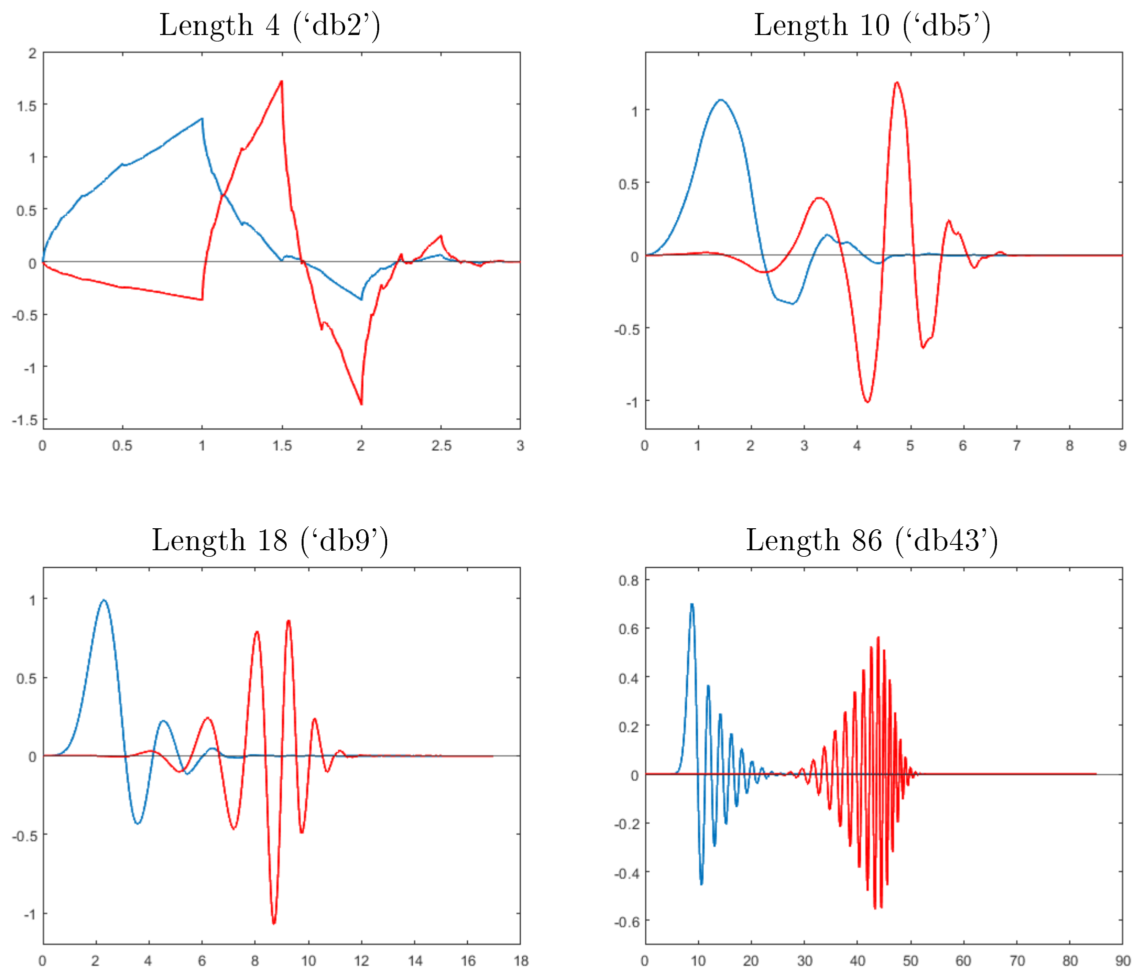


Figure 3.3: Scaling and wavelet functions using real Daubechies filters.

The square waves of the Haar scaling and wavelet functions described in Equations (3.5) and (3.4) respectively are clearly very different from the functions in Figure 3.3, although the Haar wavelet is actually a Daubechies wavelet, one with label db1. In the MATLAB® code above, the 2 in db2 may be replaced with any integer up to 43: we have used the examples of 2, 5, 9 and 43 although in theory it may be any positive integer N. As N increases, we can see the scaling and wavelet functions becoming smoother and developing more oscillations. We give four examples of frequency response plots for Daubechies scaling and wavelet filters in Figure 5.9, where it is apparent that as N increases, so does the flatness of the frequency response.

3.7 Chapter Summary

We gave a brief introduction to real wavelets and the important concept of multiresolution analysis (MRA). We showed how each level of an MRA may be performed with a quadrature mirror filter. We then went on to explain how Daubechies filter coefficients are calculated and gave an example. We finished by using the cascade algorithm on the filter we had just found, together with three others, and showed what some real scaling and wavelet functions look like.

A large body of work exists on the subject of real wavelet analysis: for further details we recommend the books by Strang and Nguyen [141] and Kovačević *et al.* [84].

Chapter 4

Prior Work on Quaternion Wavelets and their Transforms

In this chapter we review the literature on quaternion wavelets and quaternion wavelet transforms (QWTs). In our review article [44, Secs. 4–6], on which this chapter is based, we attempted to cite the majority of articles whose author(s) did anything at all with what they called a 'quaternion wavelet' or 'quaternion wavelet transform'. Since our review article was published, more has appeared on the subject, but very few recent articles report anything significant or novel. However, in order to be consistent with our review article, we have still tried to cite as many as possible.

4.1 Review of the Literature

In this section we review the literature on QWTs. The search terms used in Google Scholar were 'quaternion wavelet' and 'quaternionic wavelet'. There are a number of different definitions and we shall look in detail particularly at those from the most-cited papers per the Web of Science[1]: Bayro-Corrochano [11] with 31 citations as at 17th May 2015, Zhou *et al.* [169] with 25 and Chan *et al.* [24] with 51. As at 24th September 2018, the figures are 67, 28 and 72 respectively. These totals actually include articles where these references are mentioned only in passing, so we omit those in what follows. We could have also used search terms like 'vector wavelet' or 'hypercomplex wavelet', but the point with this document is to concentrate specifically on QWTs at this stage.

We have not split up this section into separate subsections for continuous and discrete QWTs and quaternion MRAs, but we have attempted to group papers together in a logical fashion, roughly chronologically.

4.1.1 Early Papers

Mitrea [110, ch. 2] introduced Clifford wavelets and a Clifford multiresolution analysis purely as an academic exercise: the Clifford Algebra $Cl(0,2)$ is isomorphic to the quaternions. As a specific example, he constructed theoretical Haar Clifford wavelets. This appears to be the earliest mention of any sort of hypercomplex wavelet and dates from 1994.

Traversoni [146] represented the four components of a quaternion as real wavelets and proposed using this formulation in Navier Stokes problems to express vorticity using all four dimensions. The energy would then be expressed as wavelets and low energy turbulence could be filtered out by suppressing the corresponding wavelets. This idea was not developed further and in [147], he used the ideas of Mitrea to obtain a quaternion multiresolution analysis and Haar quaternion wavelet, the latter based on a cubic representation of the Haar square pulse in 3-D. In [148], he applied his Haar quaternion wavelets to image analysis,

[1] http://apps.webofknowledge.com/UA_GeneralSearch_input.do?product=UA&search_mode=GeneralSearch&SID=T22VaVxxnQmgW1BUKFE&preferencesSaved=

creating a 3-D representation of a collection of 2-D tomography data.

Bülow [17] developed a quaternion Hilbert transform and used the polar representation of a quaternion in a quaternionic Gabor filter, on which a large body of research has been based. We review this research in Subsections 4.1.2, 4.1.3 and 4.1.4.

4.1.2 QWTs Based on a Quaternion Gabor Transform

Bayro-Corrochano [11] introduced a 2-D quaternion MRA, a ‘wavelet pyramid’, which is also covered in Moya-Sánchez and Bayro-Corrochano [112]:

$$f(x,y) = \boldsymbol{A}_n f(x,y) + \sum_{j=1}^{n} [\boldsymbol{D}_{j,1} f(x,y) + \boldsymbol{D}_{j,2} f(x,y) + \boldsymbol{D}_{j,3} f(x,y)],$$

where the Approximations $\boldsymbol{A}$ and detail $\boldsymbol{D}$ are defined as

$$\boldsymbol{A}_j f(x,y) = \sum_{k=-\infty}^{\infty} \sum_{\ell=-\infty}^{\infty} a_{j,k,\ell} \boldsymbol{\Phi}_{j,k,\ell}(x,y),$$

$$\boldsymbol{D}_{j,p} f(x,y) = \sum_{k=-\infty}^{\infty} \sum_{\ell=-\infty}^{\infty} d_{j,p,k,\ell} \boldsymbol{\Psi}_{j,p,k,\ell}(x,y)$$

with

$$\boldsymbol{\Phi}_{j,k,\ell}(x,y) = \frac{1}{2^j} \boldsymbol{\Phi} \left(\frac{x-k}{2^j}, \frac{y-\ell}{2^j} \right), \qquad (j,k,\ell) \in \mathbb{Z}^3,$$

$$\boldsymbol{\Psi}_{j,p,k,\ell}(x,y) = \frac{1}{2^j} \boldsymbol{\Psi} \left(\frac{x-k}{2^j}, \frac{y-\ell}{2^j} \right)$$

and

$$\begin{aligned} a_{j,k,\ell}(x,y) &= \langle f(x,y), \mathbf{\Phi}_{j,k,\ell}(x,y)\rangle, \\ d_{j,p,k,\ell} &= \langle f(x,y), \mathbf{\Psi}_{j,p,k,\ell}(x,y)\rangle. \end{aligned}$$

He also decomposed the scaling and wavelet functions into 1-D functions:

$$\begin{aligned} \mathbf{\Phi}(x,y) &= \phi(x)\phi(y), \\ \mathbf{\Psi}_1(x,y) &= \phi(x)\psi(y), \\ \mathbf{\Psi}_2(x,y) &= \psi(x)\phi(y), \\ \mathbf{\Psi}_3(x,y) &= \psi(y)\psi(x). \end{aligned}$$

The way this wavelet pyramid works is that given a 2-D image $f(x,y)$, for each level j we have four matrices $\boldsymbol{A}_j$, $\boldsymbol{D}_p$, $p = 1,2,3$ with elements $\boldsymbol{A}_j(k,\ell) = a_{j,k,\ell}$ and $\boldsymbol{D}_{j,p}(k,\ell) = d_{j,p,k,\ell}$.

For his QWT, Bayro-Corrochano [11] used two quaternionic modulated Gabor filters in quadrature:

$$\begin{aligned} h^q = \frac{1}{\sigma_h\sqrt{2\pi}} \exp\left(-\frac{x^2+(\varepsilon y)^2}{\sigma_h^2}\right) \exp\left(\mathbf{i}\frac{C_{hH}\,\omega_{hH}\,x}{\sigma_h}\right) \exp\left(\mathbf{j}\frac{C_{hV}\,\omega_{hV}\,\varepsilon y}{\sigma_h}\right) \\ = h^q_{ee} + h^q_{oe}\mathbf{i} + h^q_{eo}\mathbf{j} + h^q_{oo}\mathbf{k} \end{aligned} \tag{4.1}$$

and

$$\begin{aligned} g^q = \frac{1}{\sigma_g\sqrt{2\pi}} \exp\left(-\frac{x^2+(\varepsilon y)^2}{\sigma_g^2}\right) \exp\left(\mathbf{i}\frac{C_{gH}\,\omega_{gH}\,x}{\sigma_g}\right) \exp\left(\mathbf{j}\frac{C_{gV}\,\omega_{gV}\,\varepsilon y}{\sigma_g}\right) \\ = g^q_{ee} + g^q_{oe}\mathbf{i} + g^q_{eo}\mathbf{j} + g^q_{oo}\mathbf{k}, \end{aligned} \tag{4.2}$$

where ε is the aspect ratio, the σ's are standard deviations, the ω's are modulation frequencies, $C_{aB} = \omega_{aB}\,\sigma_a$ with $a = h$ or g for low- or high-pass and $B = H$ or V for horizontal or vertical, *ee* means even-even, *eo* means even-odd, *oe* means odd-even and *oo* means odd-odd. These last four come from the cosine and sine products of the equations' expansions. These filters are quaternionic versions of the low-pass filter H and high-pass filter G of figure 3.2. Bayro-Corrochano gives conditions on the frequencies as follows:

$$\omega_{hH} + \omega_{hV} = \pi, \qquad \omega_{hH} > \omega_{hV}, \qquad \omega_{hH} = 3\,\omega_{gH}, \qquad \omega_{hH} = \frac{5\pi}{6} \quad \text{and} \quad \omega_{gH} = \frac{\pi}{6}.$$

We have changed some of the notation to avoid using tildes to indicate different constants and to give more meaning to the subscripts.

This quaternion wavelet also appeared in [10] and a joint conference paper by Bayro-Corrochano and de La Torre Gomora [12] and while he did include Bülow [17] in these two articles' and his earlier article's references, he did not explicitly state that Equations (4.1) and (4.2) came directly from Bülow's Equation (3.40) for a quaternionic Gabor filter.

Despite all the citations, only a few researchers have actually done anything with this QWT. Naouai *et al.* [113] used it in the extraction of road map information from high resolution remotely sensed images: conventional methods for updating road maps rely on human intervention and are expensive and time consuming. Ding *et al.* [34] used it in the analysis of black and white images in preparation for colourisation. Wang *et al.* [155] used it in a watermarking algorithm applicable to colour images, but which used QWTs of greyscale images at one stage of the process. Han *et al.* [71] also used it in a watermarking algorithm, but they found the QWT of the luminance channel of an image to give four approximation subimages and then added a watermark to two of them at random, using a discrete cosine transform.

4.1.3 QWTs for Phase-Based Stereo Matching

Xu *et al.* [160] worked on phase-based stereo matching for uncalibrated images. They revised and expanded their work in [161], [149] and [169], extending the work of Bayro-Corrochano to symmetric/asymmetric biorthogonal wavelet bases in order to build linear-phase quaternion wavelet filters (LPQWFs). Given the scale function $\phi_{\ell,m}(x)$ and the wavelet base $\psi_{\ell,m}(x)$ of a biorthogonal wavelet where ℓ denotes the scale factors 2^ℓ, $\ell = 0, 1, \ldots, K$ and $m \in \mathbb{Z}$, the discrete positions, four corresponding 1-D analytic wavelets $\psi^H_{\ell,m}(x)$, $\psi^V_{\ell,m}(y)$, $\phi^H_{\ell,m}(x)$ and $\phi^V_{\ell,m}(y)$ can be built as:

$$\begin{aligned}
\psi^H_{\ell,m}(x) &= \psi_{\ell,m}(x) + \mathbf{i}\mathcal{H}_x(\psi_{\ell,m}(x)) \\
\psi^V_{\ell,m}(y) &= \psi_{\ell,m}(y) + \mathbf{j}\mathcal{H}_y(\psi_{\ell,m}(y)) \\
\phi^H_{\ell,m}(x) &= \phi_{\ell,m}(x) + \mathbf{i}\mathcal{H}_x(\phi_{\ell,m}(x)) \\
\phi^V_{\ell,m}(y) &= \phi_{\ell,m}(y) + \mathbf{j}\mathcal{H}_y(\phi_{\ell,m}(y)),
\end{aligned}$$

where $\mathcal{H}_x(\cdot)$ and $\mathcal{H}_y(\cdot)$ denote the partial Hilbert transforms along the x-axis and y-axis and H and V refer to horizontal and vertical respectively.

The 2-D scale function $\Phi^q(x, y)$ and its three associated quaternion wavelet functions $\Psi^q_H(x, y)$, $\Psi^q_V(x, y)$ and $\Psi^q_D(x, y)$ can then be built as:

$$\begin{aligned}
\Phi^q(x, y) &= \phi_{\ell,m}(x)\phi_{\ell,m}(y) \\
\Psi^q_H(x, y) &= \phi^H_{\ell,m}(x)\psi^V_{\ell,m}(y) \\
\Psi^q_V(x, y) &= \psi^H_{\ell,m}(x)\phi^V_{\ell,m}(y) \\
\Psi^q_D(x, y) &= \psi^H_{\ell,m}(x)\psi^V_{\ell,m}(y),
\end{aligned}$$

where D refers to diagonal. $\Phi^q(x, y)$ is a real 2-D scale function and expanding the three wavelet equations leads to 12 products that can be written succinctly as:

$$\Psi_P^q(x,y) = \psi^P(x,y) + \mathbf{i}\mathcal{H}_x(\psi^P(x,y)) + \mathbf{j}\mathcal{H}_y(\psi^P(x,y)) + \mathbf{k}\mathcal{H}_{xy}(\psi^P(x,y)), \qquad P \in \{H,V,D\},$$

where $\mathcal{H}_{xy}(\cdot)$ denotes the total Hilbert transform and $\psi^P(x,y)$, $P \in \{H,V,D\}$ is a general 2-D real wavelet with different orientations. Thus each quaternion wavelet consists of a Hilbert quadruple and is suited to the construction of 2-D analytic signals.

Zhou *et al.* developed a practical technique for constructing their LPQWFs using real biorthogonal bases. They used a cost function approach, minimisation of which avoided false matches in their stereo matching algorithm.

Despite the relatively high number of citations mentioned in the introduction, all the papers citing Zhou *et al.* [169] appear to report something other than any extension to or application of their QWT.

4.1.4 The Dual Tree Quaternion Wavelet Transform

The dual tree QWT was introduced in 2004 by Chan *et al.* [21]. It went through several refinements in [20], [22] and [23], culminating in 2008 with [24]. In Table 4.1, we analyse by application all the articles up to 2018 that make use of this version of the QWT. Chan *et al.* themselves used it for edge geometry estimation and image disparity estimation and have made their MATLAB® code for the latter, including that for their QWT, available on the website of the Digital Signal Processing Group at Rice University.[2] This QWT is based on the 2-D analytic signal of Bülow [17]:

[2]https://web.archive.org/web/20160512013513/http://dsp.rice.edu:80/software/qwt, accessed 24th September 2018

Definition 4.1.1. Let f be a real-valued 2-D signal. the 2-D *quaternion analytic signal* is defined as

$$f_A^q(x,y) = f(x,y) + \mathbf{i} f_{\mathcal{H}i_1}(x,y) + \mathbf{j} f_{\mathcal{H}i_2}(x,y) + \mathbf{k} f_{\mathcal{H}i}(x,y),$$

where

$$f_{\mathcal{H}i_1}(x,y) = f(x,y) ** \frac{\delta(y)}{\pi x},\ f_{\mathcal{H}i_2}(x,y) = f(x,y) ** \frac{\delta(x)}{\pi y} \quad \text{and} \quad f_{\mathcal{H}i}(x,y) = f(x,y) ** \frac{1}{\pi^2 xy},$$

with $\mathcal{H}i_1(\cdot)$ and $\mathcal{H}i_2(\cdot)$ being the partial Hilbert transforms and $\mathcal{H}i(\cdot)$, the total Hilbert transform; $\delta(x)$ and $\delta(y)$ are impulse sheets along the y and x axes respectively and $**$ denotes 2-D convolution.

Per Bow [15, pp. 422/3], an impulse sheet is defined to be infinite in one direction and to have as its cross-section, the usual 1-D δ-function.

This QWT is constructed by simply arranging the four components of a 2-D complex wavelet as a quaternion, using appropriate filters for the calculation of the coefficients. The basis functions are shifted and scaled copies of the following; the superscripts H, V and D label the horizontal, vertical and diagonal sub-bands respectively.

$$\begin{aligned}
f_{A_0}^q(x,y) &= \quad \phi(x,y) = \phi_h(x)\phi_h(y) + \mathbf{i}\phi_h(x)\phi_g(y) + \mathbf{j}\phi_g(x)\phi_h(y) + \mathbf{k}\phi_g(x)\phi_g(y)\\
f_{A_1}^q(x,y) &= \psi^H(x,y) = \psi_h(x)\phi_h(y) + \mathbf{i}\psi_h(x)\phi_g(y) + \mathbf{j}\psi_g(x)\phi_h(y) + \mathbf{k}\psi_g(x)\phi_g(y)\\
f_{A_2}^q(x,y) &= \psi^V(x,y) = \phi_h(x)\psi_h(y) + \mathbf{i}\phi_h(x)\psi_g(y) + \mathbf{j}\phi_g(x)\psi_h(y) + \mathbf{k}\phi_g(x)\psi_g(y)\\
f_{A_3}^q(x,y) &= \psi^D(x,y) = \psi_h(x)\psi_h(y) + \mathbf{i}\psi_h(x)\psi_g(y) + \mathbf{j}\psi_g(x)\psi_h(y) + \mathbf{k}\psi_g(x)\psi_g(y)
\end{aligned} \tag{4.3}$$

Chan *et al.*'s QWT function has four arguments: the image to be analysed, the level of analysis required J and four pairs of filter coefficients, two pairs `Faf{1} Lo` and `Hi` and `Faf{2} Lo` and `Hi` for the first level and two pairs `af{1} Lo` and `Hi` and `af{2} Lo` and `Hi` for subsequent levels. It returns the wavelet coefficients for each of levels 1 to J and the scaling coefficients at level $J + 1$.

Table 4.1: Summary of research by application which has used the QWT formulation of Chan *et al.* [24].

Application	**Research articles**
Image disparity and optical flow	Chan *et al.* [24], Wang *et al.* [153, 154], Kumar *et al.* [85], Liu *et al.* [97]
Edge geometry	Chan *et al.* [24]
Texture recognition	Soulard and Carré [137, 139], Sathyabama *et al.* [132], Gai *et al.* [56], Li *et al.* [93]
Image coding	Soulard and Carré [138], Khelifi *et al.* [83], Madhu and Anant Shankar [106]
Feature extraction and object recognition	Gai *et al.* [52, 53, 54, 55], Li *et al.* [94], Greenblatt *et al.* [66], Priyadharshini and Arivazhagan [121], Sangeetha *et al.* [123], Katunin [79], Shen *et al.* [133], Mosquera-Lopez *et al.* [111], Gai and Luo [50], Gai [49], Agaian *et al.* [1], Wang *et al.* [156], Arivazhagan *et al.* [3], Ren *et al.* [122]
Speckle reduction	Jin *et al.* [76], Liu *et al.* [99], Wu *et al.* [158]
Image denoising	Yin *et al.* [162], Kadiri *et al.* [77, 78], Gai *et al.* [57], Yu *et al.* [164], Gai and Luo [51], Zhang [165], Liu *et al.* [104], Malleswari and Madhu [108], Umam and Yunus [150]
Image fusion	Liu *et al.* [100, 102, 103], Yin *et al.* [163], Geng *et al.* [59, 60], Zheng *et al.* [168], Chai *et al.* [19]
Scale saliency	Le Ngo *et al.* [88, 89]
Image quality metrics	Liu *et al.* [101], Traoré *et al.* [144, 145], Liu and Du [98], Tang *et al.* [143]
Watermarking	Lei *et al.* [90, 91], Han *et al.* [71, 72]
Image sharpening	Kumar *et al.* [86]

The way this function works is to use one filter on each column and downsample by two and then use another filter on each row and downsample by two a second time. For the four scalar terms in equations (4.3), `Faf{1}` (`af{1}` from level 2 onwards) filters are used in the order `Lo-Lo`, `Lo-Hi`, `Hi-Lo`, `Hi-Hi`. The **i**-terms use `Faf{1}` and `Faf{2}` (`af{1}` and `af{2}` from level 2 onwards) respectively in the order $\texttt{Lo}_1$-$\texttt{Lo}_2$, $\texttt{Lo}_1$-$\texttt{Hi}_2$, $\texttt{Hi}_1$-$\texttt{Lo}_2$, $\texttt{Hi}_1$-$\texttt{Hi}_2$, the subscript indicating the source of the filter. The **j**-terms use `Faf{2}` and `Faf{1}` (`af{2}` and `af{1}` from level 2 onwards) respectively in the order $\texttt{Lo}_2$-$\texttt{Lo}_1$, $\texttt{Lo}_2$-$\texttt{Hi}_1$, $\texttt{Hi}_2$-$\texttt{Lo}_1$, $\texttt{Hi}_2$-$\texttt{Hi}_1$. Finally, the **k**-terms use `Faf{2}` (`af{2}` from level 2 onwards) respectively in the order `Lo-Lo`, `Lo-Hi`, `Hi-Lo`, `Hi-Hi`.

Chan *et al.* do not actually mention anything in their articles about the filters they used, although they do say in the comments on their code that the `Faf` filters are "Farras filters organized for the dual-tree complex DWT" and that the `af` filters are "Kingsbury Q-shift filters for the dual-tree complex DWT". Each one consists of 10 coefficients, including a few zeros.

4.1.5 Other Novel QWTs

Peng and Zhao [118] succeeded in designing three symmetric quaternion scaling filters but as shown in Ginzberg [61, pp. 130/1], they were in fact trivial.

Shi [135, pp. 49-51] suggested a 'quaternion' Haar wavelet transform, starting from a real Haar wavelet transform in one dimension:

$$\begin{aligned} l * X(r) &= \frac{X(r) + X(r-1)}{2} \quad \mod N \\ h * X(r) &= \frac{X(r) - X(r-1)}{2} \quad \mod N, \end{aligned}$$

where l and h are low- and high-pass filters respectively and $X(r)$ is some 1-D signal of length N. If C_l and C_h are the above results downsampled by 2, then

$$X = (C_l + C_h) \oplus (C_l - C_h).$$

Then doing the above convolutions along the rows and columns of a colour image would give a QWT, which could easily be used to reconstruct the original signal. As Shi admits, this would be equivalent to three separate monochrome QWTs and therefore not truly quaternionic.

Carré and Denis [18] seem to have been the first to think of trying to develop a QWT to use for processing colour images represented as pure quaternions. We explain this representation further in Subsection 4.2.4. They considered the two-channel filter bank with perfect reconstruction in Figure 3.2, but with quaternion input and output and quaternion coefficients. They found that the conditions on the filters for perfect reconstruction are exactly the same as for filter banks with real coefficients, except that the order of the elements is important. The only example they tried was a Shannon QWT, for which the filter coefficients turned out to be real. This was a conference paper and the follow-up peer-reviewed paper did not mention the filter banks.

Zhao and Peng [167] define a continuous quaternion wavelet $\psi \in L^2(\mathbb{R}, \mathbb{H})$ as

$$\psi_{a,\theta,\boldsymbol{b}}(\boldsymbol{x}) = a^{-1}\psi\left(a^{-1}r_{-\theta}(\boldsymbol{x}-\boldsymbol{b})\right)$$

where $$r_{-\theta}(\boldsymbol{x}) = \left(x_1\cos(\theta) - x_2\sin(\theta), x_1\sin(\theta) + x_2\cos(\theta)\right),$$

$$0 \leqslant \theta \leqslant 2\pi, \quad a > 0, \quad \boldsymbol{b} \in \mathbb{R}^2.$$

They then define their QWT as

$$W_\psi : L^2\left(\mathbb{R}^2, \mathbb{H}\right) \to L^2\left(IG(2), \mathbb{H}, a^{-3}\mathrm{d}a\mathrm{d}\theta\mathrm{d}\boldsymbol{b}\right)$$

$$\begin{aligned} f(\boldsymbol{x}) \mapsto W_\psi(a,\theta,\boldsymbol{b}) &= C_\psi^{-\frac{1}{2}} \langle \psi_{a,\theta,\boldsymbol{b}}, f\rangle \\ &= C_\psi^{-\frac{1}{2}} \int_{\mathbb{R}^2} \overline{\psi_{a,\theta,\boldsymbol{b}}(\boldsymbol{x})} f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}, \end{aligned} \tag{4.4}$$

where $\quad IG(2) = \left\{(a, r_\theta, \boldsymbol{b}), a > 0, \theta \in [0, 2\pi], \boldsymbol{b} \in \mathbb{R}^2\right\}\quad$ and $\quad C_\psi = \int_{\mathbb{R}^2} \frac{|\hat{\psi}(\xi)|^2}{|\xi|^2} \mathrm{d}\xi.$

Thus the signal is decomposed by rotated as well as scaled and translated copies of the mother wavelet. This QWT apparently first appeared in 2001 in Birkbeck College Mathematical Research Unit's Journal of Natural Geometry.

Bahri *et al.* [7] define their 2-D continuous QWT as:

$$T_\psi : L^2\left(\mathbb{R}^2, \mathbb{H}\right) \to L^2\left(\mathbb{R}^2, \mathbb{H}\right)$$

$$\begin{aligned} f \mapsto T_\psi f(a,\theta,\boldsymbol{b}) &= \langle f, \psi_{a,\theta,\boldsymbol{b}}\rangle_{L^2(\mathbb{R}^2,\mathbb{H})} \\ &= \int_{\mathbb{R}^2} f(\boldsymbol{x}) \frac{1}{a} \overline{\psi\left(r_{-\theta}\left(\frac{\boldsymbol{x}-\boldsymbol{b}}{a}\right)\right)} \mathrm{d}\boldsymbol{x}, \end{aligned} \tag{4.5}$$

where

$$r_{-\theta}(\boldsymbol{z}) = \left(z_1 \cos(\theta) + z_2 \sin(\theta), -z_1 \sin(\theta) + z_2 \cos(\theta)\right), \qquad 0 \leqslant \theta \leqslant 2\pi.$$

This QWT is very similar to Zhao and Peng's in equation (4.4) and although Bahri *et al.* do not cite Zhao and Peng [167], they do cite the earlier paper mentioned below this equation in passing. It also appears in Bahri [6], Bahri *et al.* [8] and [9]. In the last of these, the authors define a quaternion Fourier transform as

$$\mathcal{F}_q[f(\boldsymbol{x})](\boldsymbol{\omega}) = \hat{f}(\boldsymbol{\omega}) = \int_{\mathbb{R}^2} f(\boldsymbol{x}) \exp(-\boldsymbol{\mu}[\boldsymbol{\omega} \cdot \boldsymbol{x}]) \mathrm{d}\boldsymbol{x},$$

where $\boldsymbol{\mu} = \dfrac{\mathbf{i}+\mathbf{j}+\mathbf{k}}{\sqrt{3}}$. They then use this to rewrite equation (4.5) as

$$T_\psi f(a,\theta,\boldsymbol{b}) = \frac{1}{(2\pi)^2}\int_{\mathbb{R}^2} a\hat{f}(\boldsymbol{\omega})\exp(-\boldsymbol{\mu}[\boldsymbol{\omega}\bullet\boldsymbol{b}])\overline{\hat{\psi}(ar_{-\theta}(\boldsymbol{\omega}))}\mathrm{d}\boldsymbol{\omega}.$$

They go on to establish some theorems involving this formulation of their 2-D QWT.

Guo *et al.* [68] define their quaternion curvelet transform in virtually the same way as Bahri *et al.* define their 2-D QWT and use it in colour image fusion. They recognise that treating the R, G and B channels holistically as described in Subsection 4.2.4 reduces blur, preserving the greatest amount of colour information, and use colour images represented as pure quaternions but do not describe exactly how they evaluate their transform. However, Pang *et al.* [117], which has two authors in common with Guo *et al.* [68], do explain how their QWTs are computed. They use low- and high-pass decomposition filters ϕ_d and φ_d and low- and high-pass reconstruction filters ϕ_r and φ_r, which they define as:

$$\begin{aligned}
\phi_d &= \{0.0000 \quad -0.1768 \quad 0.3536 \quad 1.0607 \quad 0.3536 \quad -0.1768\}\exp(\boldsymbol{\mu}\pi/4),\\
\varphi_d &= \{0.0000 \quad 0.3536 \quad -0.7071 \quad 0.3536 \quad 0.0000 \quad 0.0000\}\exp(\boldsymbol{\mu}\pi/4),\\
\phi_r &= \{0.0000 \quad 0.3536 \quad 0.7071 \quad 0.3536 \quad 0.0000 \quad 0.0000\}\exp(\boldsymbol{\mu}\pi/4),\\
\varphi_r &= \{0.0000 \quad 0.1768 \quad 0.3536 \quad -1.0607 \quad 0.3536 \quad 0.1768\}\exp(\boldsymbol{\mu}\pi/4),
\end{aligned}$$

where $\boldsymbol{\mu} = (\mathbf{i}+\mathbf{j}+\mathbf{k})/\sqrt{2}$, although presumably $\sqrt{2}$ should read $\sqrt{3}$. Guo *et al.* do not explain where these filter coefficients come from. They report an improvement on previous methods of colour image fusion.

Hogan and Morris [75] developed some theory for quaternionic signals using the Clifford-Fourier transform of Brackx *et al.* [16] for $C\ell(0,2)$, which is a quaternion Fourier transform (QFT). They used this QFT in the proof of a quaternionic analogue of the QMF condition: a quaternionic orthonormal scaling function must necessarily satisfy this. They went on to find conditions to be satisfied by the corresponding quaternion wavelet functions. These conditions can be expressed as a matrix equation and they found an equivalent system of quadratic equations that it would be possible to solve numerically. They then found conditions that would be sufficient to guarantee a quaternion scaling function would have compact support. They were not, however, able to actually construct a quaternion wavelet basis from a QFT series and conclude that their theory must be incomplete. They were, however, able to construct a quaternionic biorthogonal wavelet basis and give an example of such a wavelet basis and illustrate the resulting wavelets.

Xia and Suter [159] introduced vector-valued wavelets for the analysis of vector-valued signals. Their MRA is similar to that in Subsection 3.3, except that $L^2(\mathbb{R})$ is replaced by $L^2(\mathbb{R}, \mathbb{C}^{N\times N})$. Matrix valued wavelets (MVWs), as they are most commonly called, have since been studied in their own right. He and Yu [73] appear to have been the first to consider quaternion MVWs and an associated quaternion-valued MRA analysis, using the 2×2 complex matrix representation of quaternions. However, they tried to design filters in the frequency domain and forgot the noncommutativity of matrices, resulting in their method only working for trivial scaling and wavelet filters. Bahri [5] did something very similar to He and Yu with 2×2 complex matrices and did not notice a similar noncommutativity problem. Ginzberg and Walden [62] constructed some novel families of non-trivial 2×2 and 4×4 MVWs. As per Ginzberg [39, p. 122], "We define an $n \times n$ MVW to be trivial if it can be decomposed into independent lower-dimensional MVWs (in some appropriate orthogonal basis of $\mathbb{R}^n$). Every MVW is then composed of one or more non-trivial MVWs". They went on to construct a 4×4 non-trivial symmetric quaternion wavelet with compact support, specifically a length 10 Daubechies quaternion scaling filter together with a corresponding wavelet filter. Daubechies wavelets are characterised by maximal vanishing moments for a given length of filter: the low- and high-frequency passbands can be made as flat as one wishes by increasing the lengths of the filters. We postpone full discussion of these filters to

Chapter 5.

Augereau and Carré [4] found a 'hypercomplex polynomial wavelet-filter bank transform'. They began by defining a Quaternion Bivariate Polynomial (QBP) of $\boldsymbol{x} = (x_1, x_2) \in \mathbb{R}^2$, of degree $d = \max(d_1 + d_2) \in \mathbb{Z}_0^+$, as

$$P(\boldsymbol{x}) = \sum_{\substack{(d_1,d_2)\in[0;d]^2 \\ d_1+d_2\leq d}} q_{d_1,d_2} x_1^{d_1} x_2^{d_2},$$

where $q_{d_1,d_2} \in \mathbb{H}$. They then took $D = \{(d_1, d_2)\} = \{0, 1, \ldots, N_1\} \times \{0, 1, \ldots, N_2\}$ to be a finite set of pairs of integers and let $\mathbb{E}_D$ be the set of QBPs such that $q_{e_1,e_2} \equiv 0$ if $(e_1, e_2) \notin D$. Thus they had a finite set of QBPs and using these, they went on to use the Gram-Schmidt process to construct a discrete orthonormal basis with respect to a non-zero reference quaternion $\mu_{d_1,d_2,\boldsymbol{x}}$. With appropriate combinations of elements of $\mathbb{E}_D$, they were then able to approximate any bivariate quaternion function, *e.g.* a colour vector image, with respect to $\mu_{d_1,d_2,\boldsymbol{x}}$. This reference direction would commonly be the grey axis, but it need not be. They showed that their polynomial filters gave perfect reconstruction; and since they are normalised, they would have a squared norm of 1. They do say that their filters can be symmetric or antisymmetric, but only the latter would guarantee a mean of zero. Thus they are not exactly quaternion wavelets, which is something they do not claim anyway. However, they do describe their filters as being wavelet-like and because they were quaternion-valued, they deserve a mention here.

4.2 Discussion

We mentioned one piece of research dealing with Clifford wavelets, that of Mitrea [110]. As with this, the dozen or so other articles on Clifford wavelets (which we do not cite), as opposed to quaternion wavelets, all deal just with the continuous case and offer little insight into exactly how to construct practical (discrete) Clifford wavelets.

We discussed several possible approaches to wavelet transforms for quaternionic, (*i.e.* not real-valued or general Clifford-valued) signals, and we now consider the ramifications of these ideas and how they relate to each other. We also consider how classical filter theory can be extended to complex and vector-valued wavelets and this leads us to the question of what makes a truly quaternionic wavelet.

4.2.1 Short Time Fourier Transform Approaches

The STFT approach developed from Gabor's ideas is a form of wavelet transform, but it is based on Fourier transforms, which use sinusoidal basis functions. This type of wavelet transform is not based on basis functions with finite support. As we have seen, there are wavelet transforms based on wavelets which do have finite support and we make a distinction between the two in what follows. It is not surprising that the STFT-based approach to defining wavelet transforms has been adopted by many researchers to provide quaternion STFT/wavelet transforms. The Fourier transforms needed already existed (including numerical implementations) and the use of Gaussian or other window functions presented no problems because they are real-valued.

Extending the theory for a complex wavelet transform to a quaternion wavelet transform seems an obvious thing to try to do, but in the same way that the dual-tree complex wavelet transform has little to do with complex numbers, so the dual-tree QWT of Chan *et al.* [24] has little to do with quaternions as we saw. However, due to the apparent fixation a great many researchers have on this particular formulation, the name 'quaternion wavelet transform' will very likely remain attached to it.

4.2.2 Generalising Classical Wavelets

Turning to the more difficult problem of generalising classical wavelet transforms based on wavelets with finite support (*e.g.* Haar or Daubechies), there are significant research questions still to be studied. This can be seen even in the complex case. Putting aside the dual-tree approaches to wavelets as not being truly complex, let us consider the complex case in some detail, partly for its own sake and partly because it provides a simpler model to work with initially than the quaternions. Consider a complex signal and a truly complex discrete wavelet transform. Referring to Figure 3.2, an immediate problem is how to define analysis and synthesis filters which process complex signals (and which therefore presumably must have complex filter coefficients). We do not know of any significant theoretical work done on the concept of frequency response for such filters, nor on the frequency content of a signal. As an example of the problems here, consider how the classical Fourier transform represents a real signal using positive and negative frequency complex exponentials, whose imaginary parts cancel out. What is a negative frequency? In the real case it appears to be a mathematical artefact due to the use of complex exponentials, but in the complex case it is simply explained: a negative frequency exponential rotates in the complex plane in the negative sense (that is clockwise, by mathematical convention), whereas a positive frequency exponential rotates in the positive sense. Now, a complex signal can clearly have both positive and negative frequency content (being composed of exponentials rotating in either sense) and a Fourier transform of a complex signal will, in general, have no conjugate symmetry, unlike the real case. We believe that the concept of frequency response needs to represent the response of a filter to positive and negative frequencies independently.

4.2.3 Filter Theory, *etc.*

Noting that there is a gap in knowledge in the area of complex filter theory, we then note that the same gap occurs with vector-valued signals, whether represented by quaternions or by vectors in the linear algebra sense or by elements of other hypercomplex algebras. We know that oscillation at a single frequency in a vector-valued signal is confined to a planar ellipse, regardless of the dimensionality of the vector space [127], but we do not know how to design filters to handle such signals, even for such apparently simple tasks as separation of oscillations in different planes or in different senses of rotation (polarization). The problem becomes even harder if we consider a modulated ellipse, as described by Lilly [95]. We have seen in Subsection 4.1.5 that Carré and Denis attempted to generalise the classical QMF representation of a DWT to a QWT by using filter banks with quaternion-valued coefficients, but did not actually demonstrate a filter with quaternion coefficients. Ginzberg and Walden overcame the problem by using the matrix representation of quaternions. Augereau and Carré took an entirely new approach and it will be interesting to see how this develops in the future.

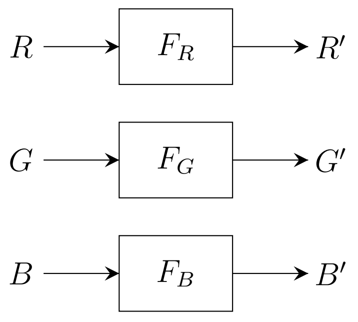
4.2.4 What Makes a QWT Truly Quaternionic?

The majority of the QWTs we have looked at could have been implemented using DWTs or $\mathbb{C}$WTs without reference to quaternions, so should these ones really be called QWTs? Alfsmann *et al.* [2] give the main reason for the use of hypercomplex algebras in signal processing:

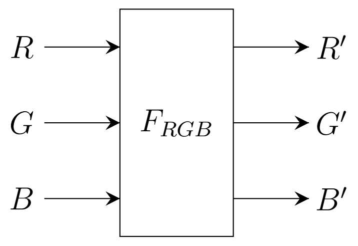
> "The holistic, compact processing of vector-valued signals that are a function of one or more independent parameters (*e.g.*, time, location, physical quantities). Here, the dimension of the algebra must be chosen in compliance with the dimension of the signal vector. This means that each vector-sample is treated as a whole rather than treating its components separately. Classically, the reason for this is that *the sample as a whole conveys information (direction in vector space) that is lost if the components of the sample are processed independently.*" (our emphasis).

We contend that the 'quaternion' in 'quaternion wavelet' and 'quaternion wavelet transform' should refer to a (pure or full) quaternion-valued signal. In colour image processing this would mean that the proportions of the three primary colours of each pixel would be the factors multiplying a quaternion's three imaginary parts. In RGB colour space, these would be the co-ordinates of the end of each pixel's colour vector. This representation would thus not lose the potentially useful information that might be contained in the correlation between different primary colours in an individual pixel.

Instead of treating each primary colour separately as illustrated diagrammatically in Figure 4.1a, a *true* QWT should treat each pixel holistically rather as in Figure 4.1b, where each of R', G' and B' depends on all of R, G and B. This is what the quaternion Fourier transform of Ell and Sangwine [36] achieves.



(a) Colour image processing with each primary colour processed separately.



(b) Processing of colour images represented as quaternions.

Figure 4.1: Diagrams illustrating the essential difference between 'conventional' colour image processing and with quaternion or vector images.

4.3 Chapter Summary

We saw how QWTs came about and how few researchers look beyond the dual-tree formulation of Chan *et al.* [24], which is actually closer to a $\mathbb{C}$WT than a QWT. However, once the term 'QWT' had been invented, it was only a matter of time before someone realised that it was a misnomer. In recent years a few researchers have envisioned the development of a 'true' QWT, one that was not equivalent to several separate DWTs or $\mathbb{C}$WTs in parallel. In particular, Carré and Denis [18] apparently understood the problem, but their QWT had real filter coefficients and so was equivalent to three separate DWTs; Hogan and Morris [75] had some success with implementing a biorthogonal QWT; and Ginzberg and Walden [62] and Ginzberg [61] introduced 'true' quaternion wavelet filters. It remains to be seen what will become of Augereau and Carré's approach [4].

Our way ahead in Chapter 5 will be to study and develop Ginzberg's methods further.

Chapter 5

Hypercomplex Scaling and Wavelet Filters and Functions

In this chapter we extend Paul Ginzberg's methods, [62] and [61, ch. 5], which we mentioned in Subsection 4.1.5, and find more discrete scaling and wavelet filters with quaternion coefficients and the first with Clifford $Cl(1,0)$, $Cl(1,1)$ and $Cl(2,0)$ coefficients.

In Section 5.1 we give Ginzberg's matrix equations, which we solve to find our scaling filters. We also explain how we find associated wavelet filters. In Section 5.2, we list some solutions and in Section 5.3, we give some illustrations of functions calculated from the filter coefficients. We find the frequency responses of our filters in Section 5.4, which show that the filters we find are truly hypercomplex scaling and wavelet filters, and we discuss our results in Section 5.5.

5.1 Ginzberg's Equations

In this section we present the equations on which this whole thesis rests. To begin with, however, we say a few words about our notation. There is no single accepted notation for all scaling and wavelet filters and their coefficients, for example Keinert [81] uses h_k and g_k respectively and also s_k and d_k, Kovačević *et al.* [84] use g_k and h_k and Strang and Nguyen [141] use $h_0(k)$ and $h_1(k)$. Rather than adopt one of these, we shall use the more intuitive $\boldsymbol{S}_k$ and $\boldsymbol{W}_k$ for matrix-valued scaling and wavelet coefficients, $\boldsymbol{s}$ and $\boldsymbol{w}$ for quaternion and Clifford scaling and wavelet filters with $s[1], s[2], s[3], \ldots$ and $w[1], w[2], w[3], \ldots$ for the actual filter coefficients. For quaternion and Clifford scaling and wavelet functions we shall use the conventional $\Phi(t)$ and $\Psi(t)$, which we define iteratively in Equations (3.9) and (3.10) respectively.

5.1.1 Scaling Filters

Ginzberg [61, ch. 5] was the first to bring the following equations together and to solve them for matrix representations of quaternions and thus find a discrete scaling filter with quaternion coefficients:

Omnidirectional balancing

$$\sum_{k=0}^{L-1} \mathbf{S}_k = \sqrt{2}\,\mathbf{I}_n \tag{5.1}$$

$L/2$ vanishing moments

$$\sum_{k=0}^{L-1} (-1)^k k^d \mathbf{S}_k = \mathbf{0}_n \qquad d = 0, 1, \ldots, L/2 \tag{5.2}$$

Orthogonality

$$\sum_{k=0}^{L-1-2m} \mathbf{S}_k \mathbf{S}_{k+2m}^T = \delta_{0,m} \mathbf{I}_n \qquad m = 0, 1, \ldots, L/2, \tag{5.3}$$

where the $\mathbf{S}_k$ are 4×4 real matrices with their elements arranged so that they are isomorphic to quaternions, $n = 4$ so that $\mathbf{I}_n$ and $\mathbf{0}_n$ are the 4×4 identity and zero matrices respectively and $\delta_{0,m}$ is the Kronecker delta. Each $\mathbf{S}_k$ was the sum of the four matrices in Equations (2.5), in turn multiplied by coefficients whose four numerical values Ginzberg found for each $\mathbf{S}_k$. He had $L = 10$ and thus found 40 real numbers, being the coefficients of s, $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$ in ten quaternions. Each quaternion was then a scaling filter coefficient.

Ginzberg [61, Sec. 5.4] explains how Equations (5.1) and (5.3), Ginzberg's (5.10) and (5.11) respectively, are derived, although he was not the first to do this: *e.g.* [152] has these as Equations (2.5) and (2.3) respectively. Equation (5.2), Ginzberg's (5.12), comes from [141, Sec. 7.1].

5.1.2 A Simplification for Quaternion Scaling Filters

Ginzberg found a way to reduce the number of real unknowns in his equations by three: this is not insignificant, as we found when we came to solve the equations. To do this, he used orthogonal similarity [61, pp. 145/6]: two matrix filters $\{\mathbf{G}_k\}$ and $\{\mathbf{J}_k\}$ are orthogonally similar if $\mathbf{G}_k = \mathbf{O}\mathbf{J}_k\mathbf{O}^T, \ \forall k \in \mathbb{Z}$, with $\mathbf{O}$ being an orthogonal matrix.

We do not actually need $\mathbf{O}$ to find orthogonally similar filters. Consider two vectors in 3-D space, neither of them along any of the axes and with a fixed non-zero angle $\neq \pi$ between them. We may rotate the plane formed by these two vectors so that it coincides with the x-z plane. The y co-ordinates of both vectors will be reduced to zero. In the x-z plane, we may then rotate the two vectors about the y-axis so that one of them coincides with the x-axis. The z co-ordinate of this one will also be reduced to zero. In other words, if (b, c, d) and (f, g, h) are our two vectors before rotating, our rotation results in $(b, c, d) \mapsto (b', 0, d')$ and $(f, g, h) \mapsto (f', 0, 0)$. Now suppose that our 3-D space is embedded in a 4-D space, so that our 3-D vectors can be considered the vector parts of full quaternions with non-zero scalars a and e respectively. Then we can write the rotations we described as

$$(a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}) \mapsto (a + b'\mathbf{i} + d'\mathbf{k}) \quad \text{and} \quad (e + f\mathbf{i} + g\mathbf{j} + h\mathbf{k}) \mapsto (e + f'\mathbf{i}).$$

Note that a and e are not changed.

What this means is that in Ginzberg's equations, we may assume that the coefficient of $\mathbf{j}$ in one quaternion and those of $\mathbf{j}$ and $\mathbf{k}$ in a second quaternion, in their real matrix representations, are zero. This small change makes the equations much easier to solve. After finding one solution, other orthogonally similar filters could be found by rotating all the quaternion filter coefficients in 4-D space (strictly any 3-D subset) in exactly the same manner, but we shall not do this.

5.1.3 Methods of Solution for Matrix-Valued Scaling Filters

In Maple™, we first assign all the $\mathbf{S}_k$ to different symbolic matrices, each one being the sum of the four matrices in Equation (2.5), in turn multiplied by symbols for which we want to find the numerical values. We find all the matrix sums in Equations (5.1) and (5.2) and sums of products in Equations (5.3); the total number depends on the length of filter we are trying to find. We then take all the polynomials in the first columns of the sums and solve them simultaneously. The linear equations are straightforward, because we can eliminate one unknown per equation by simply expressing it in terms of other unknowns: the results are carried forward to subsequent equations. Complications arise with the nonlinear equations. Ginzberg found the Gröbner bases, see *e.g.* Fröberg [46], using Maple™ functions, and was able to solve the resulting equations for a length 10 filter. He did not assume symmetry, but his filter was symmetric. We assumed symmetry. We found that with some longer filters, even with symmetry and hence fewer unknowns, sometimes the Gröbner bases were just not a simplication and at other times Maple™ just crashed. We found one solution by using Newton's multivariate method with a random initial vector (our first length 14 quaternion scaling filter), but this method was generally unsuccessful. We had much more success using a genetic algorithm (GA) approach, see *e.g.* Poli *et al.* [120]: a GA uses ideas from genetics (mutation, crossover and selection) to allow a population of random trial solutions to 'evolve' towards an optimum solution over several generations. We give an example of the MATLAB® code we used in Appendix C.

The Maple™ function that Ginzberg wrote to find his scaling filter is in [61, App. C.2, pp. 213–220]. However, some errors crept in when he transferred it to his thesis and this does not work. One of our (working) Maple™ scripts can be found in Appendix B to illustrate our basic method.

5.1.4 Wavelet Filters from Scaling Filters

We use the same method as does Ginzberg for finding matrix-valued wavelet filters from the corresponding matrix-valued scaling filters, namely paraunitary completion of the polyphase matrix. This was first suggested by Xia and Suter [159] and Ginzberg's implementation is described in Ginzberg and Walden [62, Sec. VII]. Ginzberg wrote a MATLAB® function for doing the necessary calculations, available at [61, App. C.1, pp. 211–212], which in turn used functions from the `mw` toolbox of Keinert [82]. We adapted Ginzberg's code to take matrix-valued scaling filters with coefficients of sizes other than 4×4 and our functions are in Appendix D, with an example of a script to call them.

5.1.5 Complex, Quaternion and Other Hypercomplex Solutions

If we take each $\mathbf{S}_k$ in Equations (5.1) to (5.3) as the sum of the two matrices in Equations (2.3), each in turn multiplied by constants to be determined, and $n = 2$, then Ginzberg's equations may be solved to give numerical values for the real and imaginary parts of the coefficients of a complex scaling filter. We did this for lengths 6 and 10 and found exactly the same filter coefficients as those listed by Zhang *et al.* [166], who give complex filters up to length 26: all were solutions of Ginzberg's equations. Sherlock and Kakad [134] provide a MATLAB® script for generating longer complex scaling filters, but we do not use it. All these filters had lengths that were twice odd numbers; the reason for this is explained in Lawton [87] and Lina and Mayrand [96].

Since the basis of complex numbers, $\{1, \mathrm{i}\}$, is isomorphic to subsets $\{1, \mathbf{i}\}$, $\{1, \mathbf{j}\}$ and $\{1, \mathbf{k}\}$ of the quaternion basis, complex scaling filters may be extended to quaternion scaling filters by simply replacing the complex i with the quaternion $\mathbf{i}$, $\mathbf{j}$ or $\mathbf{k}$. In a similar fashion, complex scaling filters may be extended to any Clifford algebra where at least one basis element, bivector or trivector, *etc.*, squares to $-\mathbf{e}_0$. Such filters would be expected to also have lengths that are twice odd numbers. In fact we could not find quaternion or Clifford scaling filters with full quaternions or full multivectors that were not twice odd numbers in length.

Quaternions are isomorphic to the even subalgebra of $Cl(0, 3)$, so quaternion scaling filters could therefore be extended to Clifford scaling filters in this algebra. Subalgebras of other Clifford algebras involving three bivectors and the scalar can be found that are also isomorphic to the quaternions. For example, a glance at the multiplication table for $Cl(0, 8)$ reveals that $\mathbf{e}_{67}^2 = \mathbf{e}_{68}^2 = \mathbf{e}_{78}^2 = -\mathbf{e}_0$, $\mathbf{e}_{67}\mathbf{e}_{68} = \mathbf{e}_{78}$, $\mathbf{e}_{68}\mathbf{e}_{78} = \mathbf{e}_{67}$ and $\mathbf{e}_{78}\mathbf{e}_{67} = \mathbf{e}_{68}$, so that $1 \mapsto \mathbf{e}_0$, $\mathbf{i} \mapsto \mathbf{e}_{67}$, $\mathbf{j} \mapsto \mathbf{e}_{68}$ and $\mathbf{k} \mapsto \mathbf{e}_{78}$, so we could construct a scaling wavelet filter in this algebra if we wished.

Instead of using real matrices isomorphic to complex numbers or quaternions, we can very easily use the $Cl(1,0)$, $Cl(1,1)$ or $Cl(2,0)$ patterns of real matrices given in Equations (2.4), (2.6) and (2.7) respectively. Solutions for these algebras could then be extended to other, larger, Clifford algebras as we have just described for complex and quaternion solutions. A complication arises with the scaling filters in these algebras when writing down Ginzberg's equations: where the basis vector or bivector squares to $-\mathbf{e}_0$, the filter is symmetric in that component and where it squares to $+\mathbf{e}_0$, it is antisymmetric. As we have mentioned, the quaternion scaling filters are symmetric and we can see that this is related to the fact that the basis vectors each square to -1.

Alternatively, we could start off with $2^n \times 2^n$, $n = 3, 4, \dots,$ matrices in Ginzberg's equations, using appropriate patterns for the basis elements of higher algebras, but solving the equations would be difficult.

5.2 Scaling and Wavelet Filter Coefficients

In this section we give some discrete scaling and wavelet filters with quaternion and Clifford valued coefficients, which we found by solving Ginzberg's equations using real matrices in the appropriate patterns as explained in Section 5.1, and then extracting the scalar and imaginary parts from the first column of each matrix.

In each of the explanations we give in Sections 5.2.1 and 5.2.2, the length of the filter in question is $2n$ with n odd and the coefficient at $2n + 1 - k$ is given in terms of the one at $k < n$.

5.2.1 The Complex and $Cl(1,0)$ Cases

For the complex case, Zhang *et al.* [166] list all the scaling filters up to length 26. They give two sets of coefficients for all the lengths, apart from length 6 which only has one. These filters are all symmetric, so if $s[1], s[2], \ldots, s[n]$ are the coefficients they have listed, the full scaling filter is of the form

$$\boldsymbol{s} = s[1], s[2], \ldots, s[k], \ldots, s[n], s[n], \ldots, s[k], \ldots, s[2], s[1].$$

They do not list the corresponding wavelet filters, but the polyphase matrix method gave antisymmetric filters which, in terms of $s[1], s[2], \ldots, s[n]$ can be written

$$\boldsymbol{w} = -\overline{s[1]}, \overline{s[2]}, \ldots, (-1)^k\overline{s[k]}, \ldots, -\overline{s[n]}, \overline{s[n]}, \ldots, (-1)^{k+1}\overline{s[k]}, \ldots, -\overline{s[2]}, \overline{s[1]},$$

the overlines indicating complex conjugates.

In Tables 5.1 to 5.6 we list the coefficients from the first halves of some $Cl(1,0)$ scaling filters. We found that these filters are symmetric in the coefficients of $\mathbf{e}_0$ and antisymmetric in the coefficients of $\mathbf{e}_1$, so that each scaling filter in terms of the coefficients we have listed is of the form

$$\boldsymbol{s} = s[1], s[2], \ldots, s[k], \ldots, s[n], \overline{s[n]}, \ldots, \overline{s[k]}, \ldots, \overline{s[2]}, \overline{s[1]},$$

the overline now indicating the Clifford conjugate. The polyphase matrix method gave wavelet filters that were antisymmetric in the coefficients of $\mathbf{e}_0$ and symmetric in the coefficients of $\mathbf{e}_1$. In terms of $s[1], s[2], \ldots, s[n]$, these can be written

$$\boldsymbol{w} = -\overline{s[1]}, \overline{s[2]}, \ldots, (-1)^k\overline{s[k]}, \ldots, -\overline{s[n]}, s[n], \ldots, (-1)^{k+1}s[k], \ldots, -s[2], s[1].$$

Table 5.1: Our length 6 $Cl(1,0)$ filter coefficients.

$s[1] =$	$0.1839484224\mathbf{e}_0 - 0.1487221305\mathbf{e}_1$
$s[2] =$	$0.3607251177\mathbf{e}_0 - 0.4461663916\mathbf{e}_1$
$s[3] =$	$0.1624332411\mathbf{e}_0 - 0.2974442611\mathbf{e}_1$

Table 5.2: Our first length 10 $Cl(1,0)$ filter coefficients.

$s[1] =$	$0.0234359755\mathbf{e}_0 + 0.0038970928\mathbf{e}_1$
$s[2] =$	$0.0042088285\mathbf{e}_0 + 0.0253106625\mathbf{e}_1$
$s[3] =$	$-0.1072311696\mathbf{e}_0 + 0.0680969203\mathbf{e}_1$
$s[4] =$	$0.1079998199\mathbf{e}_0 + 0.0913977141\mathbf{e}_1$
$s[5] =$	$0.6786933269\mathbf{e}_0 + 0.0447143635\mathbf{e}_1$

Table 5.3: Our second length 10 $Cl(1,0)$ filter coefficients.

$s[1] =$	$0.0817190616\mathbf{e}_0 - 0.0783833363\mathbf{e}_1$
$s[2] =$	$0.2956242589\mathbf{e}_0 - 0.3082050109\mathbf{e}_1$
$s[3] =$	$0.3590335191\mathbf{e}_0 - 0.3652750093\mathbf{e}_1$
$s[4] =$	$0.1079998199\mathbf{e}_0 - 0.0304283260\mathbf{e}_1$
$s[5] =$	$-0.1372698783\mathbf{e}_0 + 0.1050250087\mathbf{e}_1$

Table 5.4: Our first length 14 $Cl(1,0)$ filter coefficients.

$s[1] =$	$0.0071536266\mathbf{e}_0 + 0.0048617927\mathbf{e}_1$
$s[2] =$	$0.0069650392\mathbf{e}_0 + 0.0102483371\mathbf{e}_1$
$s[3] =$	$-0.0415173043\mathbf{e}_0 - 0.0233906992\mathbf{e}_1$
$s[4] =$	$-0.0218335411\mathbf{e}_0 - 0.0422977487\mathbf{e}_1$
$s[5] =$	$0.2024804052\mathbf{e}_0 + 0.1577380557\mathbf{e}_1$
$s[6] =$	$0.3404553362\mathbf{e}_0 + 0.4414662571\mathbf{e}_1$
$s[7] =$	$0.2134032194\mathbf{e}_0 + 0.2702076963\mathbf{e}_1$

Table 5.5: Our second length 14 $Cl(1,0)$ filter coefficients.

$s[1] =$	$0.0064749956\mathbf{e}_0 + 0.0037931811\mathbf{e}_1$
$s[2] =$	$0.0014814300\mathbf{e}_0 + 0.0025288149\mathbf{e}_1$
$s[3] =$	$-0.0602222706\mathbf{e}_0 - 0.0475859671\mathbf{e}_1$
$s[4] =$	$-0.0547658636\mathbf{e}_0 - 0.0852813768\mathbf{e}_1$
$s[5] =$	$0.1782611626\mathbf{e}_0 + 0.1103684691\mathbf{e}_1$
$s[6] =$	$0.3591057410\mathbf{e}_0 + 0.4086585760\mathbf{e}_1$
$s[7] =$	$0.2767715861\mathbf{e}_0 + 0.2593303310\mathbf{e}_1$

Table 5.6: Our third length 14 $Cl(1,0)$ filter coefficients.

$s[1] =$	$0.0094805793\mathbf{e}_0 - 0.0078958241\mathbf{e}_1$
$s[2] =$	$0.0258220330\mathbf{e}_0 - 0.0310047220\mathbf{e}_1$
$s[3] =$	$0.0230000235\mathbf{e}_0 - 0.0357572214\mathbf{e}_1$
$s[4] =$	$0.0921193064\mathbf{e}_0 - 0.0454175577\mathbf{e}_1$
$s[5] =$	$0.2866656119\mathbf{e}_0 - 0.2231870515\mathbf{e}_1$
$s[6] =$	$0.2761793805\mathbf{e}_0 - 0.4553694978\mathbf{e}_1$
$s[7] =$	$-0.0061601534\mathbf{e}_0 - 0.2649516806\mathbf{e}_1$

5.2.2 The Quaternion, $Cl(1,1)$ and $Cl(2,0)$ Cases

In Tables 5.7 to 5.12 we list the coefficients from the first halves of some quaternion scaling filters and each one's associated wavelet filter. It will be noticed that some of these have zero **j** parts: this is a consequence of choosing two of them to be zero as per Subsection 5.1.2 and the subsequent algebraic manipulations. We found that the scaling filters were symmetric and the wavelet filters were antisymmetric, so

$$\boldsymbol{s} = s[1], s[2], \ldots, s[k], \ldots, s[n], s[n], \ldots, s[k], \ldots, s[2], s[1]$$

and

$$\boldsymbol{w} = w[1], w[2], \ldots, w[k], \ldots, w[n], -w[n], \ldots, -w[k], \ldots, -w[2], -w[1].$$

We can not now express $\boldsymbol{w}$ in terms of $s[1], s[2], \ldots, s[n]$.

In Table 5.13 we list the first halves of a $Cl(1,1)$ scaling filter and its associated wavelet filter. The scaling filter was symmetric in $\mathbf{e}_0$ and $\mathbf{e}_2$ and antisymmetric in $\mathbf{e}_1$ and $\mathbf{e}_{12}$: if we represent $s[k] = a_k\mathbf{e}_0 + b_k\mathbf{e}_1 + c_k\mathbf{e}_2 + d_k\mathbf{e}_{12}$ as the ordered quadruple (a_k, b_k, c_k, d_k) for clarity, then

$$\boldsymbol{s} = (a_1, b_1, c_1, d_1), (a_2, b_2, c_2, d_2), \ldots, (a_k, b_k, c_k, d_k), \ldots, (a_n, b_n, c_n, d_n),$$

$$(a_n, -b_n, c_n, -d_n), \ldots, (a_k, -b_k, c_k, -d_k), \ldots, (a_2, -b_2, c_2, -d_2), (a_1, -b_1, c_1, -d_1).$$

The wavelet filter was antisymmetric in $\mathbf{e}_0$ and $\mathbf{e}_2$ and symmetric in $\mathbf{e}_1$ and $\mathbf{e}_{12}$, the opposite of the scaling filter.

It is at this stage that we can now make some observations on the symmetry and antisymmetry of our scaling filters:

- The complex i squares to -1 and the imaginary part of Zhang *et al.*'s scaling filters were all symmetric.
- The quaternion $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$ each square to -1 and the vector parts of Ginzberg's and our scaling filters were symmetric.
- The $\mathbf{e}_1$ of $Cl(1,0)$ squares to $+\mathbf{e}_0$ and our scaling filters were antisymmetric in $\mathbf{e}_1$.
- The $\mathbf{e}_1$ and $\mathbf{e}_{12}$ of $Cl(1,1)$ each square to $+\mathbf{e}_0$, the $\mathbf{e}_2$ squares to $-\mathbf{e}_0$ and our scaling filters were antisymmetric in $\mathbf{e}_1$ and $\mathbf{e}_{12}$ and symmetric in $\mathbf{e}_2$.
- All scaling filters are symmetric in $\mathbf{e}_0$ or 1 (the latter if we specialise to complex or quaternion algebras).

We can see that the symmetry and antisymmetry of Clifford scaling filters is intricately dependant on the signature of the particular algebra: we hypothesise that if a component (vector, bivector, trivector, *etc.*) of an algebra squares to $-\mathbf{e}_0$ ($+\mathbf{e}_0$), any scaling filter in this algebra is symmetric (antisymmetric) in this component.

In Tables 5.14 and 5.15, we list the coefficients from the first halves of two $Cl(2,0)$ scaling filters and each one's associated wavelet filter. We found that the simplification in Subsection 5.1.2 for quaternions works with this algebra too, hence the zero $\mathbf{e}_2$ parts. From the above, since the $\mathbf{e}_1$ and $\mathbf{e}_2$ of $Cl(2,0)$ each square to $+\mathbf{e}_0$ and the $\mathbf{e}_{12}$ squares to $-\mathbf{e}_0$, we expect these filters to be antisymmetric in $\mathbf{e}_1$ and $\mathbf{e}_2$ and symmetric in $\mathbf{e}_0$ and $\mathbf{e}_{12}$: this is what we find, since the coefficients of $\mathbf{e}_2$ are zero anyway. In terms of the quadruples used above,

$$\begin{aligned}\boldsymbol{s} = {} & (a_1, b_1, c_1, d_1), (a_2, b_2, c_2, d_2), \ldots, (a_k, b_k, c_k, d_k), \ldots, (a_n, b_n, c_n, d_n), \\ & (a_n, -b_n, -c_n, d_n), \ldots, (a_k, -b_k, -c_k, d_k), \ldots, (a_2, -b_2, -c_2, d_2), (a_1, -b_1, -c_1, d_1).\end{aligned}$$

The wavelet filter was antisymmetric in $\mathbf{e}_0$ and $\mathbf{e}_{12}$ and symmetric in $\mathbf{e}_1$ and $\mathbf{e}_2$, again, the opposite of the scaling filter.

Table 5.7: Ginzberg's length 10 quaternion filter coefficients.

$s[1] =$		$0.0231096867\mathbf{i}$		
$s[2] =$	-0.0276213586			$+\,0.0231096867\mathbf{k}$
$s[3] =$	-0.0386699021	$-\,0.1617678066\mathbf{i}$		$+\,0.0693290600\mathbf{k}$
$s[4] =$	0.1933495105	$-\,0.1155484333\mathbf{i}$		$+\,0.0231096867\mathbf{k}$
$s[5] =$	0.5800485314	$+\,0.2542065532\mathbf{i}$		$-\,0.1155484333\mathbf{k}$
$w[1] =$		$0.0214246053\mathbf{i}$	$+\,0.0020140574\mathbf{j}$	$-\,0.0084254066\mathbf{k}$
$w[2] =$	-0.0276213586	$+\,0.0084254066\mathbf{i}$	$-\,0.0100702870\mathbf{j}$	$+\,0.0190173463\mathbf{k}$
$w[3] =$	0.0386699021	$-\,0.1752484571\mathbf{i}$	$+\,0.0161124592\mathbf{j}$	$+\,0.0019258072\mathbf{k}$
$w[4] =$	0.1933495105	$+\,0.1155484333\mathbf{i}$		$-\,0.0231096867\mathbf{k}$
$w[5] =$	-0.5800485314	$+\,0.2777976916\mathbf{i}$	$-\,0.0281968036\mathbf{j}$	$+\,0.0024072590\mathbf{k}$

Table 5.8: Our length 10 quaternion filter coefficients.

$s[1] =$	-0.0115088994	$-\,0.0200400312\mathbf{i}$		
$s[2] =$	-0.0391302581	$+\,0.0224723305\mathbf{i}$		$+\,0.0166753392\mathbf{k}$
$s[3] =$	0.0073656956	$+\,0.2076972100\mathbf{i}$		$+\,0.0500260177\mathbf{k}$
$s[4] =$	0.2393851082	$+\,0.1226724866\mathbf{i}$		$+\,0.0166753392\mathbf{k}$
$s[5] =$	0.5109951348	$-\,0.3328019959\mathbf{i}$		$-\,0.0833766962\mathbf{k}$
$w[1] =$	0.0115088994	$-\,0.0193286077\mathbf{i}$	$+\,0.0007856106\mathbf{j}$	$-\,0.0052336021\mathbf{k}$
$w[2] =$	-0.0391302581	$-\,0.0260294481\mathbf{i}$	$-\,0.0039280528\mathbf{j}$	$+\,0.0094926715\mathbf{k}$
$w[3] =$	-0.0073656956	$+\,0.2133885981\mathbf{i}$	$+\,0.0062848845\mathbf{j}$	$+\,0.0081572006\mathbf{k}$
$w[4] =$	0.2393851082	$-\,0.1226724866\mathbf{i}$		$-\,0.0166753392\mathbf{k}$
$w[5] =$	-0.5109951348	$-\,0.3427619251\mathbf{i}$	$-\,0.0109985479\mathbf{j}$	$-\,0.0101062662\mathbf{k}$

Table 5.9: Our first length 14 quaternion filter coefficients.

$s[1] =$	0.0008726799	$+ 0.0051745314\mathbf{i}$		
$s[2] =$	-0.0034248375	$+ 0.0005775956\mathbf{i}$		$+ 0.0106706344\mathbf{k}$
$s[3] =$	-0.0277961071	$- 0.0483604296\mathbf{i}$	$- 0.0034163721\mathbf{j}$	$+ 0.0181899480\mathbf{k}$
$s[4] =$	-0.0382273962	$- 0.0440624205\mathbf{i}$	$- 0.0170818603\mathbf{j}$	$- 0.0797804100\mathbf{k}$
$s[5] =$	0.0603628899	$+ 0.1000047973\mathbf{i}$	$- 0.0273309765\mathbf{j}$	$- 0.2279526191\mathbf{k}$
$s[6] =$	0.2708947155	$+ 0.1051995865\mathbf{i}$		$- 0.0640238062\mathbf{k}$
$s[7] =$	0.4444248367	$- 0.1185336607\mathbf{i}$	$+ 0.0478292089\mathbf{j}$	$+ 0.3428962529\mathbf{k}$
$w[1] =$	-0.0010074032	$+ 0.0027633983\mathbf{i}$	$+ 0.0011659825\mathbf{j}$	$- 0.0041864781\mathbf{k}$
$w[2] =$	-0.0027512208	$+ 0.0081218935\mathbf{i}$	$- 0.0057253119\mathbf{j}$	$+ 0.0044285439\mathbf{k}$
$w[3] =$	0.0269877671	$- 0.0393339930\mathbf{i}$	$+ 0.0028473183\mathbf{j}$	$+ 0.0339035658\mathbf{k}$
$w[4] =$	-0.0395746296	$- 0.0438162635\mathbf{i}$	$+ 0.0307290983\mathbf{j}$	$- 0.0729154060\mathbf{k}$
$w[5] =$	-0.0564559129	$+ 0.2404073634\mathbf{i}$	$- 0.0633410837\mathbf{j}$	$+ 0.0159527059\mathbf{k}$
$w[6] =$	0.2696822054	$- 0.1067627257\mathbf{i}$	$+ 0.0098662386\mathbf{j}$	$+ 0.0613447773\mathbf{k}$
$w[7] =$	-0.4492748771	$- 0.3462938645\mathbf{i}$	$+ 0.0941978079\mathbf{j}$	$- 0.0528118784\mathbf{k}$

Table 5.10: Our second length 14 quaternion filter coefficients.

$s[1] =$		$0.0052476038\mathbf{i}$		
$s[2] =$				$- 0.0113708959\mathbf{k}$
$s[3] =$	-0.0096674755	$- 0.0526307873\mathbf{i}$		$- 0.0245744726\mathbf{k}$
$s[4] =$	-0.0372888342	$- 0.0532497837\mathbf{i}$		$+ 0.0590619712\mathbf{k}$
$s[5] =$		$0.0932188760\mathbf{i}$		$+ 0.2013855752\mathbf{k}$
$s[6] =$	0.2320194126	$+ 0.1101996803\mathbf{i}$		$+ 0.0682253753\mathbf{k}$
$s[7] =$	0.5220436783	$- 0.1027855891\mathbf{i}$		$- 0.2927275533\mathbf{k}$
$w[1] =$	0.0000019848	$+ 0.0027914221\mathbf{i}$	$- 0.0000045798\mathbf{j}$	$+ 0.0044435665\mathbf{k}$
$w[2] =$	-0.0000099238	$+ 0.0096286485\mathbf{i}$	$- 0.0000043007\mathbf{j}$	$- 0.0060486598\mathbf{k}$
$w[3] =$	0.0096793841	$- 0.0488020568\mathbf{i}$	$+ 0.0001629185\mathbf{j}$	$- 0.0315010126\mathbf{k}$
$w[4] =$	-0.0372689866	$- 0.0217049757\mathbf{i}$	$- 0.0005625900\mathbf{j}$	$+ 0.0765409580\mathbf{k}$
$w[5] =$	-0.0000575580	$+ 0.2201456083\mathbf{i}$	$+ 0.0007040049\mathbf{j}$	$- 0.0282416724\mathbf{k}$
$w[6] =$	0.2320372754	$- 0.1163917550\mathbf{i}$	$+ 0.0001219796\mathbf{j}$	$- 0.0570229387\mathbf{k}$
$w[7] =$	-0.5219722270	$- 0.3026030559\mathbf{i}$	$- 0.0013072548\mathbf{j}$	$+ 0.0687684780\mathbf{k}$

Table 5.11: Our third length 14 quaternion filter coefficients.

$s[1] =$		$0.0052476038\mathbf{i}$		
$s[2] =$				$-\ 0.0059188911\mathbf{k}$
$s[3] =$	-0.0096674755	$-\ 0.0436491641\mathbf{i}$		$-\ 0.0073956026\mathbf{k}$
$s[4] =$	-0.0372888342	$-\ 0.0083416676\mathbf{i}$		$+\ 0.0577242451\mathbf{k}$
$s[5] =$		$0.1650718619\mathbf{i}$		$+\ 0.1479963688\mathbf{k}$
$s[6] =$	0.2320194126	$+\ 0.1101996803\mathbf{i}$		$+\ 0.0355133468\mathbf{k}$
$s[7] =$	0.5220436783	$-\ 0.2285283143\mathbf{i}$		$-\ 0.2279194669\mathbf{k}$
$w[1] =$	-0.0000029730	$+\ 0.0043083101\mathbf{i}$	$+\ 0.0000131791\mathbf{j}$	$+\ 0.0029959351\mathbf{k}$
$w[2] =$	0.0000148650	$+\ 0.0033791830\mathbf{i}$	$+\ 0.0000033533\mathbf{j}$	$-\ 0.0048594405\mathbf{k}$
$w[3] =$	0.0096496375	$-\ 0.0400639282\mathbf{i}$	$-\ 0.0004056673\mathbf{j}$	$-\ 0.0188293991\mathbf{k}$
$w[4] =$	-0.0373185642	$-\ 0.0260796925\mathbf{i}$	$+\ 0.0014475190\mathbf{j}$	$+\ 0.0520606396\mathbf{k}$
$w[5] =$	0.0000862171	$+\ 0.2199743729\mathbf{i}$	$-\ 0.0018364197\mathbf{j}$	$-\ 0.0271139706\mathbf{k}$
$w[6] =$	0.2319926556	$-\ 0.1107496109\mathbf{i}$	$-\ 0.0002968811\mathbf{j}$	$-\ 0.0337579941\mathbf{k}$
$w[7] =$	-0.5221507064	$-\ 0.3176688753\mathbf{i}$	$+\ 0.0033828991\mathbf{j}$	$+\ 0.0563906396\mathbf{k}$

Table 5.12: Our length 18 quaternion filter coefficients.

$s[1] =$	0.0006230677	$-\ 0.0010535900\mathbf{i}$		
$s[2] =$	-0.0000356563	$-\ 0.0000210863\mathbf{i}$		$-\ 0.0028442993\mathbf{k}$
$s[3] =$	-0.0064750150	$+\ 0.0137100703\mathbf{i}$	$-\ 0.0025876009\mathbf{j}$	$-\ 0.0060006527\mathbf{k}$
$s[4] =$	-0.0036872768	$+\ 0.0139636957\mathbf{i}$	$-\ 0.0112913827\mathbf{j}$	$+\ 0.0222795608\mathbf{k}$
$s[5] =$	0.0125687167	$-\ 0.0499570230\mathbf{i}$	$-\ 0.0039992540\mathbf{j}$	$+\ 0.0711446181\mathbf{k}$
$s[6] =$	-0.0196163538	$-\ 0.0978839453\mathbf{i}$	$+\ 0.0571618185\mathbf{j}$	$+\ 0.0038337494\mathbf{k}$
$s[7] =$	-0.0404858759	$-\ 0.0120836135\mathbf{i}$	$+\ 0.1096194778\mathbf{j}$	$-\ 0.1615686050\mathbf{k}$
$s[8] =$	0.1949326422	$+\ 0.0788793402\mathbf{i}$	$+\ 0.0134086782\mathbf{j}$	$-\ 0.1004103109\mathbf{k}$
$s[9] =$	0.5692825324	$+\ 0.0544461519\mathbf{i}$	$-\ 0.1623117370\mathbf{j}$	$+\ 0.1735659396\mathbf{k}$
$w[1] =$	-0.0007442939	$-\ 0.0003676904\mathbf{i}$	$+\ 0.0002656821\mathbf{j}$	$+\ 0.0008593653\mathbf{k}$
$w[2] =$	0.0004726395	$-\ 0.0020079097\mathbf{i}$	$-\ 0.0019526825\mathbf{j}$	$+\ 0.0001539350\mathbf{k}$
$w[3] =$	0.0066749539	$+\ 0.0071682822\mathbf{i}$	$+\ 0.0021090994\mathbf{j}$	$-\ 0.0133071235\mathbf{k}$
$w[4] =$	-0.0077749905	$+\ 0.0188996896\mathbf{i}$	$+\ 0.0126157855\mathbf{j}$	$+\ 0.0177058801\mathbf{k}$
$w[5] =$	-0.0070348284	$-\ 0.0695802235\mathbf{i}$	$-\ 0.0352783032\mathbf{j}$	$+\ 0.0425690412\mathbf{k}$
$w[6] =$	-0.0134698405	$-\ 0.0075083629\mathbf{i}$	$+\ 0.0140024315\mathbf{j}$	$-\ 0.1197473888\mathbf{k}$
$w[7] =$	0.0208299860	$+\ 0.1928419305\mathbf{i}$	$+\ 0.0476425268\mathbf{j}$	$+\ 0.0756371013\mathbf{k}$
$w[8] =$	0.2013427742	$-\ 0.1073662726\mathbf{i}$	$-\ 0.0425174369\mathbf{j}$	$+\ 0.0614701231\mathbf{k}$
$w[9] =$	-0.5462620162	$-\ 0.2280451544\mathbf{i}$	$-\ 0.0325909074\mathbf{j}$	$-\ 0.1461758348\mathbf{k}$

Table 5.13: Our length 10 $Cl(1,1)$ filter coefficients.

$s[1] =$	$-0.0334224383\mathbf{e}_0$	$-\ 0.0010756804\mathbf{e}_1$	$-\ 0.0390293010\mathbf{e}_2$	$-\ 0.0458817067\mathbf{e}_{12}$
$s[2] =$	$-0.0594285069\mathbf{e}_0$	$-\ 0.0058111869\mathbf{e}_1$	$-\ 0.1174129287\mathbf{e}_2$	$-\ 0.1430317142\mathbf{e}_{12}$
$s[3] =$	$0.0998657209\mathbf{e}_0$	$-\ 0.0129207280\mathbf{e}_1$	$-\ 0.0790336789\mathbf{e}_2$	$-\ 0.0269329698\mathbf{e}_{12}$
$s[4] =$	$0.3286545535\mathbf{e}_0$	$-\ 0.0146518669\mathbf{e}_1$	$+\ 0.0777335765\mathbf{e}_2$	$+\ 0.3185743083\mathbf{e}_{12}$
$s[5] =$	$0.3714374520\mathbf{e}_0$	$-\ 0.0064666454\mathbf{e}_1$	$+\ 0.1577423321\mathbf{e}_2$	$+\ 0.2483572707\mathbf{e}_{12}$
$w[1] =$	$0.0334281090\mathbf{e}_0$	$+\ 0.0031716775\mathbf{e}_1$	$-\ 0.0385025671\mathbf{e}_2$	$-\ 0.0453405887\mathbf{e}_{12}$
$w[2] =$	$-0.0594568605\mathbf{e}_0$	$-\ 0.0135660893\mathbf{e}_1$	$+\ 0.1147792589\mathbf{e}_2$	$+\ 0.1403554617\mathbf{e}_{12}$
$w[3] =$	$-0.0998203552\mathbf{e}_0$	$+\ 0.0202552839\mathbf{e}_1$	$-\ 0.0748198071\mathbf{e}_2$	$-\ 0.0216684779\mathbf{e}_{12}$
$w[4] =$	$0.3286545535\mathbf{e}_0$	$-\ 0.0110860911\mathbf{e}_1$	$-\ 0.0777335765\mathbf{e}_2$	$-\ 0.3237214495\mathbf{e}_{12}$
$w[5] =$	$-0.3715168419\mathbf{e}_0$	$+\ 0.0012252189\mathbf{e}_1$	$+\ 0.1503680566\mathbf{e}_2$	$+\ 0.2503750544\mathbf{e}_{12}$

Table 5.14: Our first length 10 $Cl(2,0)$ filter coefficients.

$s[1] =$	$-0.0231096867\mathbf{e}_0$			
$s[2] =$		$-\ 0.0334814623\mathbf{e}_1$		$+\ 0.0667879660\mathbf{e}_{12}$
$s[3] =$	$0.2059619804\mathbf{e}_0$	$-\ 0.1674073114\mathbf{e}_1$		$+\ 0.2003638981\mathbf{e}_{12}$
$s[4] =$	$0.3365193024\mathbf{e}_0$	$-\ 0.3013331606\mathbf{e}_1$		$+\ 0.0667879660\mathbf{e}_{12}$
$s[5] =$	$0.1877351851\mathbf{e}_0$	$-\ 0.1674073114\mathbf{e}_1$		$-\ 0.3339398301\mathbf{e}_{12}$
$w[1] =$	$0.0252705635\mathbf{e}_0$	$-\ 0.0103054229\mathbf{e}_1$	$+\ 0.0006167757\mathbf{e}_2$	$+\ 0.0014277227\mathbf{e}_{12}$
$w[2] =$	$-0.0108043842\mathbf{e}_0$	$+\ 0.0383946623\mathbf{e}_1$	$+\ 0.0277146118\mathbf{e}_2$	$-\ 0.0739265796\mathbf{e}_{12}$
$w[3] =$	$-0.1886749656\mathbf{e}_0$	$-\ 0.0373919674\mathbf{e}_1$	$-\ 0.1478246947\mathbf{e}_2$	$+\ 0.2117856798\mathbf{e}_{12}$
$w[4] =$	$0.3365193024\mathbf{e}_0$	$-\ 0.0151378423\mathbf{e}_1$	$+\ 0.2710186563\mathbf{e}_2$	$-\ 0.0667879660\mathbf{e}_{12}$
$w[5] =$	$-0.2179874609\mathbf{e}_0$	$+\ 0.0244405703\mathbf{e}_1$	$-\ 0.1515253491\mathbf{e}_2$	$-\ 0.3539279481\mathbf{e}_{12}$

Table 5.15: Our second length 10 $Cl(2,0)$ filter coefficients.

$s[1] =$	$-0.0370958744\mathbf{e}_0$	$+\ 0.0290180337\mathbf{e}_1$		
$s[2] =$	$-0.0613368978\mathbf{e}_0$	$+\ 0.0784114417\mathbf{e}_1$		$-\ 0.0508981554\mathbf{e}_{12}$
$s[3] =$	$0.1198546011\mathbf{e}_0$	$-\ 0.0432132968\mathbf{e}_1$		$-\ 0.1526944663\mathbf{e}_{12}$
$s[4] =$	$0.3451133432\mathbf{e}_0$	$-\ 0.3099282038\mathbf{e}_1$		$-\ 0.0508981554\mathbf{e}_{12}$
$s[5] =$	$0.3405716091\mathbf{e}_0$	$-\ 0.2173214989\mathbf{e}_1$		$+\ 0.2544907772\mathbf{e}_{12}$
$w[1] =$	$0.0352845243\mathbf{e}_0$	$-\ 0.0096312903\mathbf{e}_1$	$-\ 0.0248865744\mathbf{e}_2$	$+\ 0.0010785653\mathbf{e}_{12}$
$w[2] =$	$-0.0522801474\mathbf{e}_0$	$+\ 0.0163843250\mathbf{e}_1$	$+\ 0.0658104825\mathbf{e}_2$	$+\ 0.0455053291\mathbf{e}_{12}$
$w[3] =$	$-0.1343454017\mathbf{e}_0$	$+\ 0.0625477296\mathbf{e}_1$	$+\ 0.0442462040\mathbf{e}_2$	$-\ 0.1440659442\mathbf{e}_{12}$
$w[4] =$	$0.3451133432\mathbf{e}_0$	$-\ 0.1896362357\mathbf{e}_1$	$-\ 0.2787357629\mathbf{e}_2$	$+\ 0.0508981554\mathbf{e}_{12}$
$w[5] =$	$-0.3152127079\mathbf{e}_0$	$+\ 0.1203354714\mathbf{e}_1$	$+\ 0.1935656507\mathbf{e}_2$	$+\ 0.2393908635\mathbf{e}_{12}$

5.3 Scaling and Wavelet Functions

We introduced the cascade algorithm in Section 3.6, Equations (3.9) and (3.10), and demonstrated its use with real Daubechies filters. It was devised with real scaling and wavelet filters in mind but as we shall see, it works perfectly well with hypercomplex filters, although we cannot use the `wavefun` function. Instead, we coded the cascade algorithm using functions from the Clifford multivector toolbox [130], described in [131], and the quaternion toolbox [128], and used it to find the complex and $Cl(1,0)$ plots of Subsection 5.3.1 and the quaternion, $Cl(1,1)$ and $Cl(2,0)$ plots of Subsection 5.3.2. We give an example of our code in Appendix E.

Complex and $Cl(1,0)$ numbers are two dimensional (treating the Clifford $\mathbf{e}_0$ and $\mathbf{e}_1$ equally) but if we add the t from the cascade algorithm, we get three dimensions. Similarly, quaternion, $Cl(1,1)$ and $Cl(2,0)$ numbers are four dimensional (treating the Clifford $\mathbf{e}_0$, $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_{12}$ equally) but if we add the t from the cascade algorithm, we get five dimensions. We can display these objects by projecting them onto two and three dimensions.

5.3.1 The Complex and $Cl(1,0)$ Cases

Since we have three dimensional objects, we have three two dimensional projections and one three dimensional plot for each scaling function and each wavelet function. In Figures 5.1, 5.2 and 5.3 (the last for comparison with the quaternion plots), we show the scaling function in blue on the left and the wavelet function in red on the right. The t gives the position along the vectors of samples from the cascade algorithm, s is the scalar and x is the real-valued coefficient of i or $\mathbf{e}_1$. We discuss these plots in Section 5.5. Similar plots for the remaining $Cl(1,0)$ filters and for lengths 10 and 14 complex filters may be found in our article [43].

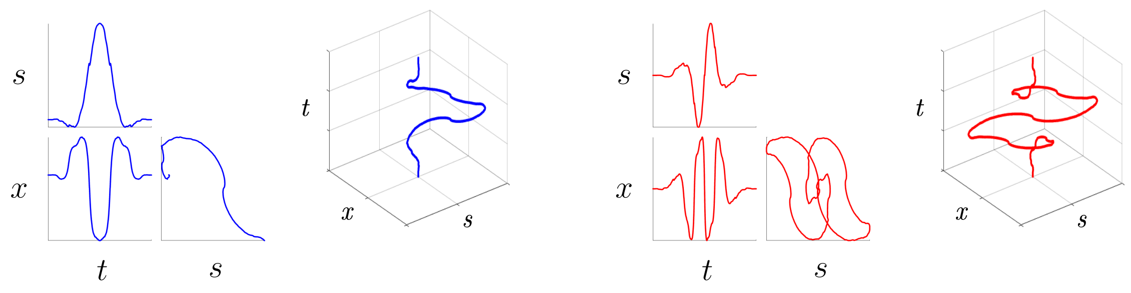


Figure 5.1: Complex length 6.

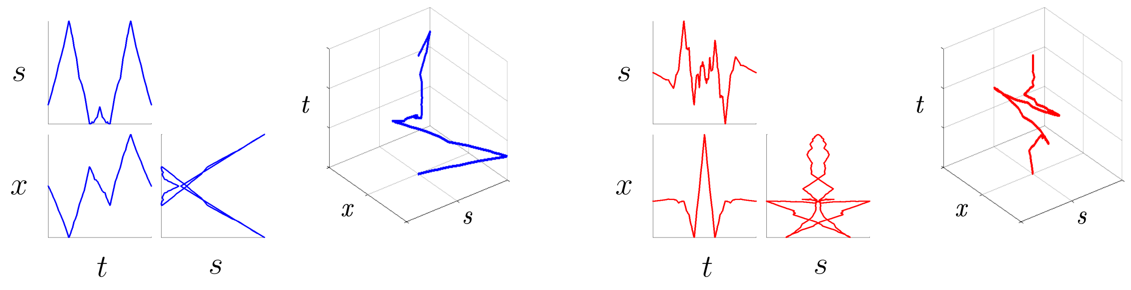


Figure 5.2: $Cl(1,0)$ length 6.

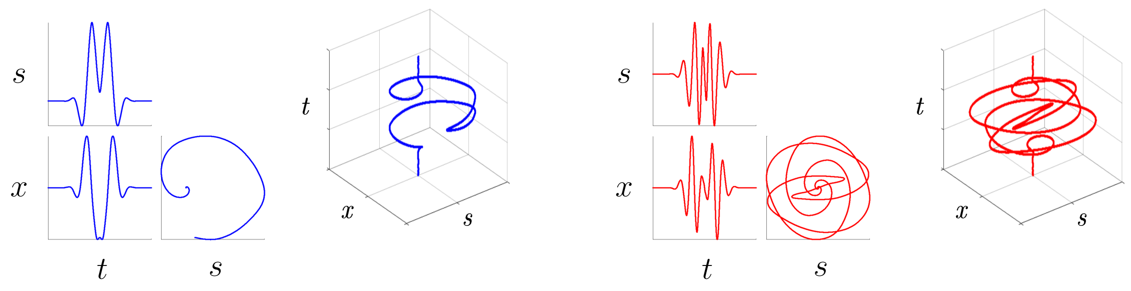
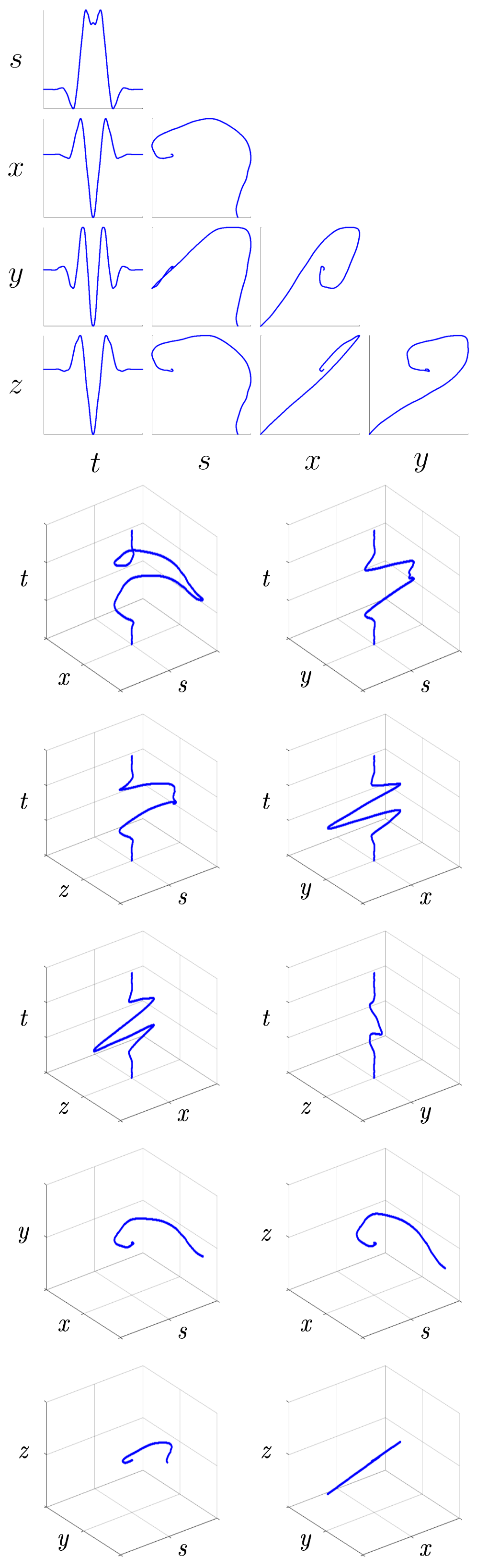


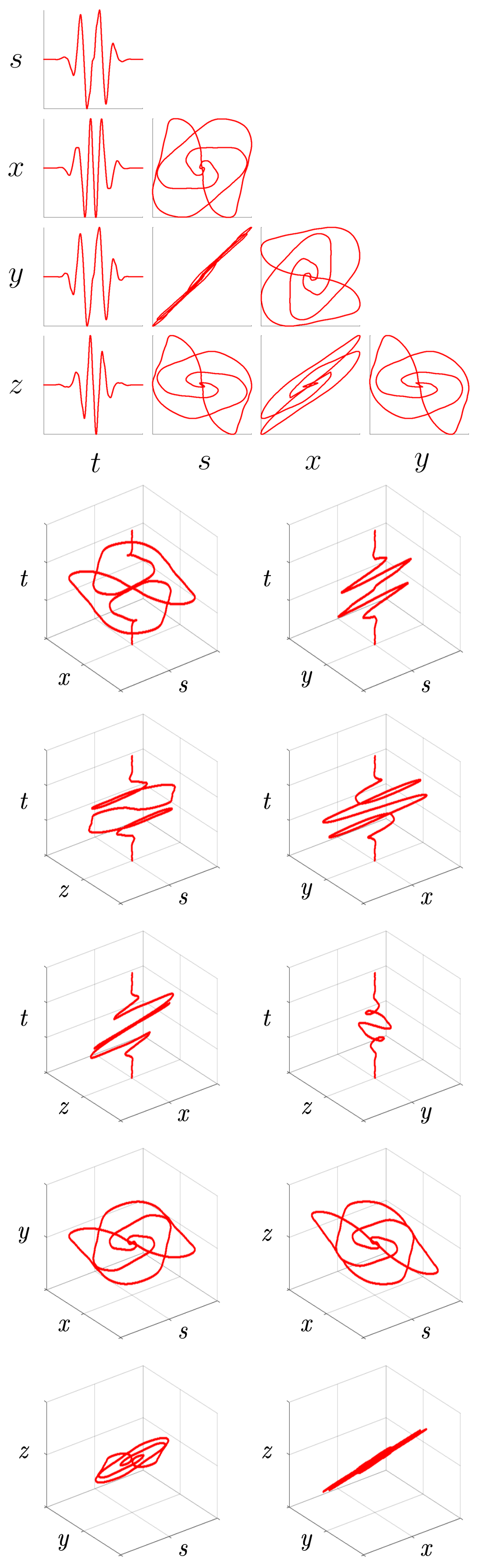
Figure 5.3: Complex length 14 case 1.

5.3.2 The Quaternion, $Cl(1,1)$ and $Cl(2,0)$ Cases

On the following five pages, we give the scaling and wavelet functions for out length 10, our first length 14 and our length 18 quaternion filters and our $Cl(1,1)$ and first $Cl(2,0)$ Clifford filters.

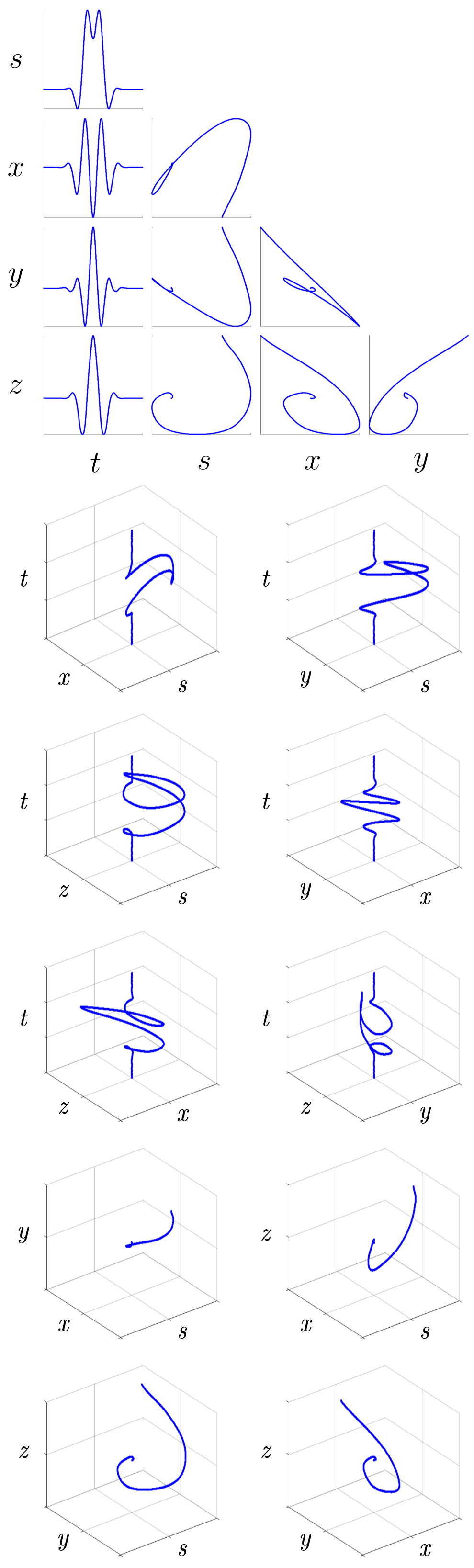


(a) $\Phi(t)$

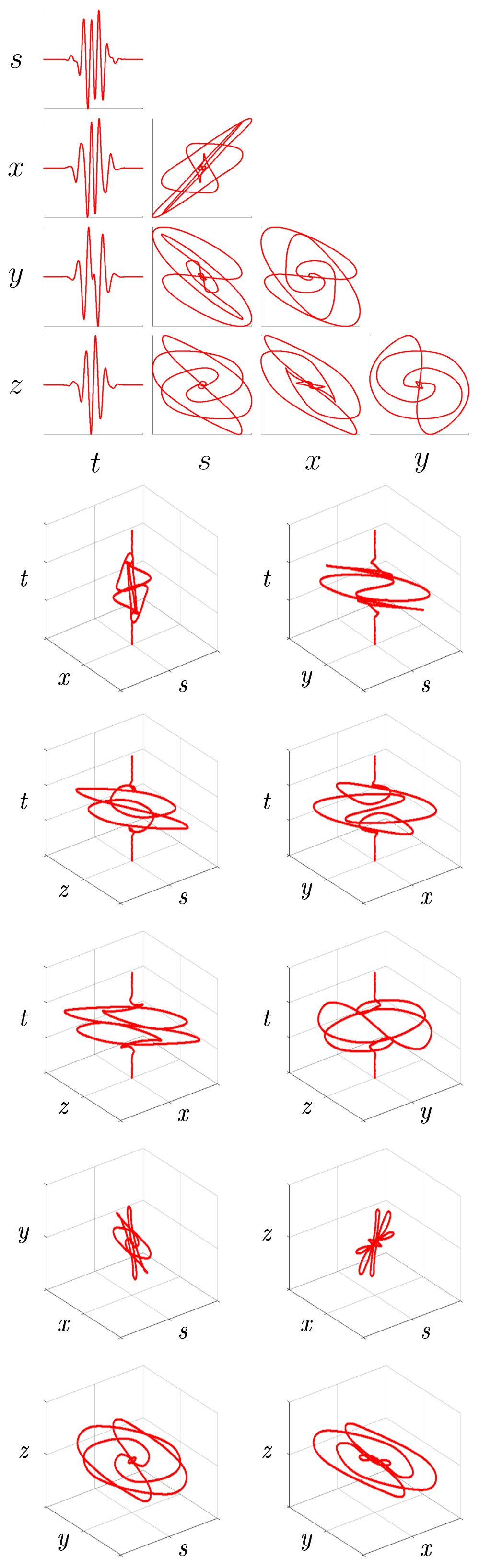


(b) $\Psi(t)$

Figure 5.4: Projections of quaternion scaling and wavelet functions onto 2-D and 3-D using our length 10 filters.

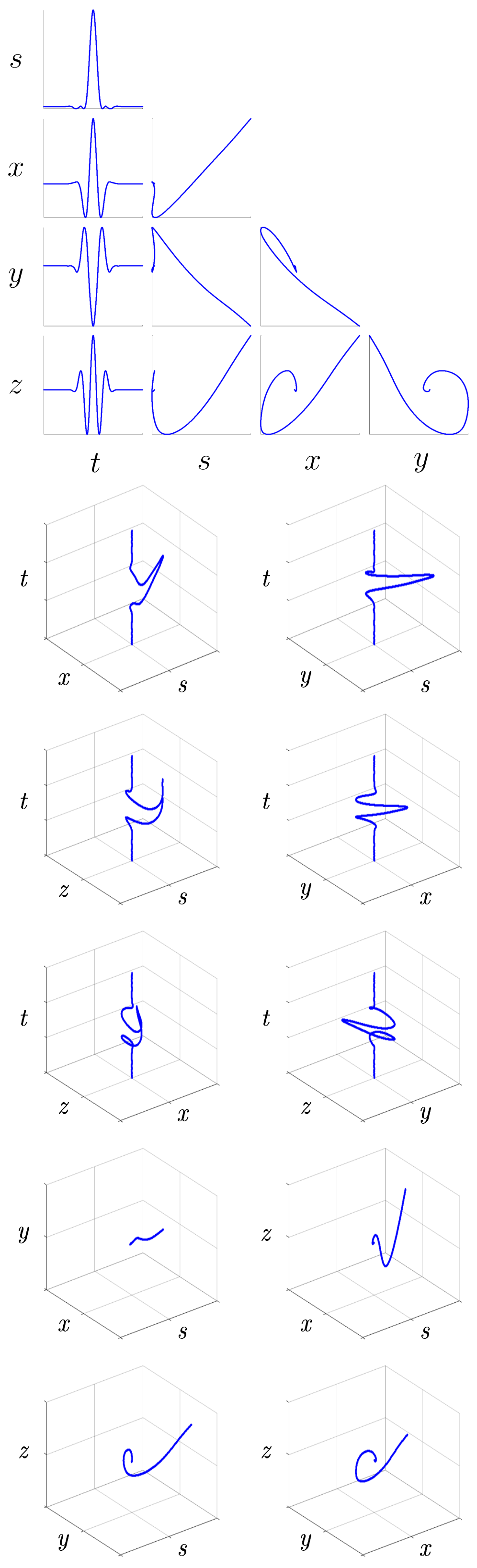


(a) $\Phi(t)$

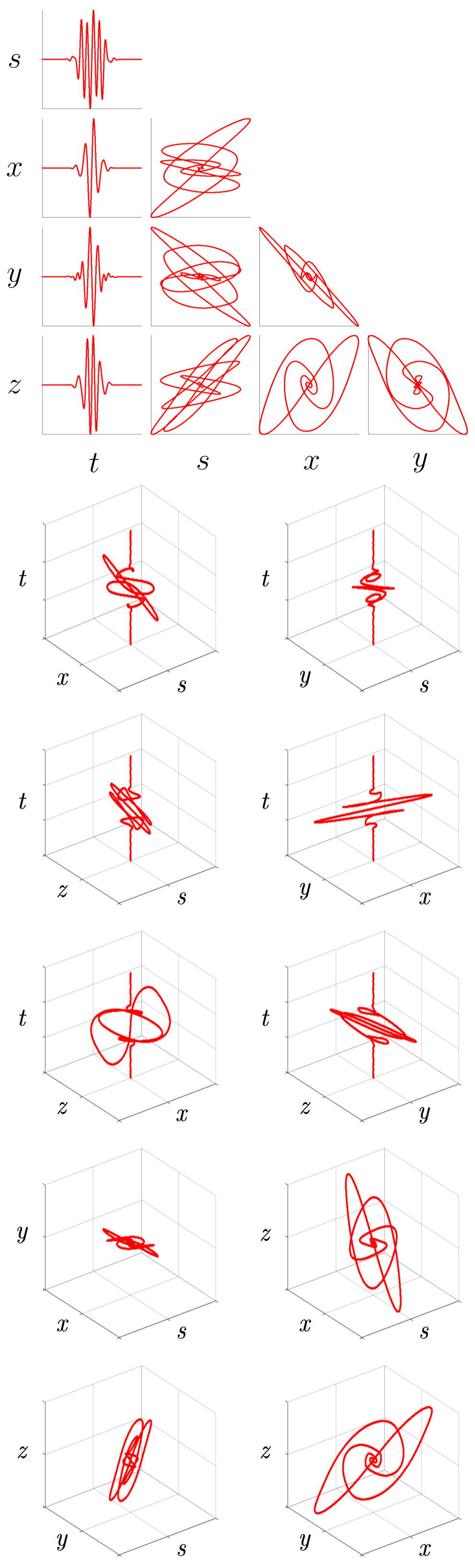


(b) $\Psi(t)$

Figure 5.5: Projections of quaternion scaling and wavelet functions onto 2-D and 3-D using our first length 14 filters.

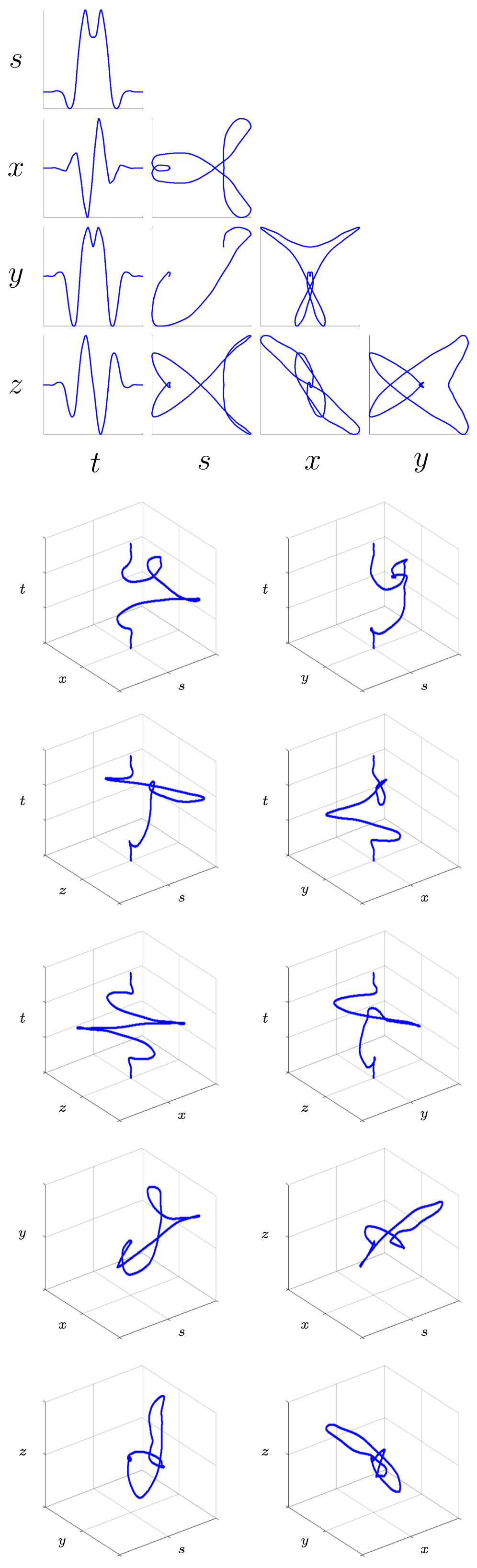


(a) $\Phi(t)$

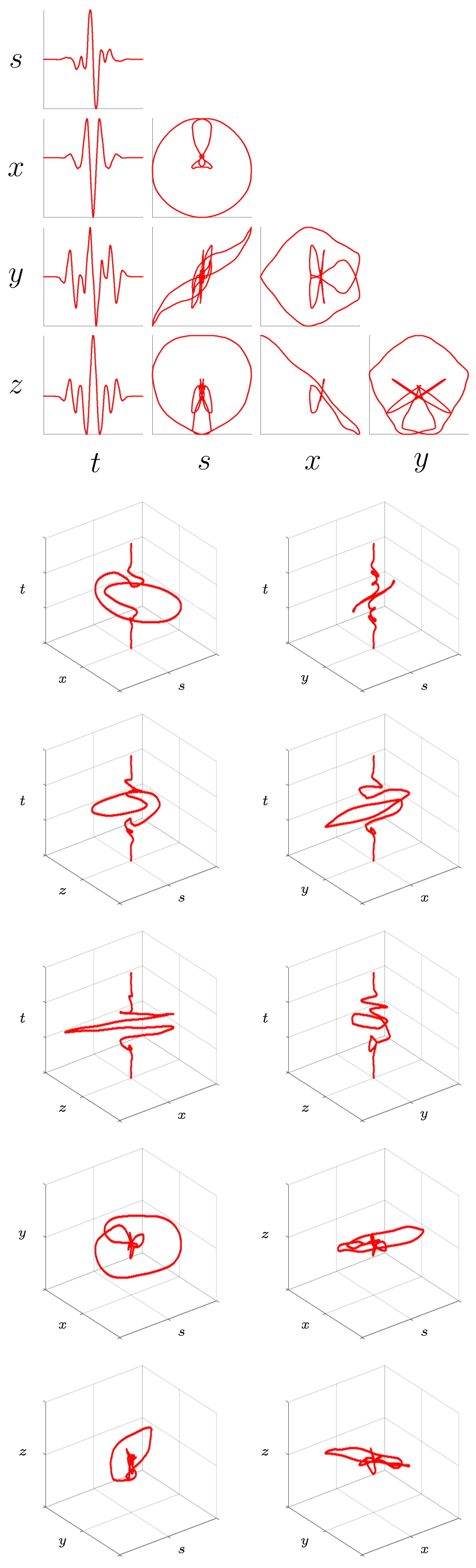


(b) $\Psi(t)$

Figure 5.6: Projections of quaternion scaling and wavelet functions onto 2-D and 3-D using our length 18 filters.

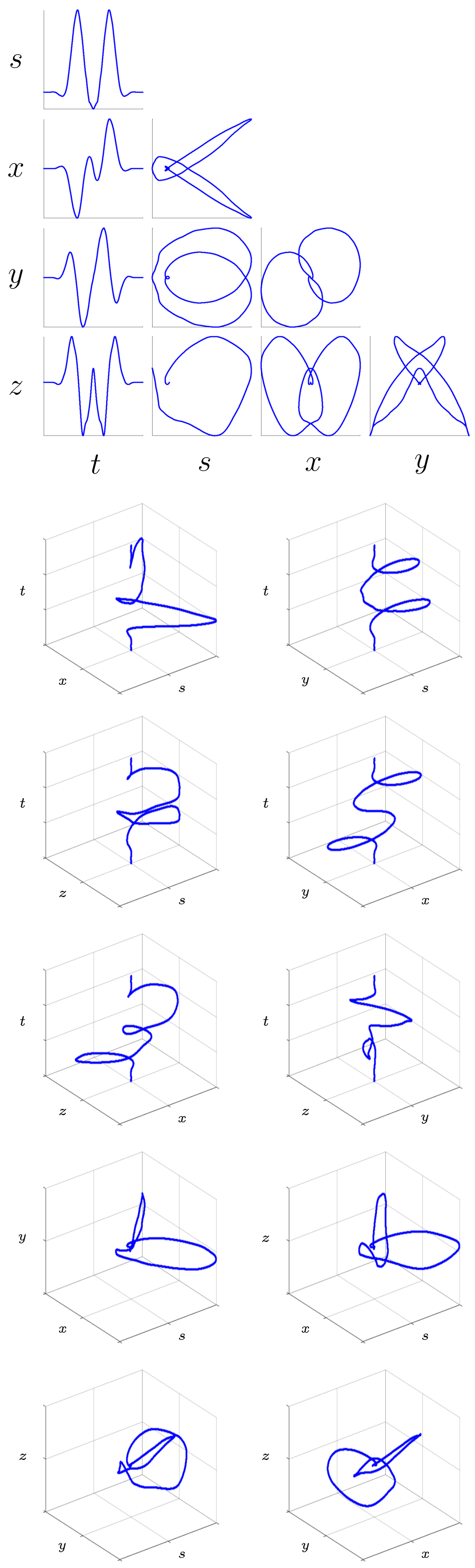


(a) $\Phi(t)$

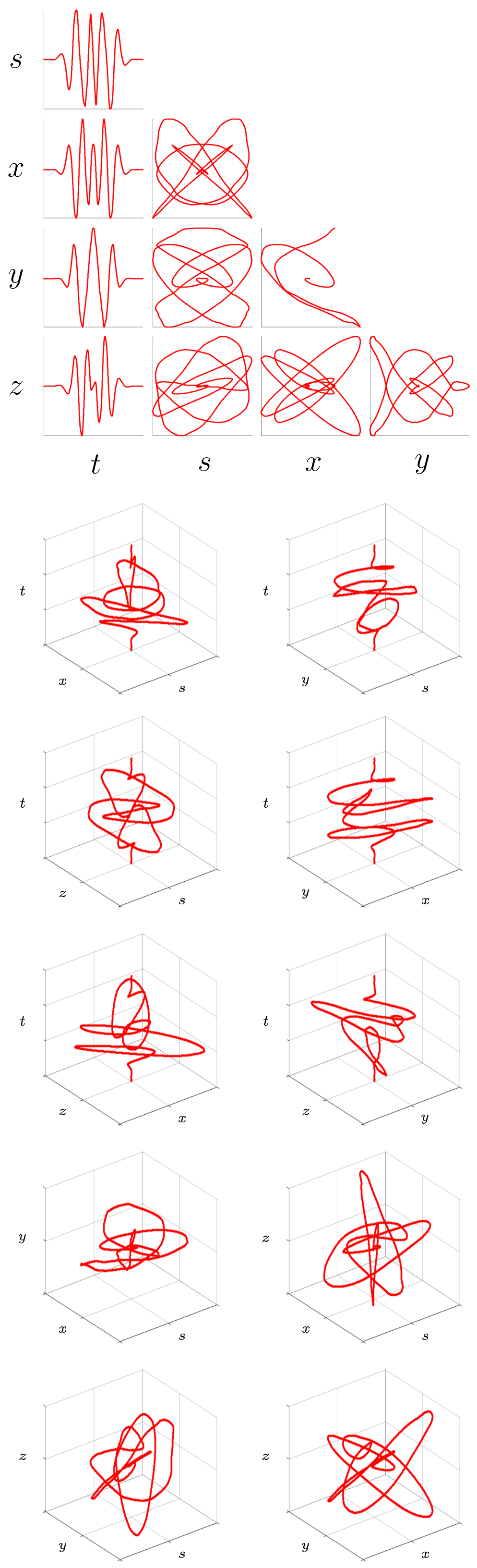


(b) $\Psi(t)$

Figure 5.7: Projections of Clifford scaling and wavelet functions onto 2-D and 3-D using our length 10 $Cl(1,1)$ filters.



(a) $\Phi(t)$



(b) $\Psi(t)$

Figure 5.8: Projections of Clifford scaling and wavelet functions onto 2-D and 3-D using our first length 10 $Cl(2,0)$ filters.

5.4 Frequency Response of Hypercomplex Filters

A common way of characterising a real digital filter is by its frequency response, defined as the spectrum of the output signal divided by the spectrum of the input signal. The input signal needs to contain as many frequencies as possible, hence an impulse would be used. We don't actually need to perform this experiment and calculation because the frequency response is equivalently given by the filter's transfer function $H(z)$ evaluated on the unit circle in the z plane, *i.e.* $H(\mathrm{e}^{\mathrm{j}\omega T})$, j being the complex $\sqrt{-1}$, ω the angular frequency and T the time. The transfer function is defined in terms of the filter's coefficients. MATLAB® provides a function `freqz`[1] that may be used to calculate a vector of samples from the impulse response of a filter thus:

```
[LoD,HiD] = wfilters('db3');

[H1,W1] = freqz(LoD);

figure
plot(W1,abs(H1),'b','Linewidth',2);
hold on

[H1,W1] = freqz(HiD);

plot(W1,abs(H1),'r','Linewidth',2);
hold off
xlim([0 3.1]);
```

Here we have taken the Daubechies length 6 scaling and wavelet filters `LoD` and `HiD` and found and plotted the impulse response of each filter together on the same graph. The result is illustrated in the top left of Figure 5.9. We have also shown similar plots in this figure for lengths 10, 14 and 18 for comparison with each other and with our later plots in this section. For each one, the response of the scaling filter is red and that of the wavelet filter

[1]https://uk.mathworks.com/help/signal/ref/freqz.html

is blue. As we expect, the scaling filters are low-pass and the wavelet filters are high-pass. The increasing flatness of the responses as the lengths of the filters increase is apparent.

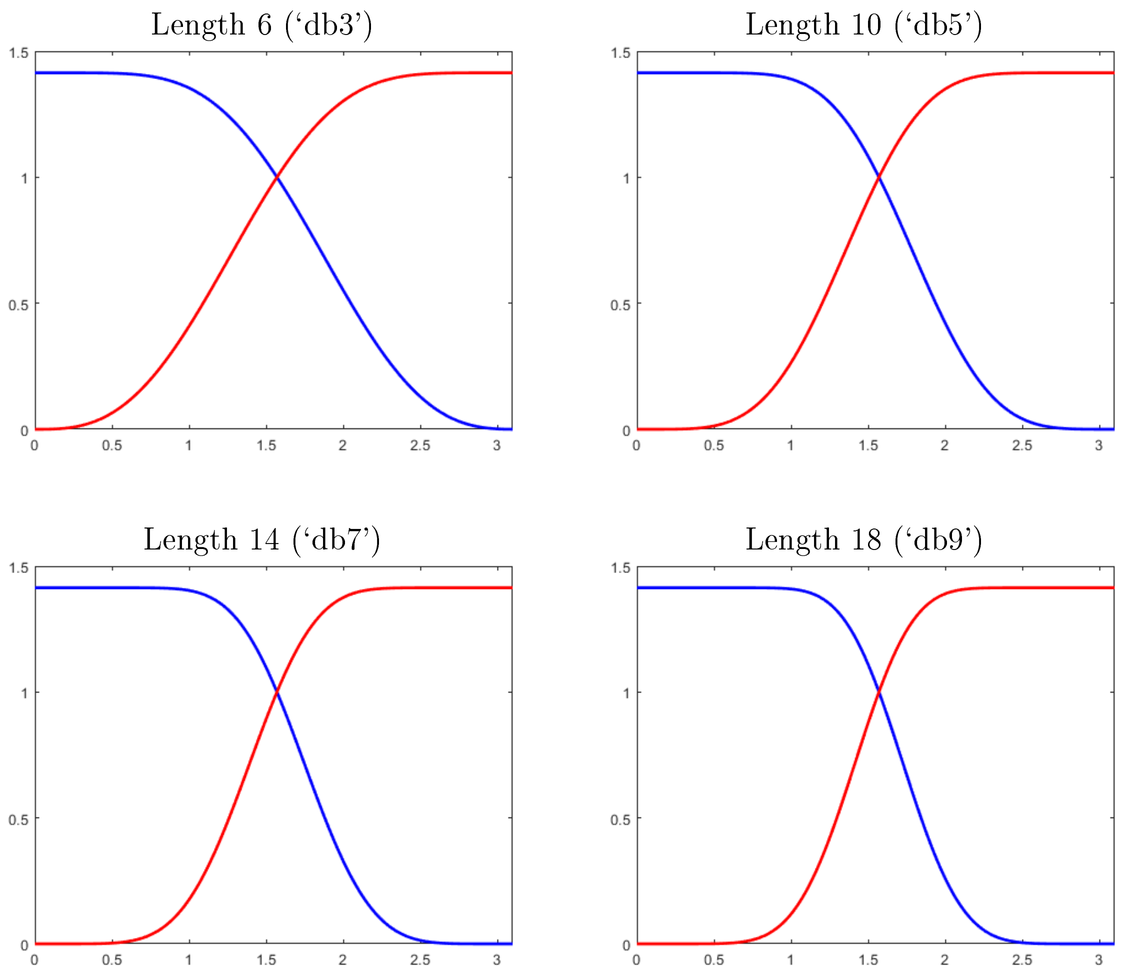


Figure 5.9: Frequency responses of a few of Daubechies' real filters.

We now use the above code, but with our own filter values for `LoD` and `HiD`.

Let $\{\mathtt{a1}, \mathtt{a2}\}$ each be vectors of the real coefficients of $\{\mathbf{e}_0, \mathbf{e}_1\}$, similarly with $\{\mathtt{a1}, \mathtt{a2}, \mathtt{a3}, \mathtt{a4}\}$ and $\{s, \mathbf{i}, \mathbf{j}, \mathbf{k}\}$ or $\{\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_{12}\}$, in our scaling and wavelet filters. In the left-hand plots of Figures 5.10 to 5.15, we show the frequency responses of our $Cl(1,0)$ filters, plotting the responses of `a1` and `a2` separately against `sqrt(abs(a1).^2+abs(a2).^2)`. The responses of the `a1` parts of the scaling filters are blue and those of the `a2` parts are light blue; and the responses of the `a1` parts of the wavelet filters are red and those of the `a2` parts are pink. In the right-hand plots we show the frequency responses of `sqrt(abs(a1).^2+abs(a2).^2)` against `sqrt(abs(a1).^2+abs(a2).^2)`. The responses of the scaling filter are blue and those of the wavelet filter are red. In Figures 5.16 to 5.24 we show the frequency responses of our quaternion, $Cl(1,1)$ and $Cl(2,0)$ filters, plotting the responses of $\{$`a1`, `a2`, `a3`, `a4`$\}$ separately against `sqrt(abs(a1).^2+abs(a2).^2+abs(a3).^2+abs(a4).^2)` in the left-hand plots and the responses of `sqrt(abs(a1).^2+abs(a2).^2+abs(a3).^2+abs(a4).^2)` against `sqrt(abs(a1).^2+abs(a2).^2+abs(a3).^2+abs(a4).^2)` in the right-hand plots. In the left-hand plots, we show the responses of `a3` and `a4` in lighter shades of red or blue.

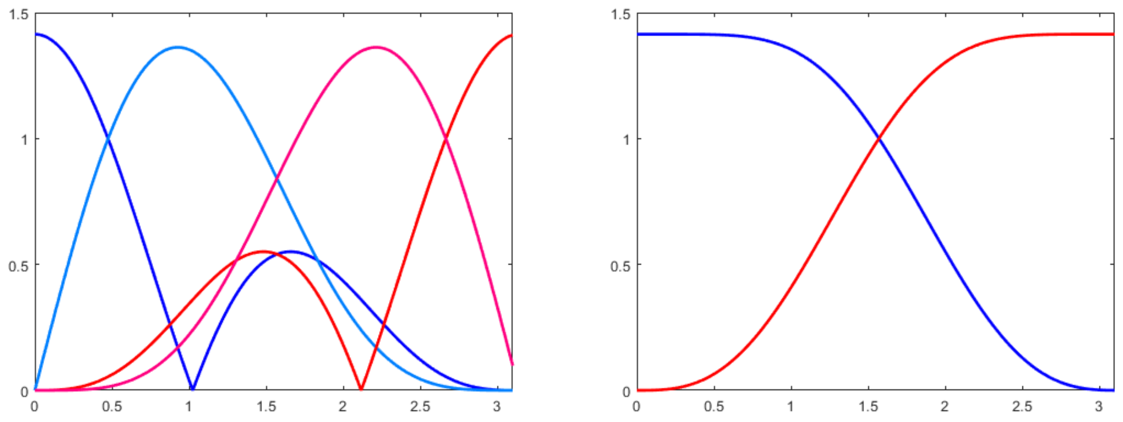


Figure 5.10: Frequency response of our length 6 $Cl(1,0)$ filters.

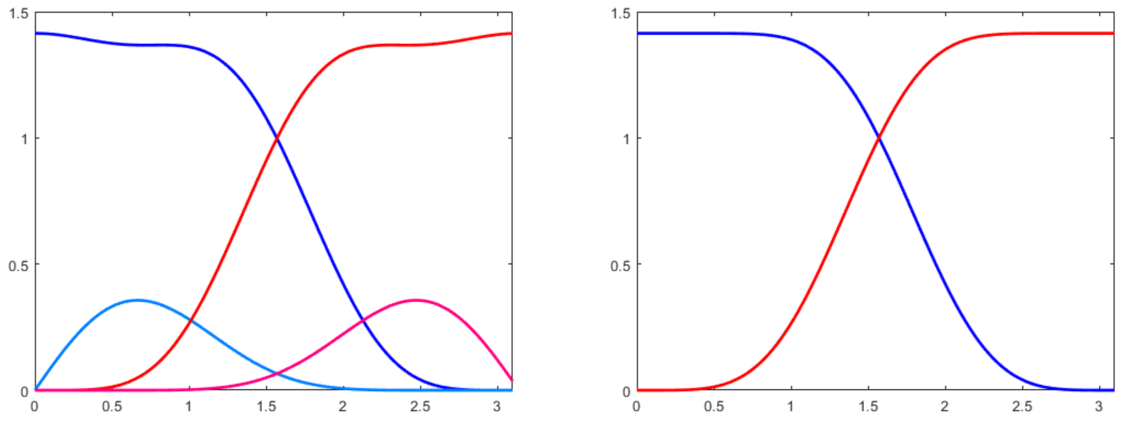


Figure 5.11: Frequency response of our first length 10 $Cl(1,0)$ filters.

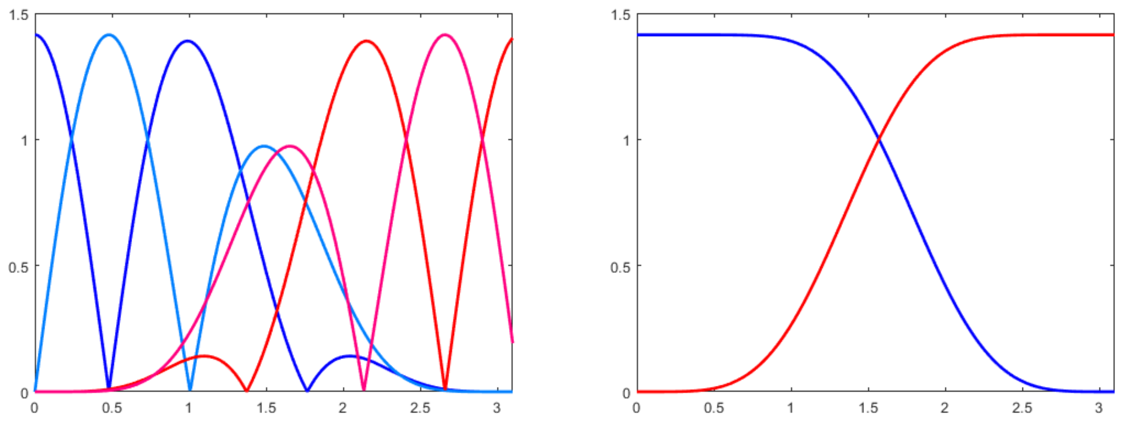


Figure 5.12: Frequency response of our second length 10 $Cl(1,0)$ filters.

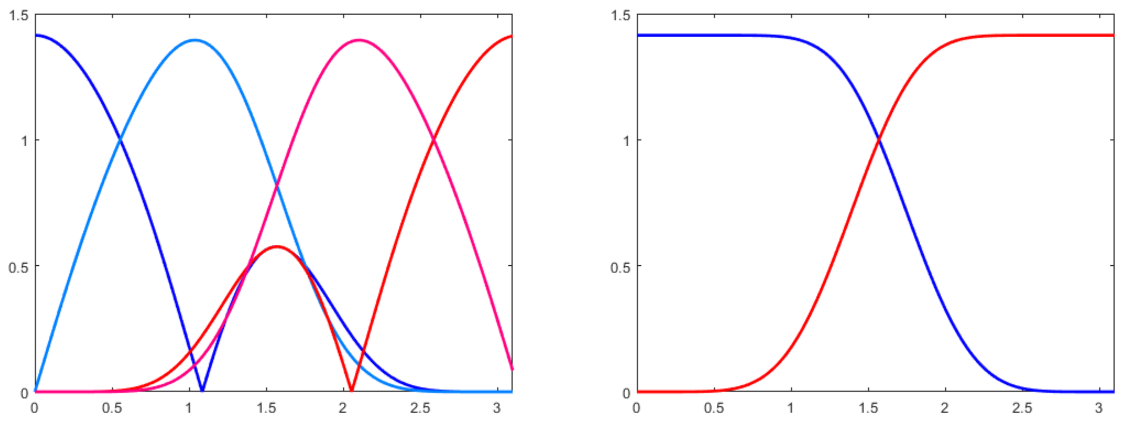


Figure 5.13: Frequency response of our first length 14 $Cl(1,0)$ filters.

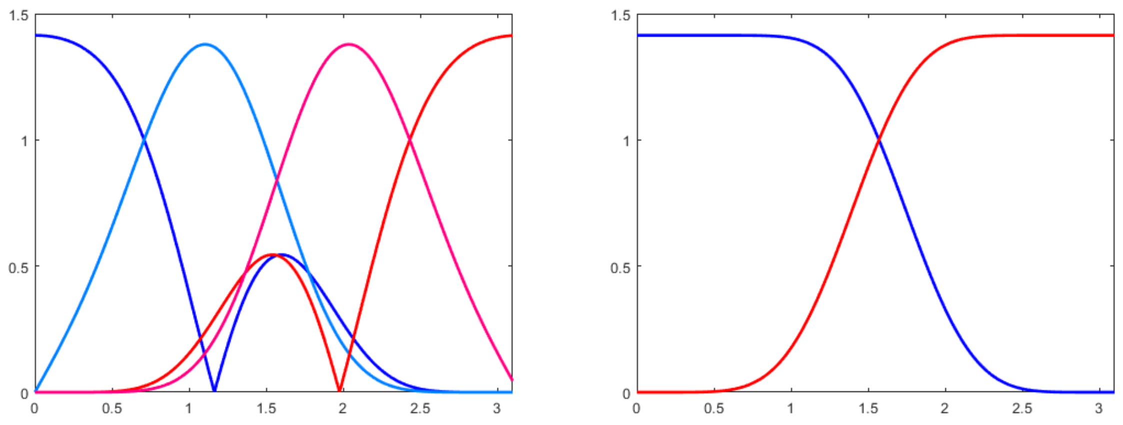


Figure 5.14: Frequency response of our second length 14 $Cl(1,0)$ filters.

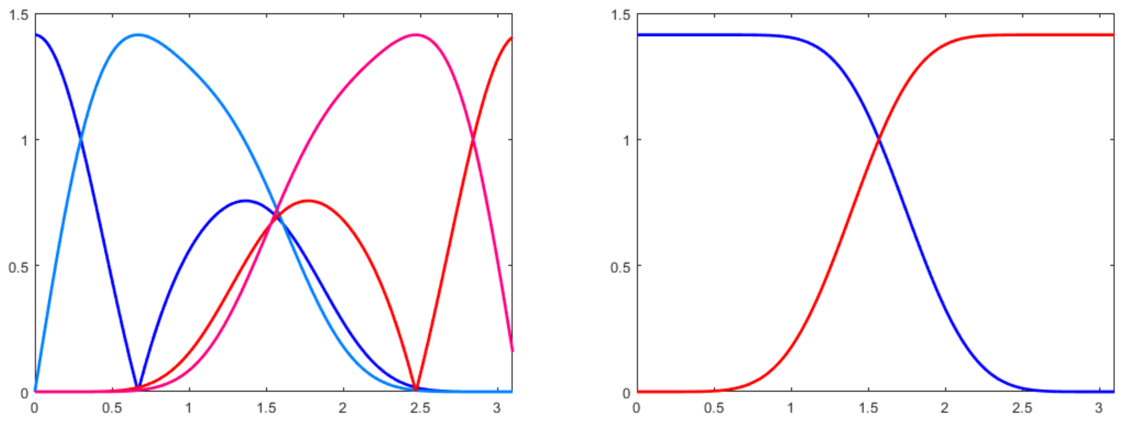


Figure 5.15: Frequency response of our third length 14 $Cl(1,0)$ filters.

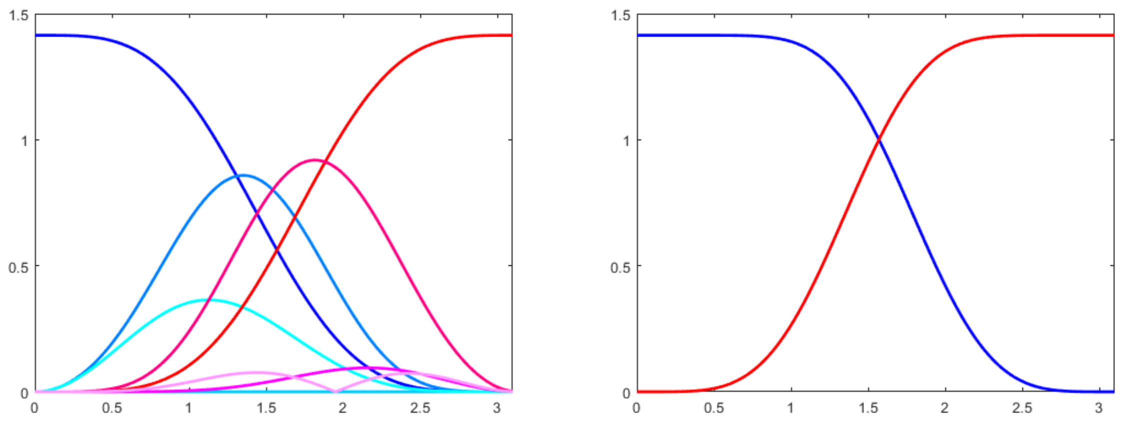


Figure 5.16: Frequency response of Ginzberg's length 10 quaternion filters.

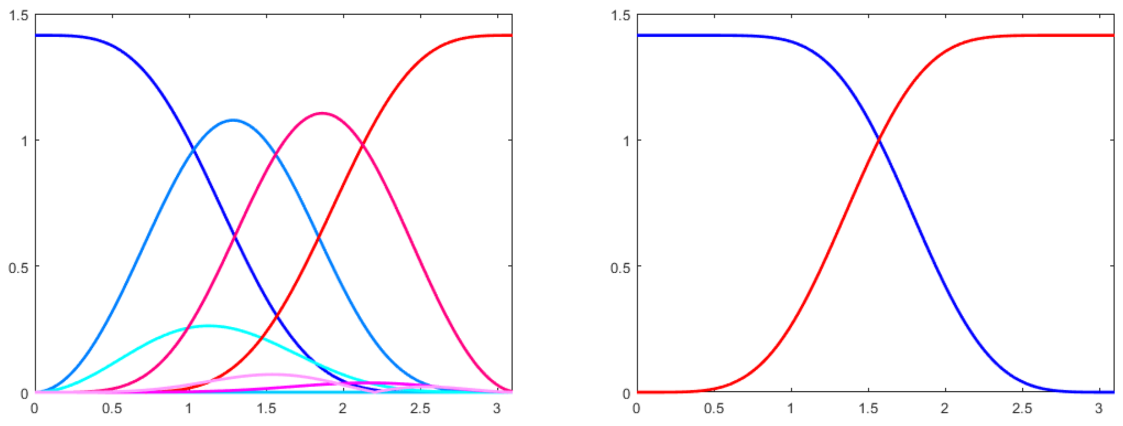


Figure 5.17: Frequency response of our length 10 quaternion filters.

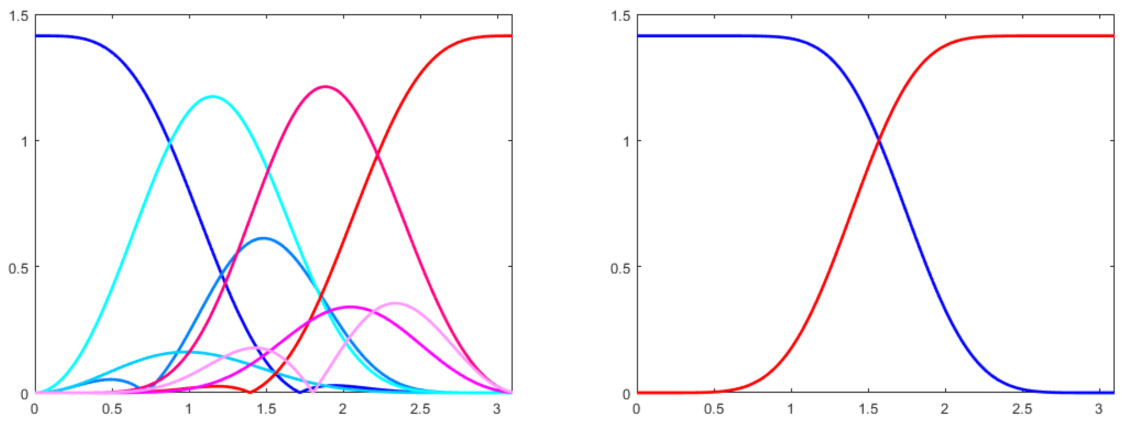


Figure 5.18: Frequency response of our first length 14 quaternion filters.

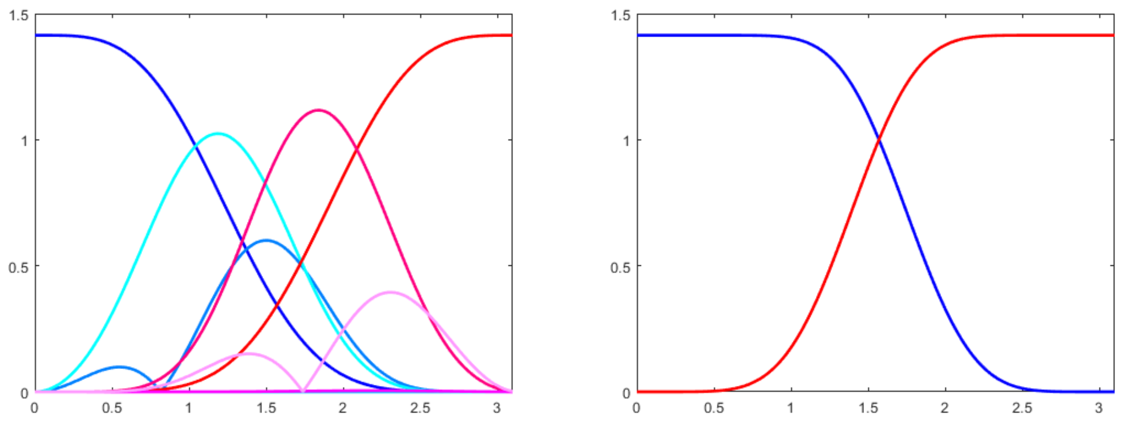


Figure 5.19: Frequency response of our second length 14 quaternion filters.

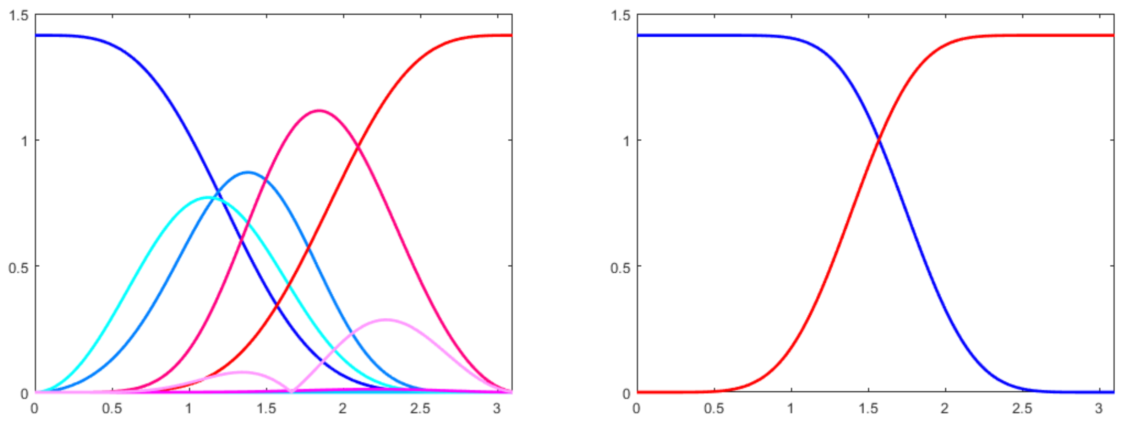


Figure 5.20: Frequency response of our third length 14 quaternion filters.

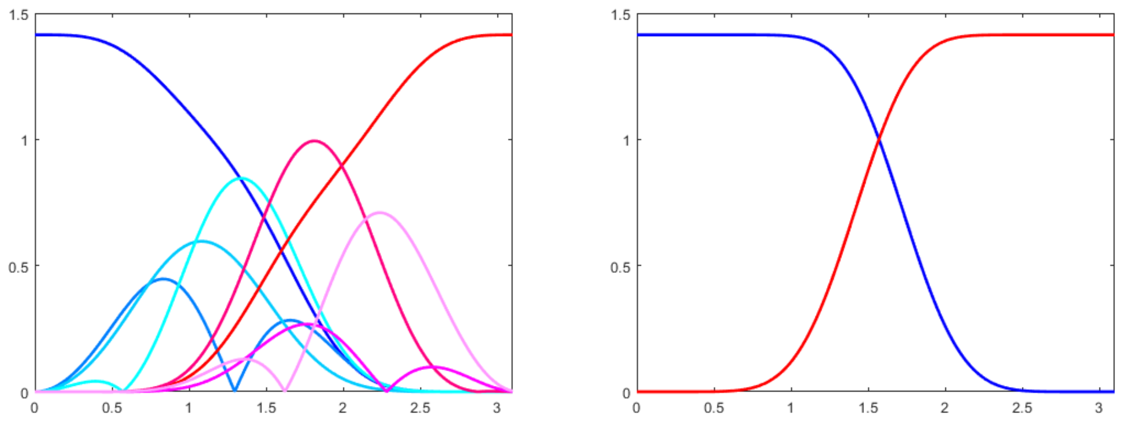


Figure 5.21: Frequency response of our length 18 quaternion filters.

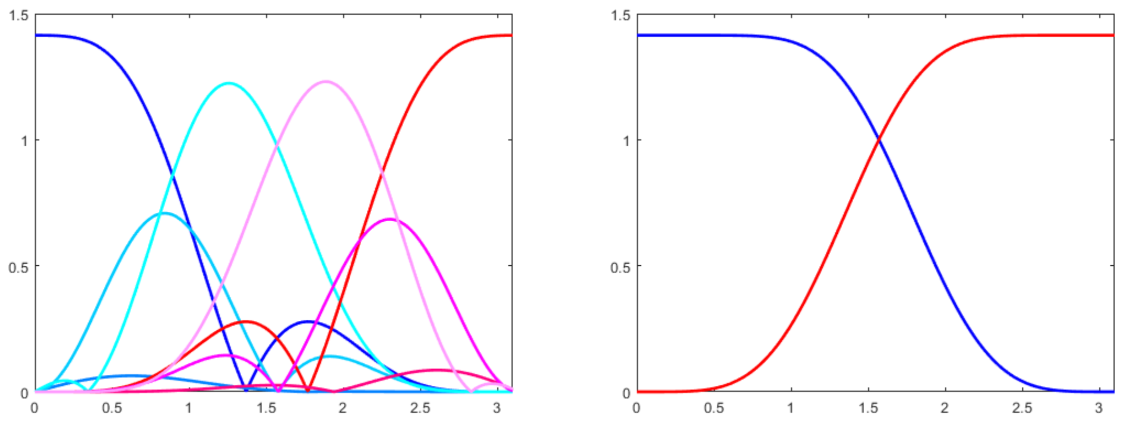


Figure 5.22: Frequency response of our length 10 $Cl(1,1)$ filters.

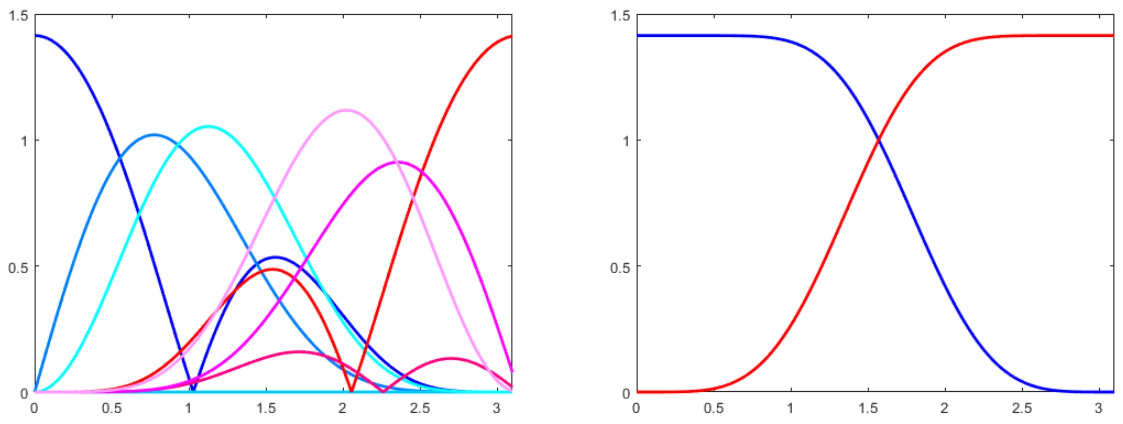


Figure 5.23: Frequency response of our first length 10 $Cl(2,0)$ filters.

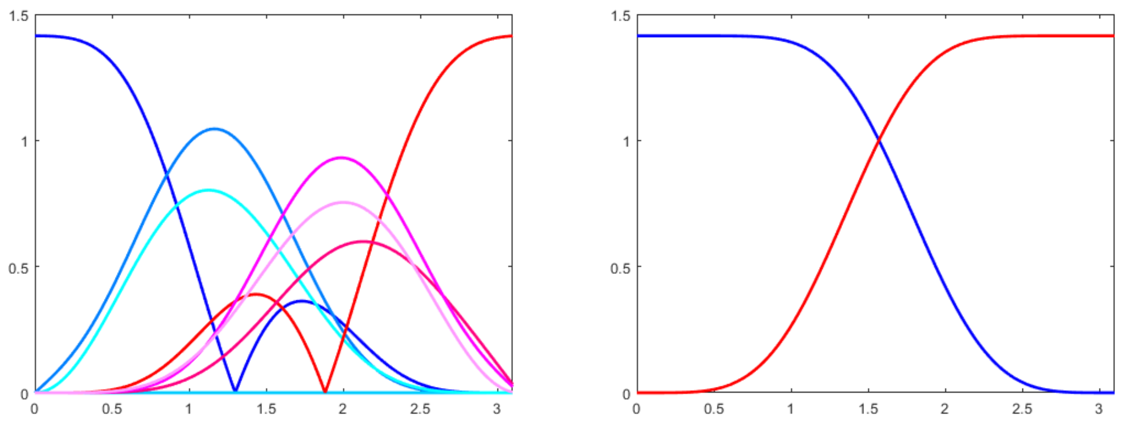


Figure 5.24: Frequency response of our second length 10 $Cl(2,0)$ filters.

5.5 Discussion

As would be expected, the scaling and wavelet functions have finite support, that is they are of zero amplitude outside a limited range of time/index values. Further, also as expected, they have greater amplitude in the middle of the support, decaying smoothly either side.

The complex and quaternion scaling and wavelet functions show oscillatory form of a particular type in the projections onto three dimensions; this is more apparent in the higher-order cases. The functions oscillate approximately helically in one sense, as they build up to maximum amplitude, then oscillate approximately helically in the opposite sense (which could be expressed as unwinding) as they decay. This is not unexpected, since a wavelet has a notional 'frequency' content, which is represented by the helical oscillation. For example, Figure 5.3 shows a complex case length 14 and it is readily seen that both the scaling and wavelet functions exhibit the helical oscillatory behaviour just outlined in the three dimensional plots. Figure 5.5 for a quaternion case length 14 shows similar behaviour in the s-x-t, s-y-t, s-z-t, x-z-t and y-z-t scaling function projections and the s-z-t, x-y-t, x-z-t and y-z-t

wavelet function projections. We surmise that the complex and quaternion cases would be very similar if the latter were plotted with the correct (as yet unknown) projection.

In the Clifford (*i.e.* not complex or quaternion) cases, for example Figure 5.8 for $Cl(2, 0)$, it is clear that the oscillatory behaviour is not as simple as in the complex and quaternion cases. Note that the scalar and bivector parts of the scaling function, plotted in two dimensions against t at the top left of the figure, are symmetric, whereas the two vector parts are antisymmetric; and that this is reversed in the wavelet function plots against t at the top right. We have already explained this earlier with reference to the scaling and wavelet filters. A problem here is that the theory of a Clifford multivector signal is not yet known: what does it mean to have a discrete-time signal with Clifford multivectors as samples? The different grades of a multivector (scalar, vector, bivector, *etc.*) are different types of mathematical object. Does it make sense to plot these components as if they were numeric co-ordinates in an n-dimensional Euclidean space, as we have done? What else could we do to visualise them? For higher-order algebras, for example with a three dimensional vector part, we could plot the separate grades independently, using projections where the grade has more than three numeric components. This would result in a larger number of plots, but it might be more physically meaningful. The quaternions are isomorphic to $Cl(0, 2)$ but there is no problem treating $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$ equally, because they are also isomorphic to the even subalgebra of $Cl(0, 3)$, *i.e.* a set of three bivectors, each of which square to $-\mathbf{e}_0$.

The frequency responses of our filters, as we have calculated and plotted them on the right of Figures 5.10 to 5.24, are virtually identical to those of the real Daubechies filters in Figure 5.9, for each length of filter. This is evidence that our filters are indeed genuine Daubechies hypercomplex scaling and wavelet filters and therefore that their use in filtering hypercomplex signals, using an arrangement such as that in Figure 3.2 for example, would indeed result in a hypercomplex wavelet transform. The plots on the left of Figures 5.10 to 5.24 simply reflect the differences between different filters. However, some are perfectly symmetrical and some are not. Would this be related in some way to the ‘quality’ of each pair of filters? We have no way of quantifying filter quality and we cannot even say if this is a meaningful question to ask.

5.6 Chapter Summary

We extended Ginzberg’s work and found five more quaternion scaling filters as well as six Clifford $Cl(1,0)$, one $Cl(1,1)$ and two $Cl(2,0)$ scaling filters. From these and Zhang *et al.*’s complex filters, we found associated wavelet filters. We are not aware of any discrete Clifford scaling and wavelet filters ever having been found before. We listed the filter coefficients (except the complex ones) and used the cascade algorithm on some of the filters to find the related scaling and wavelet functions, which we illustrated using projections onto two and three dimensions. It was natural for Zhang *et al.* and Ginzberg to use the conventional two dimensional plot from real scaling and wavelet functions to illustrate theirs [166, p. 1043] and [61, p. 171], but we believe our method is more informative and that this is the first time that such plots have been used. Lastly, we found a way to show the frequency responses of our filters that made comparison with those of real Daubechies scaling and wavelet filters possible. These showed that our filters really are hypercomplex scaling and wavelet filters.

In the next chapter we use some of our filters to find the first ‘true’ quaternion wavelet transforms and the first Clifford wavelet transforms of colour vector images.

Chapter 6

Some Hypercomplex Wavelet Transforms

In this chapter we use some of our scaling and wavelet filters from Chapter 5, in the form of vectors of hypercomplex numbers, to find quaterion and Clifford wavelet transforms of four test images. Our aim is to demonstrate one potential use of these filters; they may find other applications later.

In Section 6.1 we explain how we apply our filters to colour vector images. In Section 6.2 we present our results, split into quaternion wavelet transforms, Clifford $Cl(1,1)$ wavelet transforms, Clifford $Cl(2,0)$ wavelet transforms and lastly real wavelet transforms for comparison. Within the hypercomplex wavelet transforms subsections, we use our length 10, our first length 14 and our length 18 quaternion filters; our length 10 $Cl(1,1)$ and our first length 10 $Cl(2,0)$ Clifford filters; and length 6 complex and our $Cl(1,0)$ filter values as appropriate in quaternion, $Cl(1,1)$ and $Cl(2,0)$ filters. We discuss our results in Section 6.3.

6.1 Hypercomplex Wavelet Transforms of Colour Vector Images

We begin by considering exactly how to apply scaling and wavelet filters to an array of pixels. The obvious way of doing this is to arrange the filters in a square banded matrix, the same size as the (square) image whose wavelet transform we wish to find, and multiply. Then the question arises, exactly how should the filters be arranged? Is there a 'best' way? There is a limited number of possibilities for the design of a banded matrix, but Strang [140] did indeed find what we consider to be the 'best' way. We illustrate this for length 6 filters:

$$\begin{pmatrix}
w_4 & w_5 & w_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\
s_2 & s_3 & s_4 & s_5 & s_6 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\
w_2 & w_3 & w_4 & w_5 & w_6 & 0 & 0 & 0 & 0 & 0 & 0 & \\
0 & s_1 & s_2 & s_3 & s_4 & s_5 & s_6 & 0 & 0 & 0 & 0 & \cdots \\
0 & w_1 & w_2 & w_3 & w_4 & w_5 & w_6 & 0 & 0 & 0 & 0 & \\
0 & 0 & 0 & s_1 & s_2 & s_3 & s_4 & s_5 & s_6 & 0 & 0 & \cdots \\
0 & 0 & 0 & w_1 & w_2 & w_3 & w_4 & w_5 & w_6 & 0 & 0 & \\
 & \vdots & & \vdots & & \vdots & & \vdots & & \vdots & & \ddots
\end{pmatrix},$$

with $s_1, s_2, \ldots, s_6$ being the scaling filter coefficients and $w_1, w_2, \ldots, w_6$ being the wavelet filter coefficients.

For general length $2n$ filters, we put the right-hand half of the wavelet filter on the top row of a square matrix of zeros, starting in the top left-hand corner. We then construct a $2 \times 2n$ submatrix, with the scaling filter on the top row and the wavelet filter is on the bottom, and place copies of this down the leading diagonal, each submatrix being offset two places to the right relative to the previous one. We truncate this submatrix where it would otherwise overlap the edge of our zero matrix. As we shall see, this arrangement lends itself to multiresolution analysis and automatically gives us the down-sampling by two that we expect from a wavelet transform.

The beauty of this particular banded matrix arrangement is that the inverse is also a banded matrix: if one requires the inverse scaling and wavelet filters, they may be read from the inverse matrix (away from the corners). In fact, with random numbers in place of the filter coefficients, the inverse matrix still has exactly the same pattern.

Let $\boldsymbol{Q}$ be our banded matrix with hypercomplex scaling and wavelet filters arranged as just described, $\boldsymbol{J}$ our colour vector image with each pixel scaled so that saturated red, green and blue would each have the value 1 and $\boldsymbol{u} = 1 + \mathbf{i} + \mathbf{j} + \mathbf{k}$ or $\boldsymbol{u} = \mathbf{e}_0 + \mathbf{e}_1 + \mathbf{e}_2 + \mathbf{e}_{12}$ as appropriate. We find the hypercomplex wavelet transform of our vector image thus:

$$\boldsymbol{R} = \boldsymbol{Q}(\boldsymbol{J} - 0.5\boldsymbol{u})\boldsymbol{Q}^H + 0.5\boldsymbol{u}, \tag{6.1}$$

where the superscript H denotes the conjugate transpose[1].

The $-0.5\boldsymbol{u}$ moves the origin of co-ordinates for each pixel to the centre of a four-dimensional unit hypercube. The idea is to confine as many of the quaternions or multivectors resulting from the matrix product to the interior of this hypercube: we actually find that a few end up outside this hypercube, so we truncate these ones. The $+0.5\boldsymbol{u}$ restores the origin to its correct position at the black vertex of the colour cube subspace.

Now consider what was involved in calculating each element of $\boldsymbol{R}$: in the following, we ignore the effect of edges.

Let $\boldsymbol{s}$ and $\boldsymbol{w}$ be length L hypercomplex scaling and wavelet filters respectively and let $\boldsymbol{p}_{u,v}$

[1]as per Wolfram MathWorld http://mathworld.wolfram.com/ConjugateTranspose.html.

be the length L column vector of (origin-adjusted) hypercomplex-valued pixels inside the image with its top pixel at (u, v). Then elements of $\boldsymbol{R}$ at positions $(2k+1, 2m+1)$ for integer k, m, *i.e.* those on odd rows and odd columns, are products of the form

$$\left(\boldsymbol{w} \bullet \boldsymbol{p}_{2k+1,2m+1}, \boldsymbol{w} \bullet \boldsymbol{p}_{2k+1,2m+2}, \ldots, \boldsymbol{w} \bullet \boldsymbol{p}_{2k+1,2m+L}\right) \bullet \overline{\boldsymbol{w}}^T;$$

elements of $\boldsymbol{R}$ at positions $(2k+1, 2m)$, *i.e.* those on odd rows and even columns, are products of the form

$$\left(\boldsymbol{w} \bullet \boldsymbol{p}_{2k+1,2m}, \boldsymbol{w} \bullet \boldsymbol{p}_{2k+1,2m+1}, \ldots, \boldsymbol{w} \bullet \boldsymbol{p}_{2k+1,2m+L-1}\right) \bullet \overline{\boldsymbol{s}}^T;$$

elements of $\boldsymbol{R}$ at positions $(2k, 2m+1)$, *i.e.* those on even rows and odd columns, are products of the form

$$\left(\boldsymbol{s} \bullet \boldsymbol{p}_{2k,2m+1}, \boldsymbol{s} \bullet \boldsymbol{p}_{2k,2m+2}, \ldots, \boldsymbol{s} \bullet \boldsymbol{p}_{2k,2m+L}\right) \bullet \overline{\boldsymbol{w}}^T;$$

and elements of $\boldsymbol{R}$ at positions $(2k, 2m)$, *i.e.* those on even rows and even columns, are products of the form

$$\left(\boldsymbol{s} \bullet \boldsymbol{p}_{2k,2m}, \boldsymbol{s} \bullet \boldsymbol{p}_{2k,2m+1}, \ldots, \boldsymbol{s} \bullet \boldsymbol{p}_{2k,2m+L-1}\right) \bullet \overline{\boldsymbol{s}}^T.$$

In the above, we used $\bullet$ to indicate the scalar product.

The multiplications to find quaternions or multivectors at the intersections of even-numbered rows and even-numbered columns of $\boldsymbol{R}$ only involved scaling filters and their conjugates. If (after restoring the origin) we extract these and view the scalar or vector part, we see an approximation to the original image. If we do the same with even-numbered rows and odd-numbered columns, these products involved only scaling filters and conjugates of wavelet filters and we see evidence of vertical detail. With odd-numbered rows and even-numbered columns, these products involved only wavelet filters and conjugates of scaling filters and we see evidence of horizontal detail. With odd-numbered rows and odd-numbered columns, these products involved only wavelet filters and their conjugates and we see evidence of diagonal detail. We summarise these observations in Table 6.1.

Table 6.1: Detail of the content of $\boldsymbol{R}$.

Rows of $\boldsymbol{Q}$	Columns of $\boldsymbol{Q}^H$	Element of $\boldsymbol{R}$
Even, $\boldsymbol{s}$	Even, $\overline{\boldsymbol{s}}$	Approximation
Odd, $\boldsymbol{w}$	Even, $\overline{\boldsymbol{s}}$	Horizontal detail
Even, $\boldsymbol{s}$	Odd, $\overline{\boldsymbol{w}}$	Vertical detail
Odd, $\boldsymbol{w}$	Odd, $\overline{\boldsymbol{w}}$	Diagonal detail

The $\boldsymbol{R}$ in Equation (6.1) gives the first level of analysis. If we replace $\boldsymbol{J}$ with the approximation from $\boldsymbol{R}$, we can perform a second level of analysis (with a new $\boldsymbol{Q}$ matrix a quarter of the size of the first one). And we can do the same with the second approximation, to perform a third level of analysis. Every time we do this, the approximation and three detail image results are each a quarter of the starting image in size: we assume that the sidelength

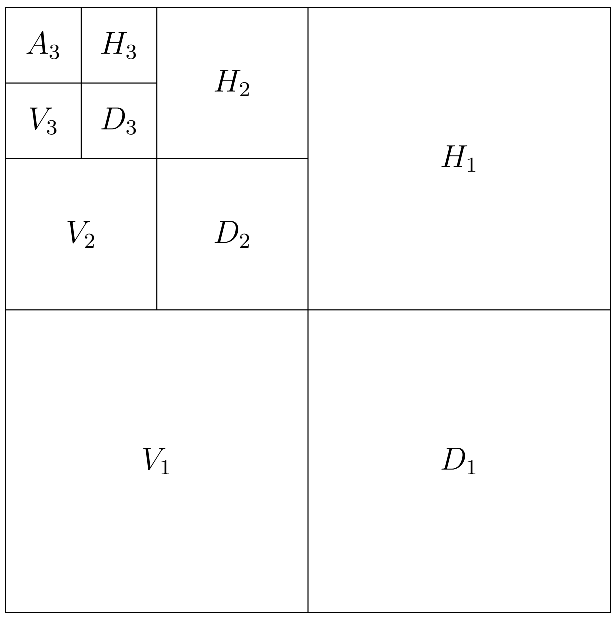


Figure 6.1: Display arrangement for our results.

of the original square image is a power of two. In this way, we perform a two-dimensional multiresolution analysis of the image. We could continue like this down to the last four pixels, but this would not be informative due to the results being swamped by edge effects well before we reached that level. We shall settle for three levels of analysis of 512×512 test images and use the conventional format of wavelet transforms of images as in Figure 6.1[2]: A is for approximation, H for horizontal detail, V for vertical detail and D for diagonal detail with the subscripts indicating the level of analysis.

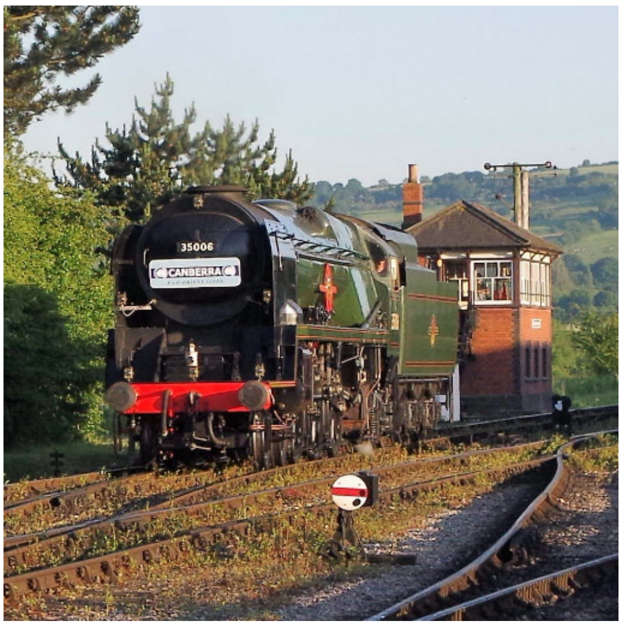


Figure 6.2: Our fourth test image, which we shall refer to as P & O.

We shall use four test images to illustrate the use of our filters: Lena, the mandrill, the sailboat on lake (which we call a yacht) and Figure 6.2. The last of these is Southern Railway Merchant Navy class 4-6-2 *no.* 35006, ‘Peninsular & Oriental S. N. Co.’, with a sponsored headboard, at Toddington on the Gloucestershire Warwickshire Railway in 2017, courtesy of Pete Collings, c/o the 35006 Locomotive Society.

[2]This arrangement appears to have been introduced by Stéphane Mallat in 1989 [107] as a means of analysing the information content of images.

6.2 Results

The original test images start as arrays of pure hypercomplex numbers (one with no scalar components), but the wavelet transforms are arrays of full hypercomplex numbers. We use the approximations in this form as the inputs for subsequent transforms and where we use a full quaternion or Clifford filter, we display the scalar and vector parts of the detail in two copies of Figure 6.1, side by side. We also give the standard deviation (σ), skewness (γ_1) and kurtosis (β_2) of the red component in the horizontal level one detail and show a third copy of Figure 6.1, but with our estimate of the amount of detail that our wavelet transform has extracted from the starting image, at each level and orientation. For our estimates of the detail, we find the length of the deviation from the mean in four dimensions by calculating $\sqrt{(s-\overline{s})^2+(x-\overline{x})^2+(y-\overline{y})^2+(z-\overline{z})^2}$ for each pixel and summing these lengths over all pixels in the starting image and each approximation and detail region. We know that the first level approximation and detail regions must contain exactly the same amount of detail in total as the starting image, so we force this to be the case in terms of our detail estimates by scaling their lengths appropriately. We do the same with the second and third level approximation and detail regions, using the first and second level approximations respectively as the starting images. Finally we express the detail in each analysed region as a percentage of that in the original image. Where we do not use a full quaternion or Clifford filter, we only show a copy of Figure 6.1 with the vector part of the wavelet transform. An example of our code is in Appendix F.

6.2.1 Quaternion Wavelet Transforms

6.2.1.1 Full Quaternions

In Figures 6.3 to 6.26, we use our length 10, first length 14 and length 18 quaternion filters.

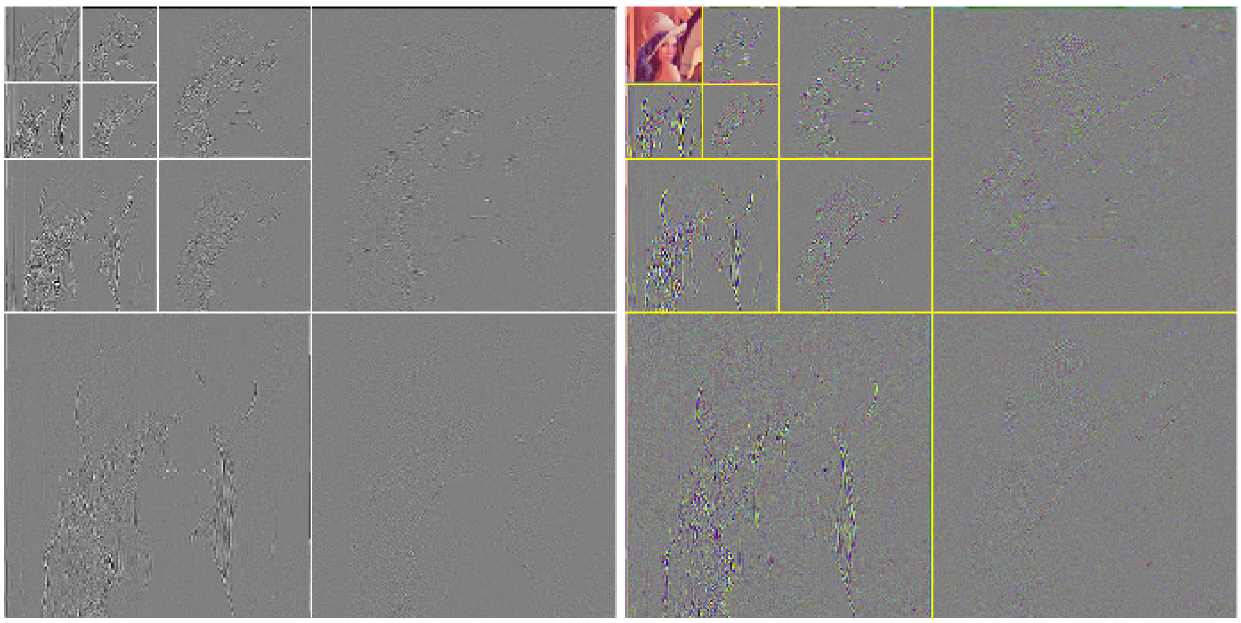


Figure 6.3: QWT of Lena as a colour vector image, using our length 10 full quaternion filters.

In the red component of the level one horizontal detail, $\sigma = 0.0083, \gamma_1 = -0.2992$ and $\beta_2 = 35.0635$.

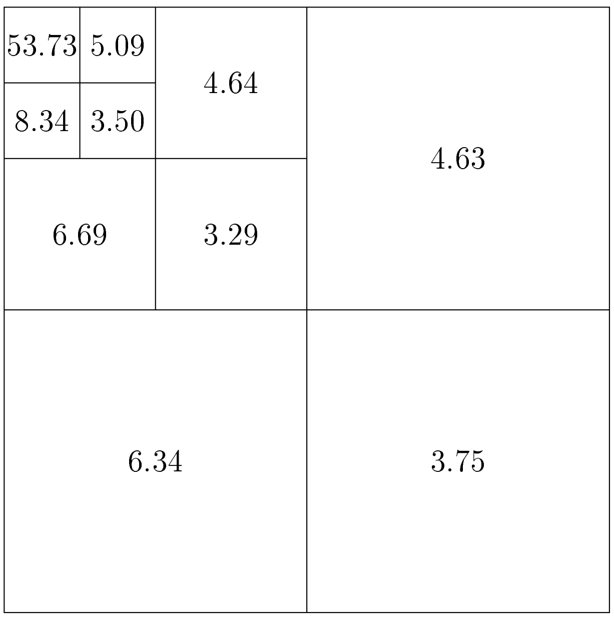


Figure 6.4: Analysis of extracted detail, 46.27% of that in the original image.

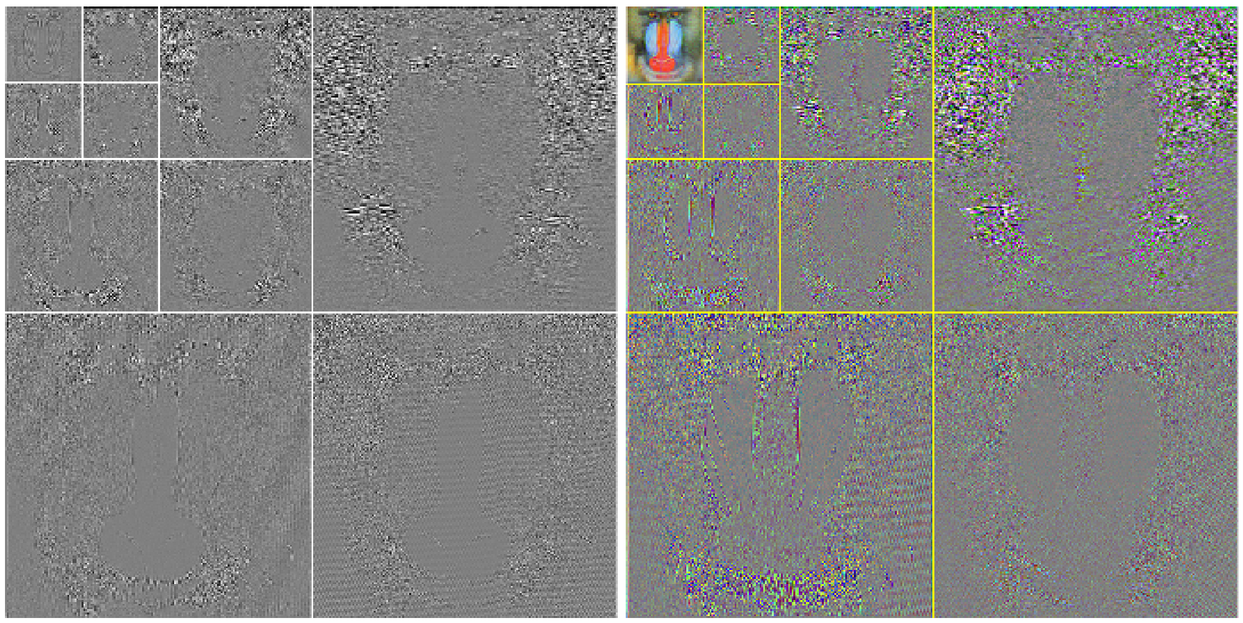


Figure 6.5: QWT of the mandrill as a colour vector image, using our length 10 full quaternion filters.

In the red component of the level one horizontal detail, $\sigma = 0.0361, \gamma_1 = 0.0164$ and $\beta_2 = 7.1681$.

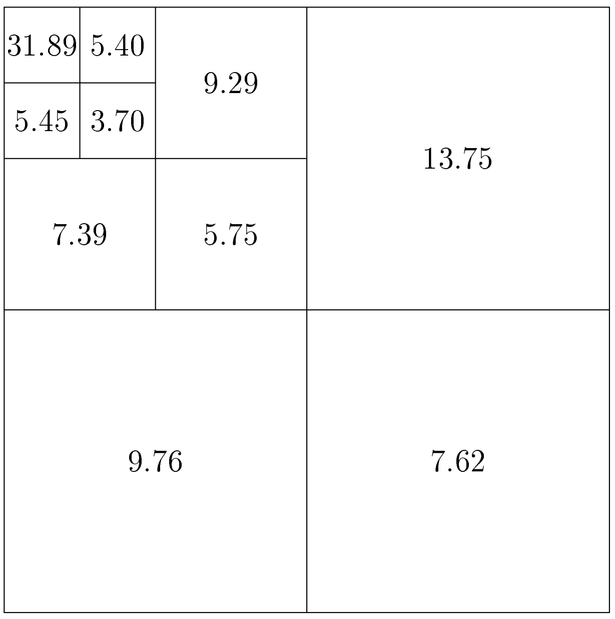


Figure 6.6: Analysis of extracted detail, 68.11% of that in the original image.

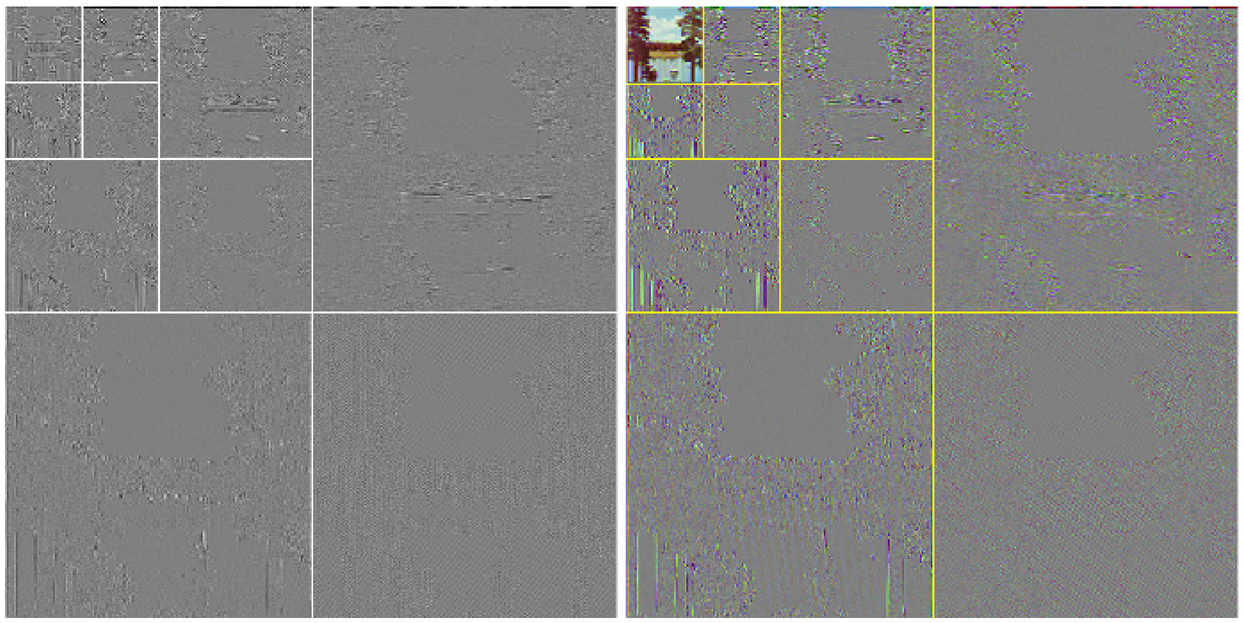


Figure 6.7: QWT of the yacht as a colour vector image, using our length 10 full quaternion filters.

In the red component of the level one horizontal detail, $\sigma = 0.0164, \gamma_1 = -3.0960$ and $\beta_2 = 54.6000$.

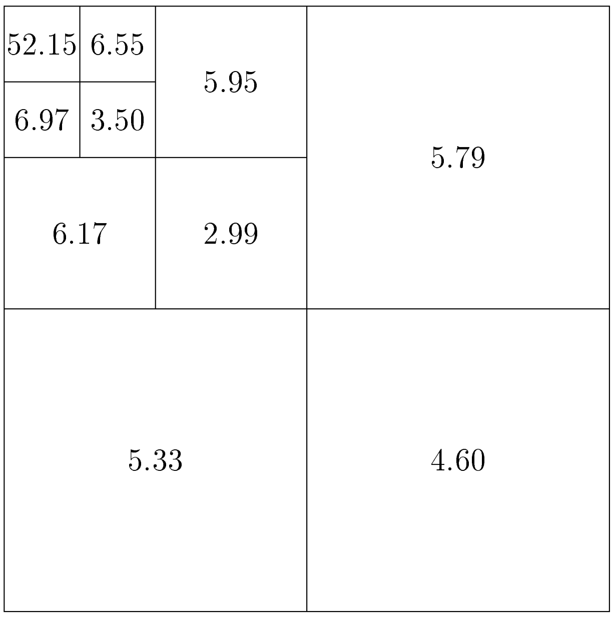


Figure 6.8: Analysis of extracted detail, 47.85% of that in the original image.

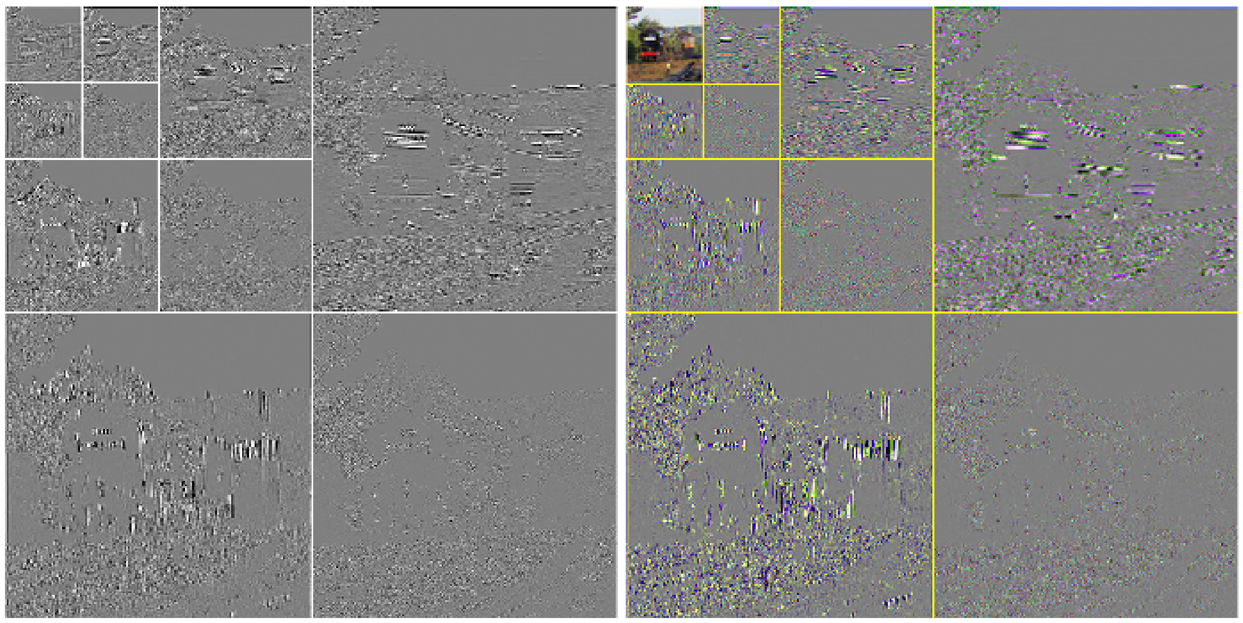


Figure 6.9: QWT of P & O as a colour vector image, using our length 10 full quaternion filters.

In the red component of the level one horizontal detail, $\sigma = 0.0369, \gamma_1 = -0.0398$ and $\beta_2 = 12.2762$.

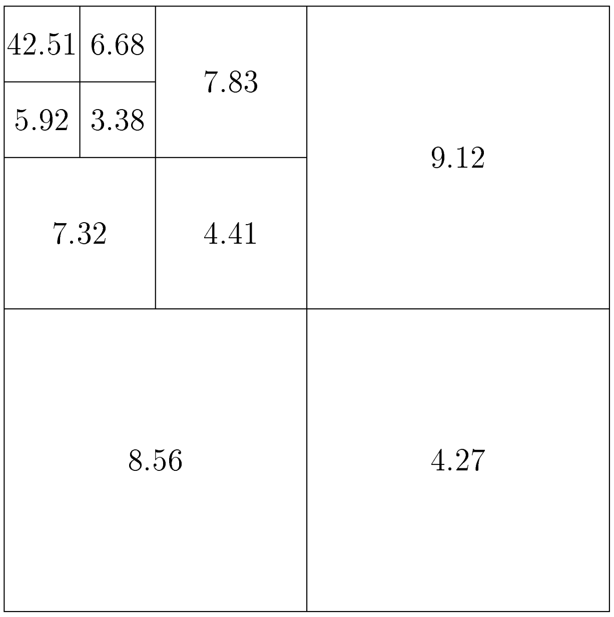


Figure 6.10: Analysis of extracted detail, 57.49% of that in the original image.

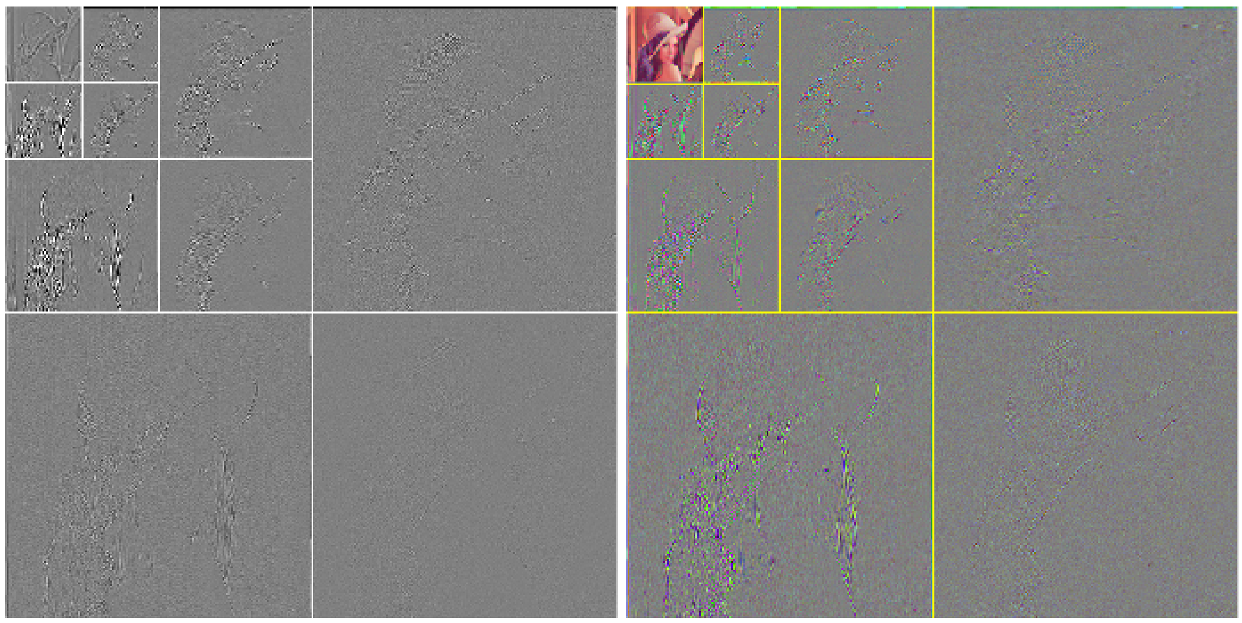


Figure 6.11: QWT of Lena as a colour vector image, using our first length 14 full quaternion filters.

In the red component of the level one horizontal detail, $\sigma = 0.0103, \gamma_1 = -0.0094$ and $\beta_2 = 23.7615$.

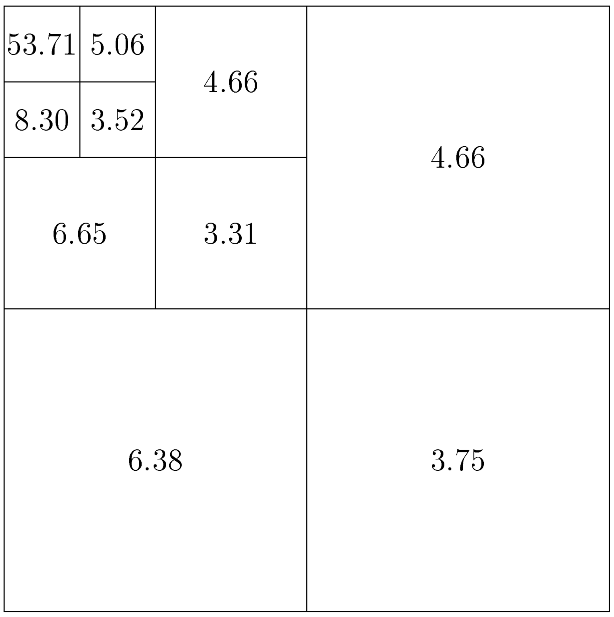


Figure 6.12: Analysis of extracted detail, 46.29% of that in the original image.

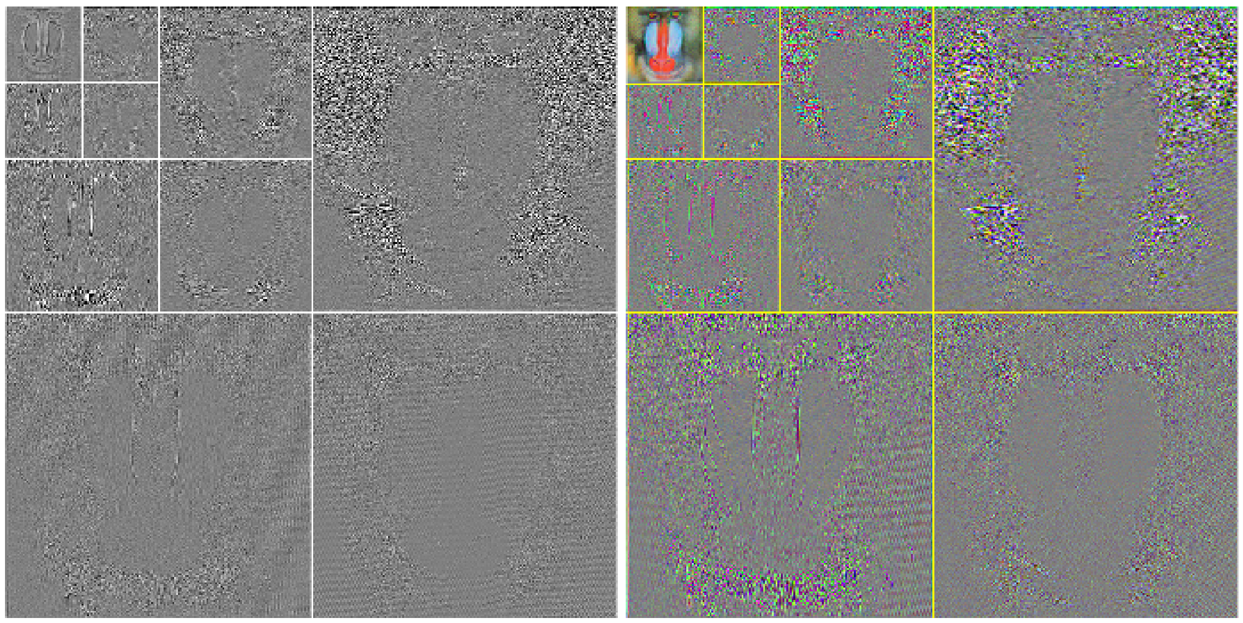


Figure 6.13: QWT of the mandrill as a colour vector image, using our first length 14 full quaternion filters.

In the red component of the level one horizontal detail, $\sigma = 0.0345, \gamma_1 = -0.0302$ and $\beta_2 = 6.3382$.

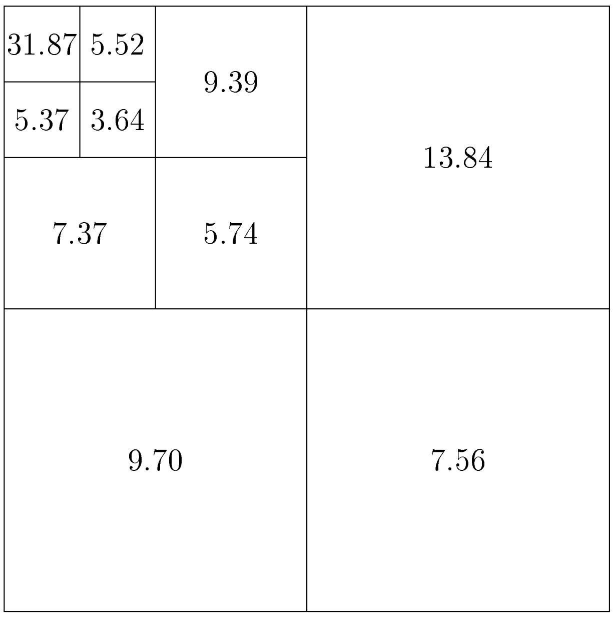


Figure 6.14: Analysis of extracted detail, 68.13% of that in the original image.

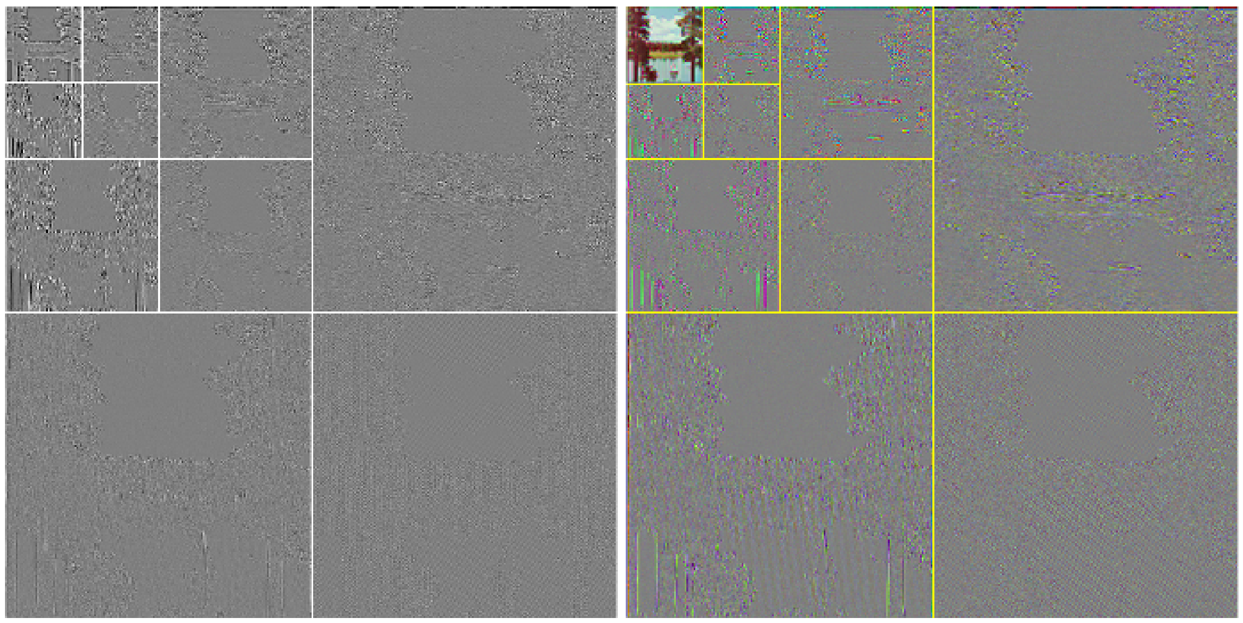


Figure 6.15: QWT of the yacht as a colour vector image, using our first length 14 full quaternion filters.

In the red component of the level one horizontal detail, $\sigma = 0.0179, \gamma_1 = -0.4586$ and $\beta_2 = 24.7724$.

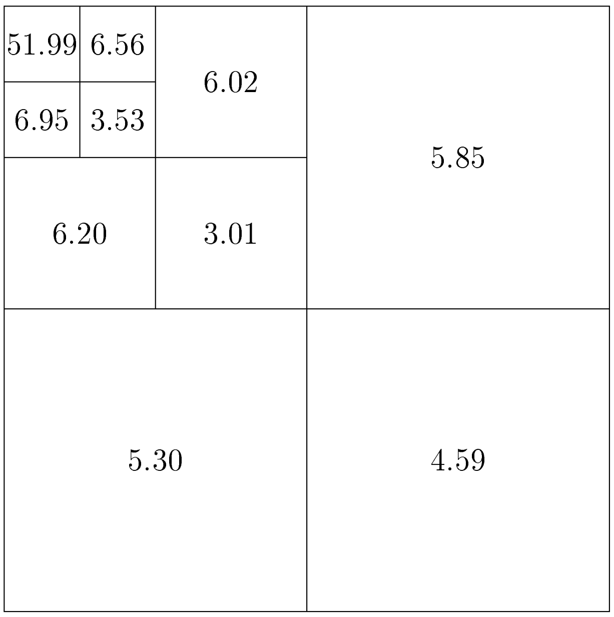


Figure 6.16: Analysis of extracted detail, 48.01% of that in the original image.

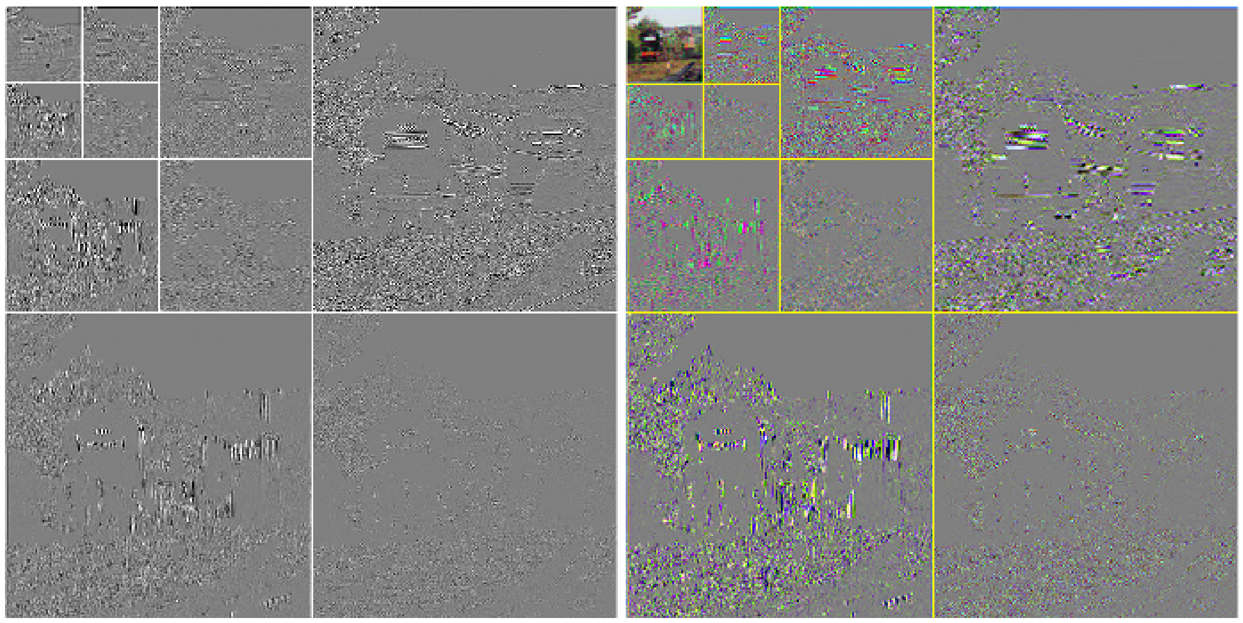


Figure 6.17: QWT of P & O as a colour vector image, using our first length 14 full quaternion filters.

In the red component of the level one horizontal detail, $\sigma = 0.0285, \gamma_1 = 0.0309$ and $\beta_2 = 10.3607$.

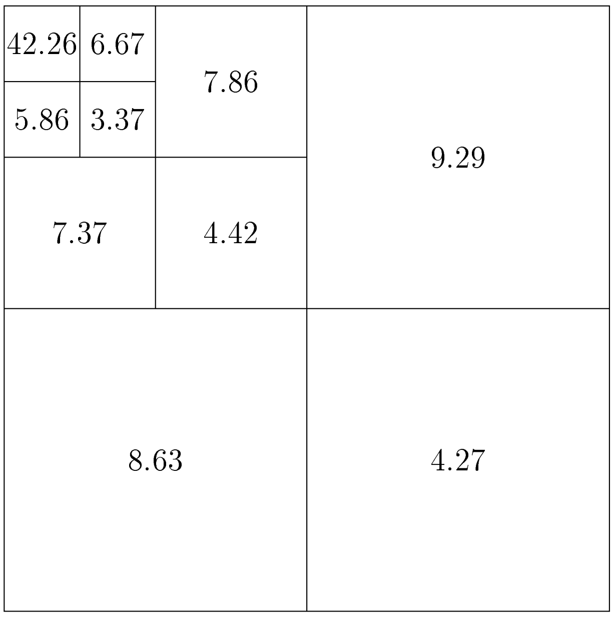


Figure 6.18: Analysis of extracted detail, 57.74% of that in the original image.

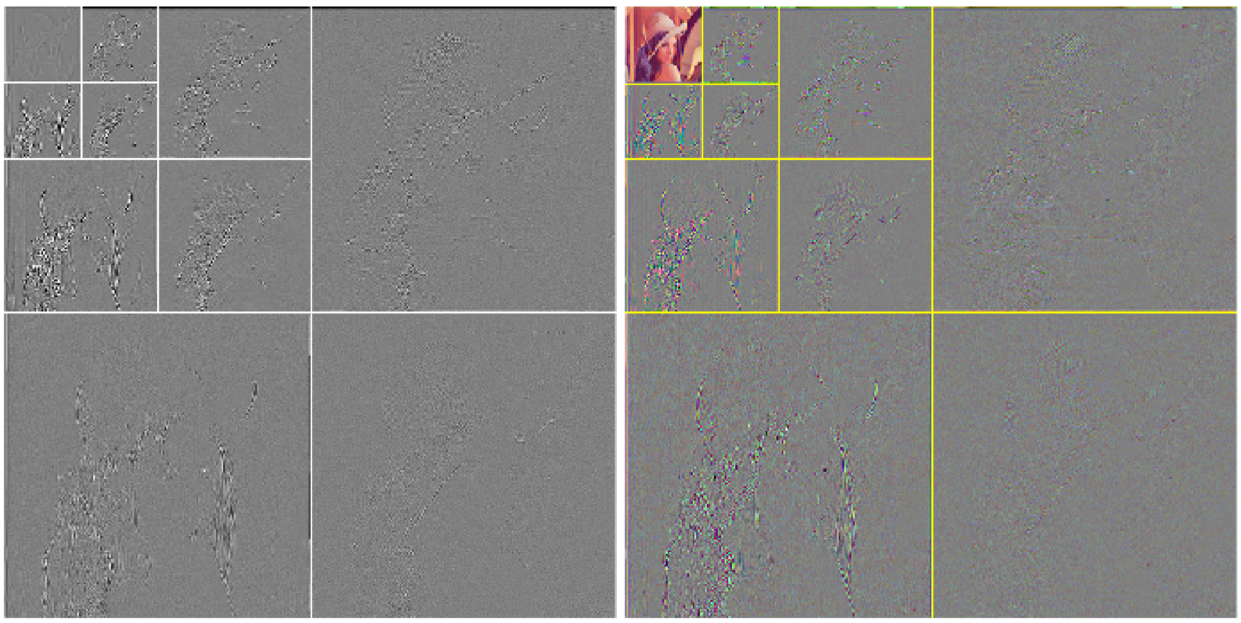


Figure 6.19: QWT of Lena as a colour vector image, using our length 18 full quaternion filters.

In the red component of the level one horizontal detail, $\sigma = 0.0084, \gamma_1 = -0.0091$ and $\beta_2 = 16.9072$.

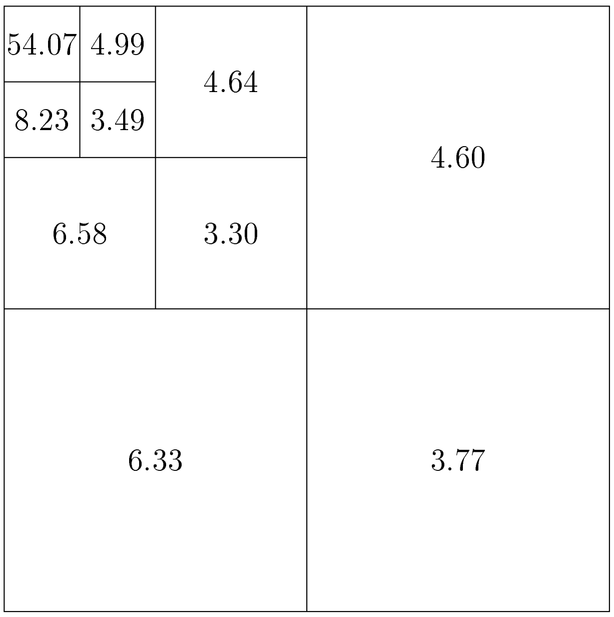


Figure 6.20: Analysis of extracted detail, 45.93% of that in the original image.

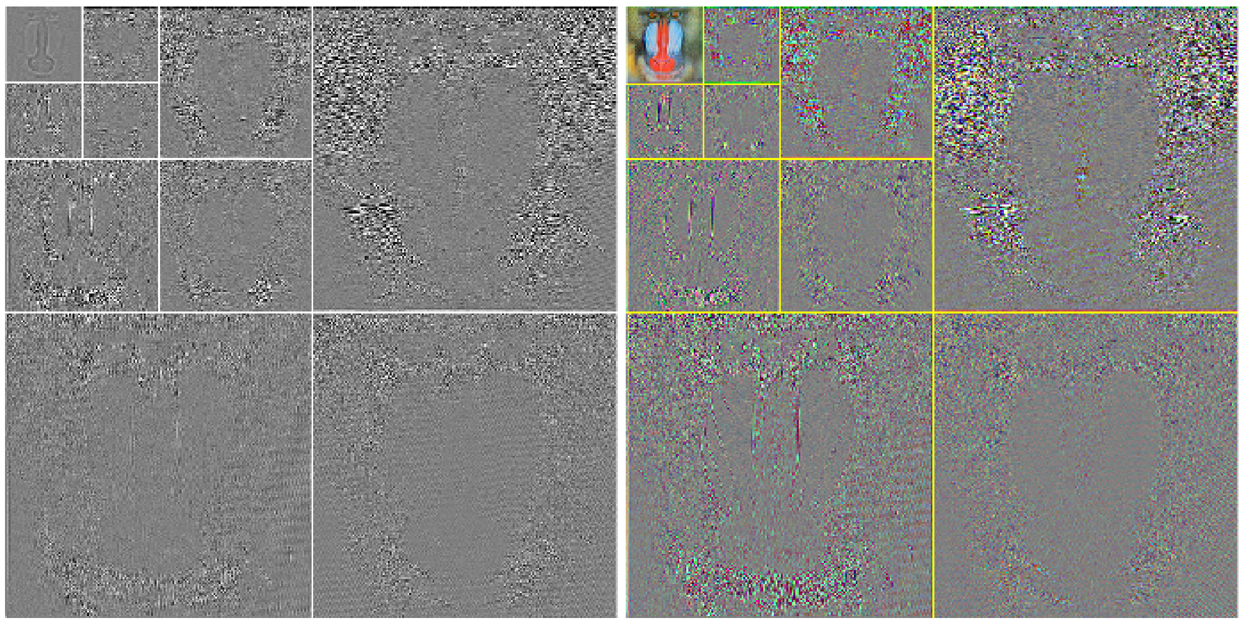


Figure 6.21: QWT of the mandrill as a colour vector image, using our length 18 full quaternion filters.

In the red component of the level one horizontal detail, $\sigma = 0.0372, \gamma_1 = 0.0266$ and $\beta_2 = 6.5159$.

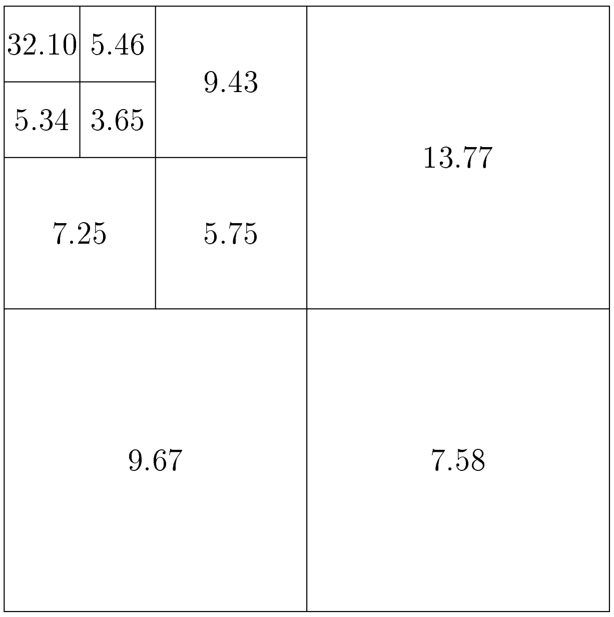


Figure 6.22: Analysis of extracted detail, 67.90% of that in the original image.

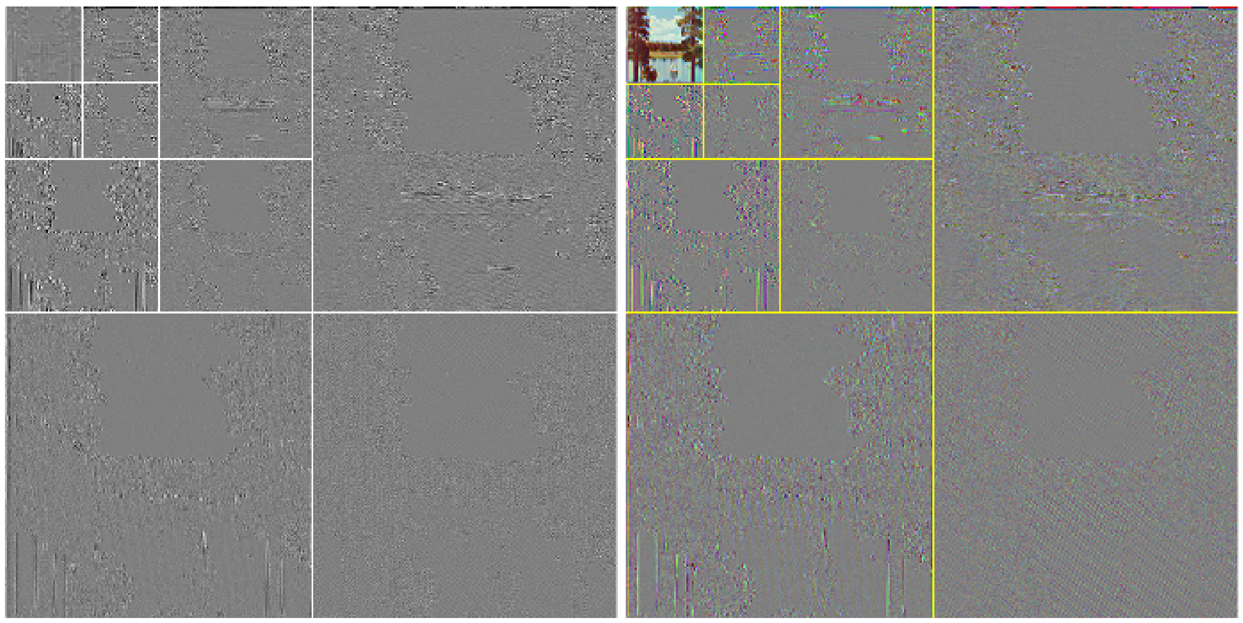


Figure 6.23: QWT of the yacht as a colour vector image, using our length 18 full quaternion filters.

In the red component of the level one horizontal detail, $\sigma = 0.0172, \gamma_1 = -0.1342$ and $\beta_2 = 18.1739$.

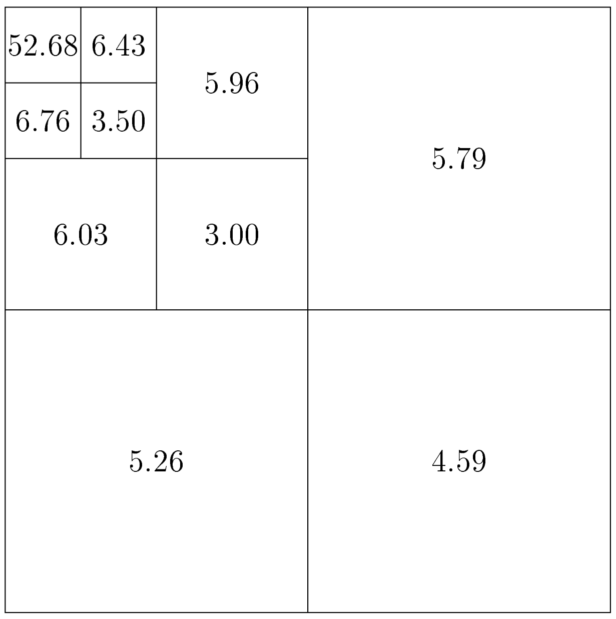


Figure 6.24: Analysis of extracted detail, 47.32% of that in the original image.

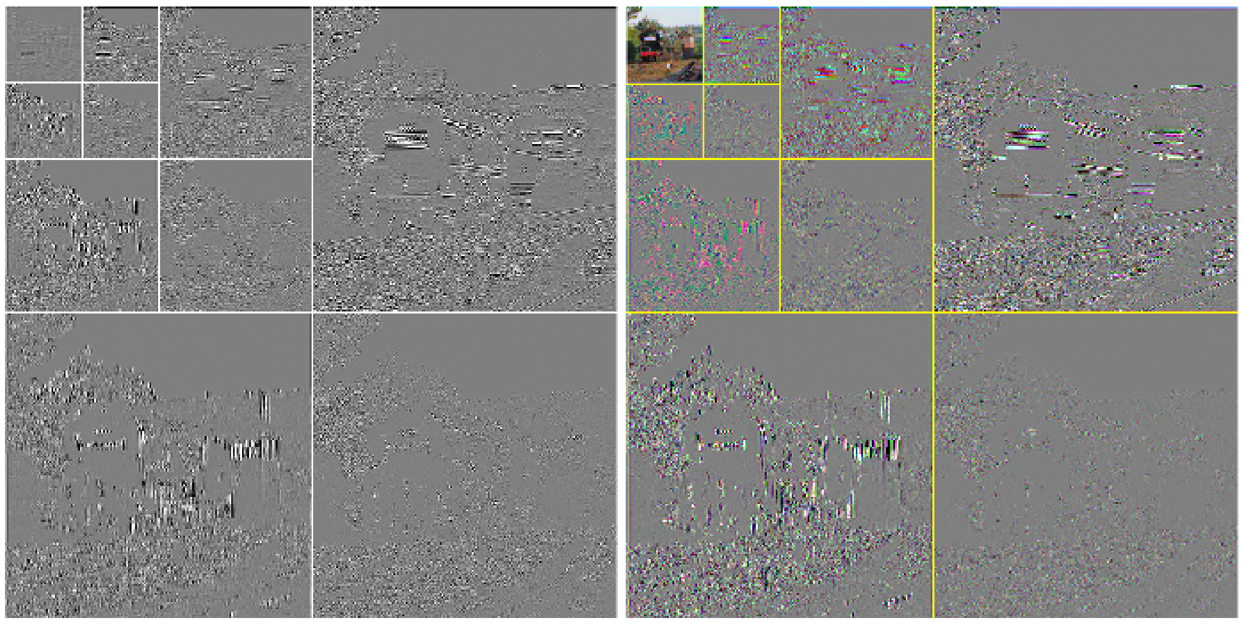


Figure 6.25: QWT of P & O as a colour vector image, using our length 18 full quaternion filters.

In the red component of the level one horizontal detail, $\sigma = 0.0371, \gamma_1 = -0.0603$ and $\beta_2 = 12.0254$.

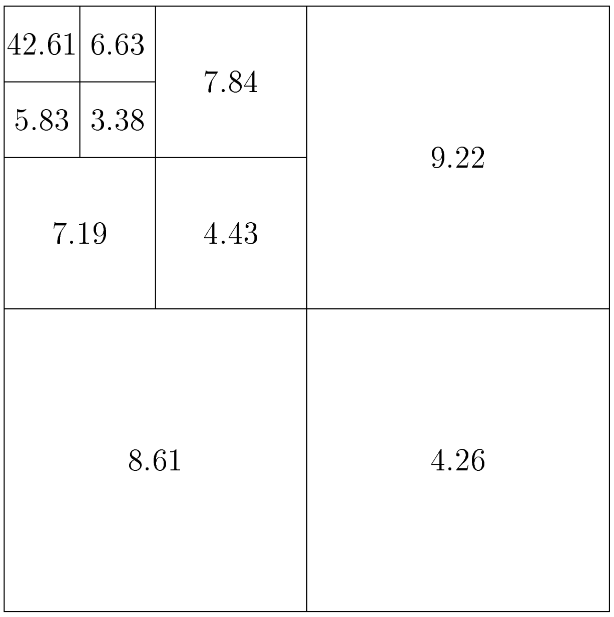


Figure 6.26: Analysis of extracted detail, 57.39% of that in the original image.

In Figures 6.27 and 6.28, we investigate the effect of using full and partial conjugates (see Definitions 2.2.14 and 2.2.17 for those of a single quaternion) of Ginzberg's length 10 and our first length 14 quaternion filters.

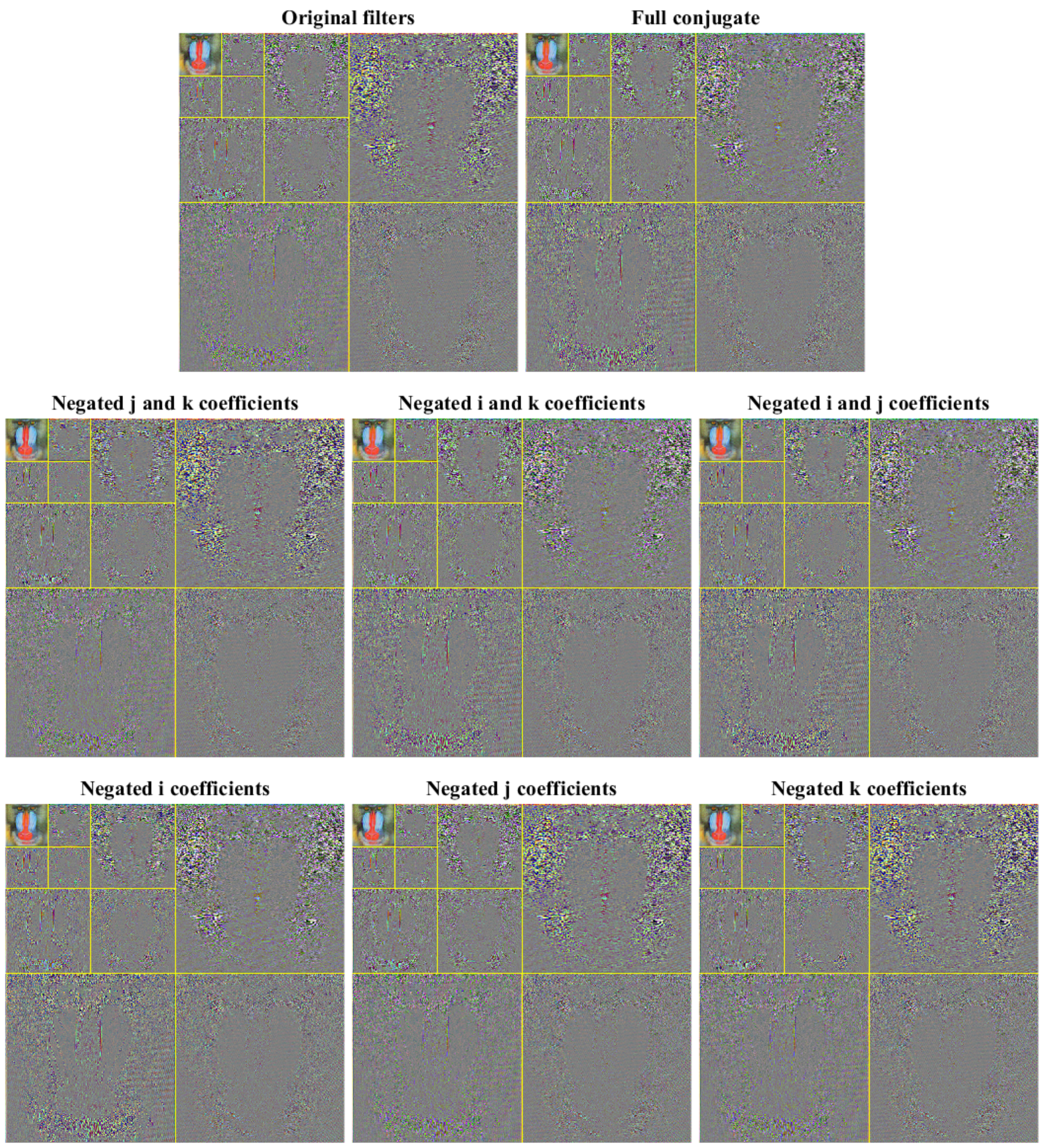


Figure 6.27: QWT of the mandrill as a colour vector image, using partial conjugates of Ginzberg's length 10 full quaternion filters.

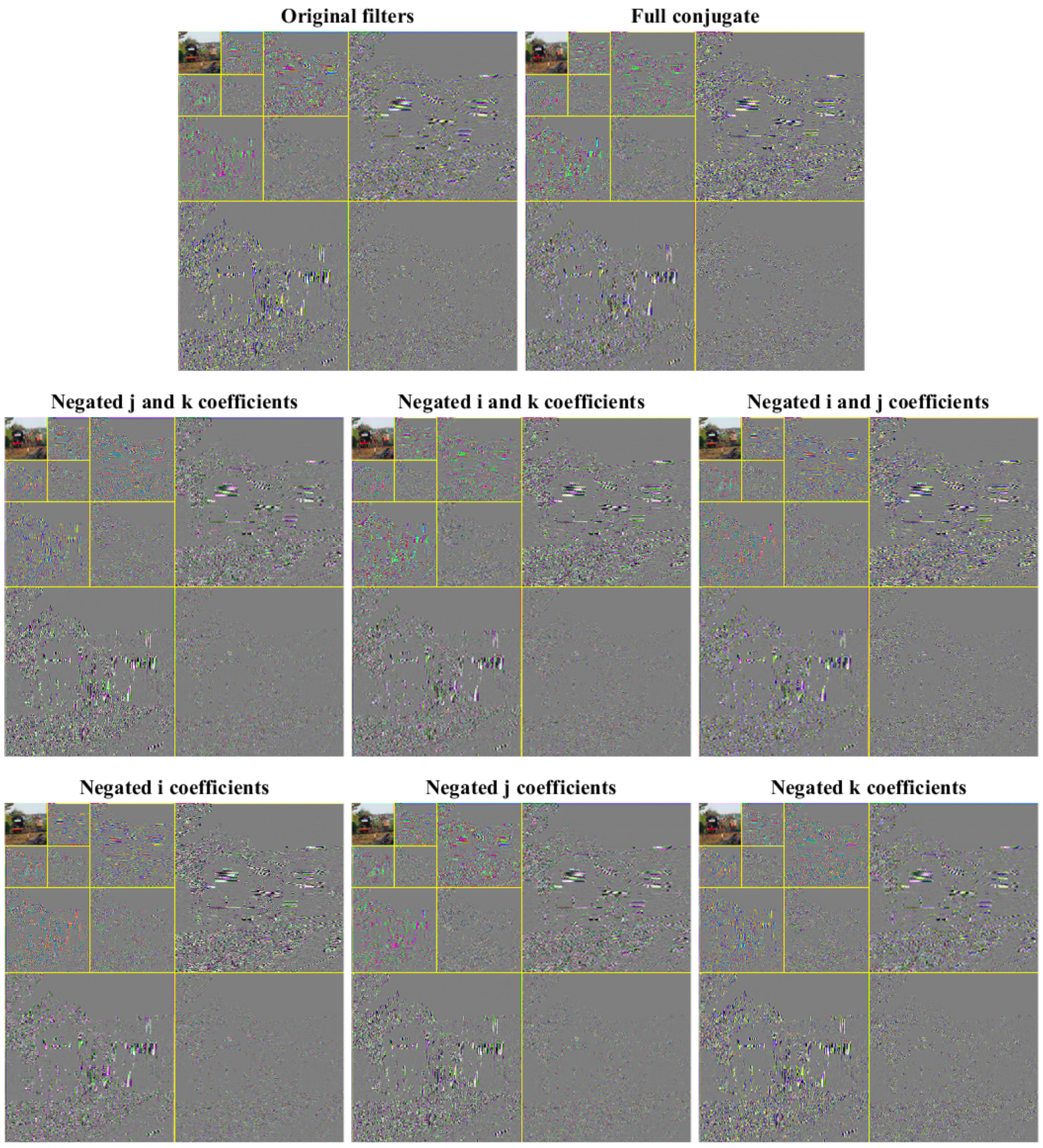


Figure 6.28: QWT of P & O as a colour vector image, using partial conjugates of our first length 14 full quaternion filters.

6.2.1.2 Complex Values in Quaternion Filters

In Figures 6.29 to 6.32, we try using complex scaling and wavelet filter values and their conjugates in the scalar and, separately, the **i**, **j** and **k** parts of quaternion filters.

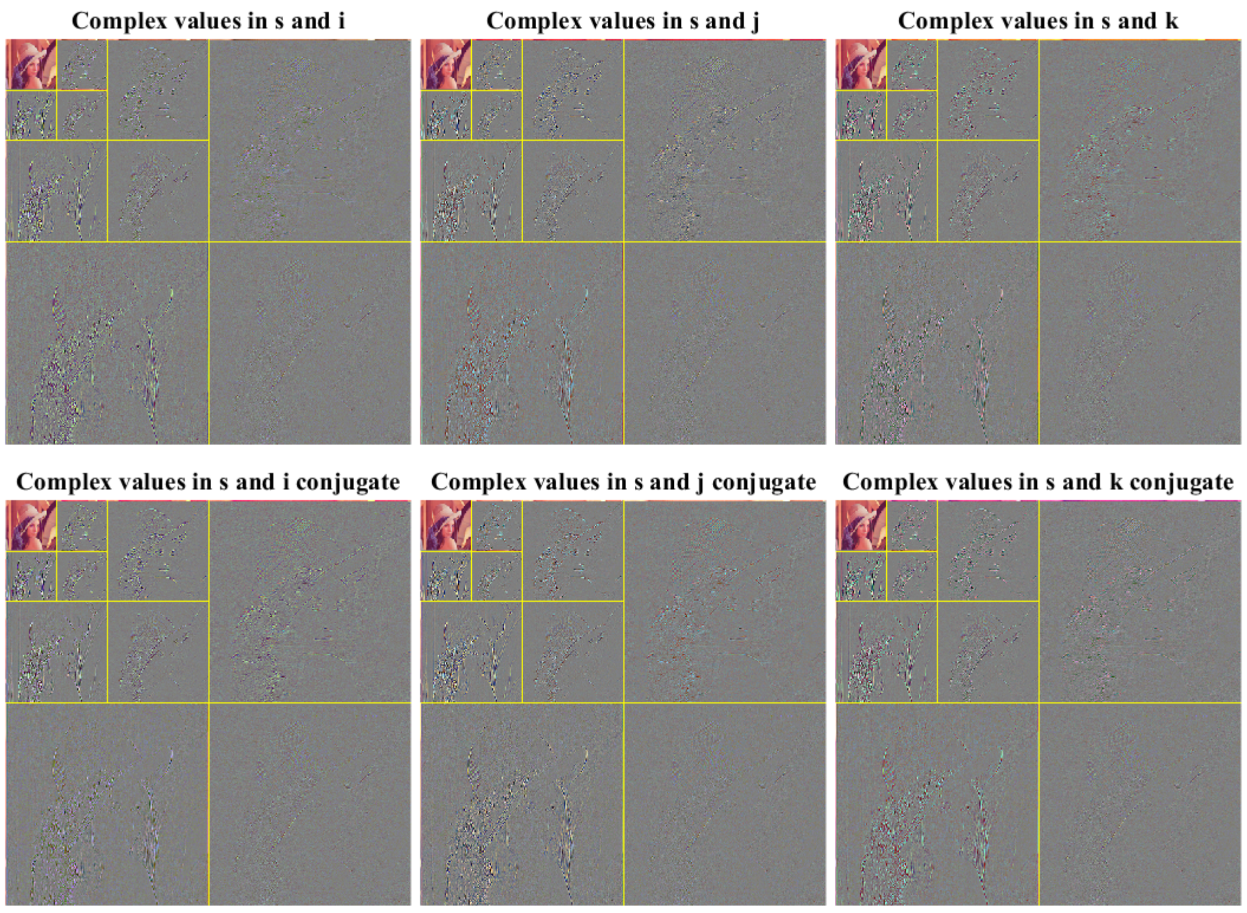

Complex values in s and i
Complex values in s and j
Complex values in s and k
Complex values in s and i conjugate
Complex values in s and j conjugate
Complex values in s and k conjugate

Figure 6.29: QWT of Lena as a colour vector image, using complex length 6 quaternion filters.

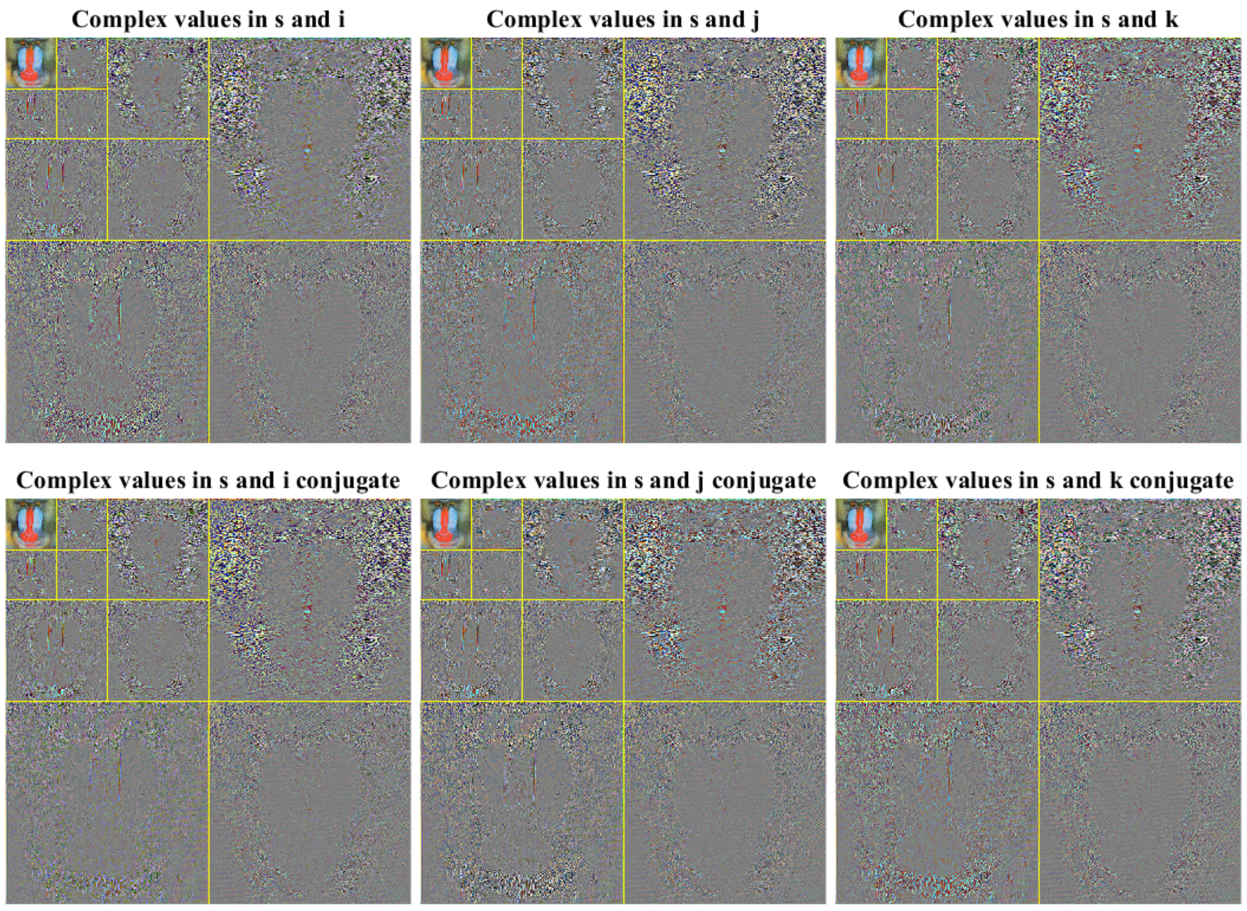

Complex values in s and i
Complex values in s and j
Complex values in s and k
Complex values in s and i conjugate
Complex values in s and j conjugate
Complex values in s and k conjugate

Figure 6.30: QWT of the mandrill as a colour vector image, using complex length 6 quaternion filters.

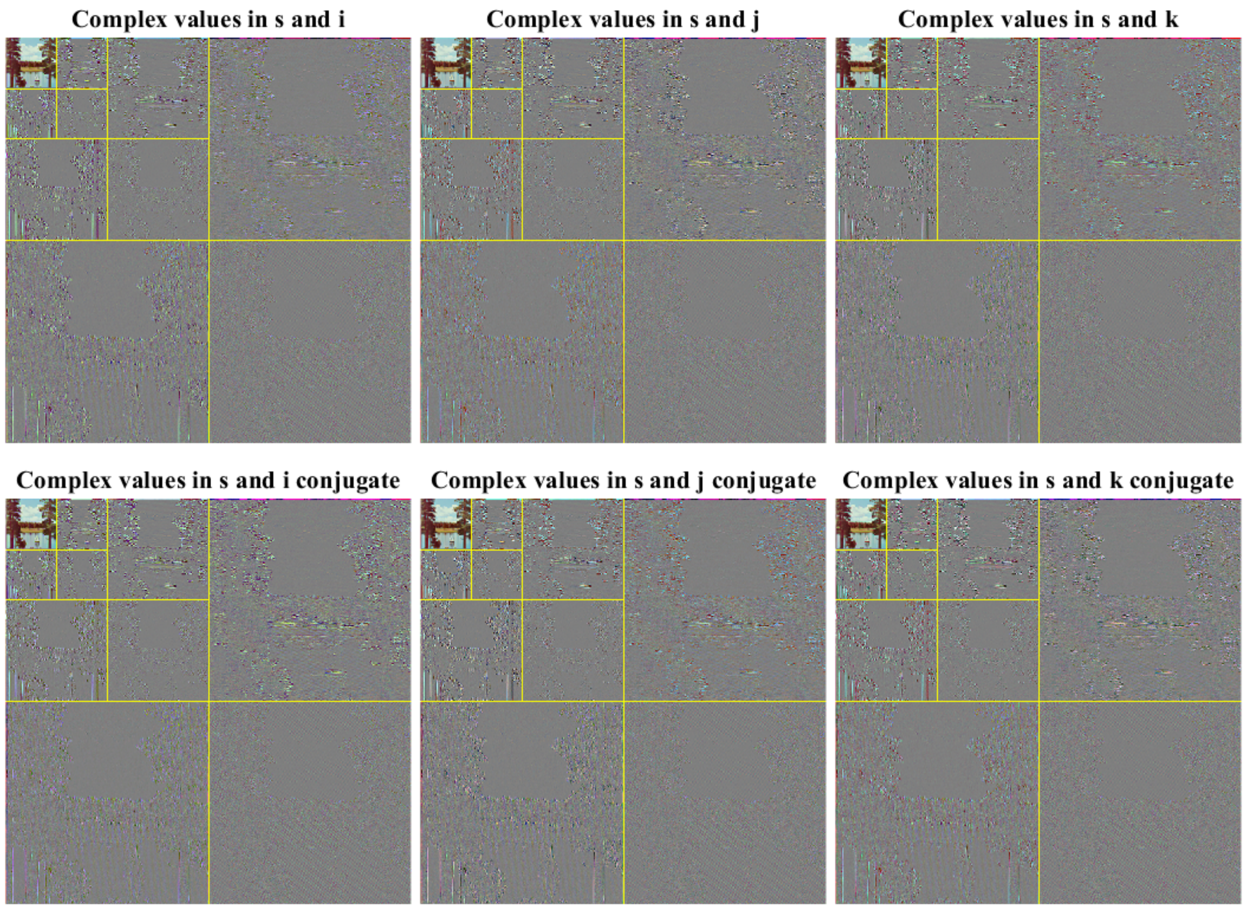

Complex values in s and i
Complex values in s and j
Complex values in s and k
Complex values in s and i conjugate
Complex values in s and j conjugate
Complex values in s and k conjugate

Figure 6.31: QWT of the yacht as a colour vector image, using complex length 6 quaternion filters.

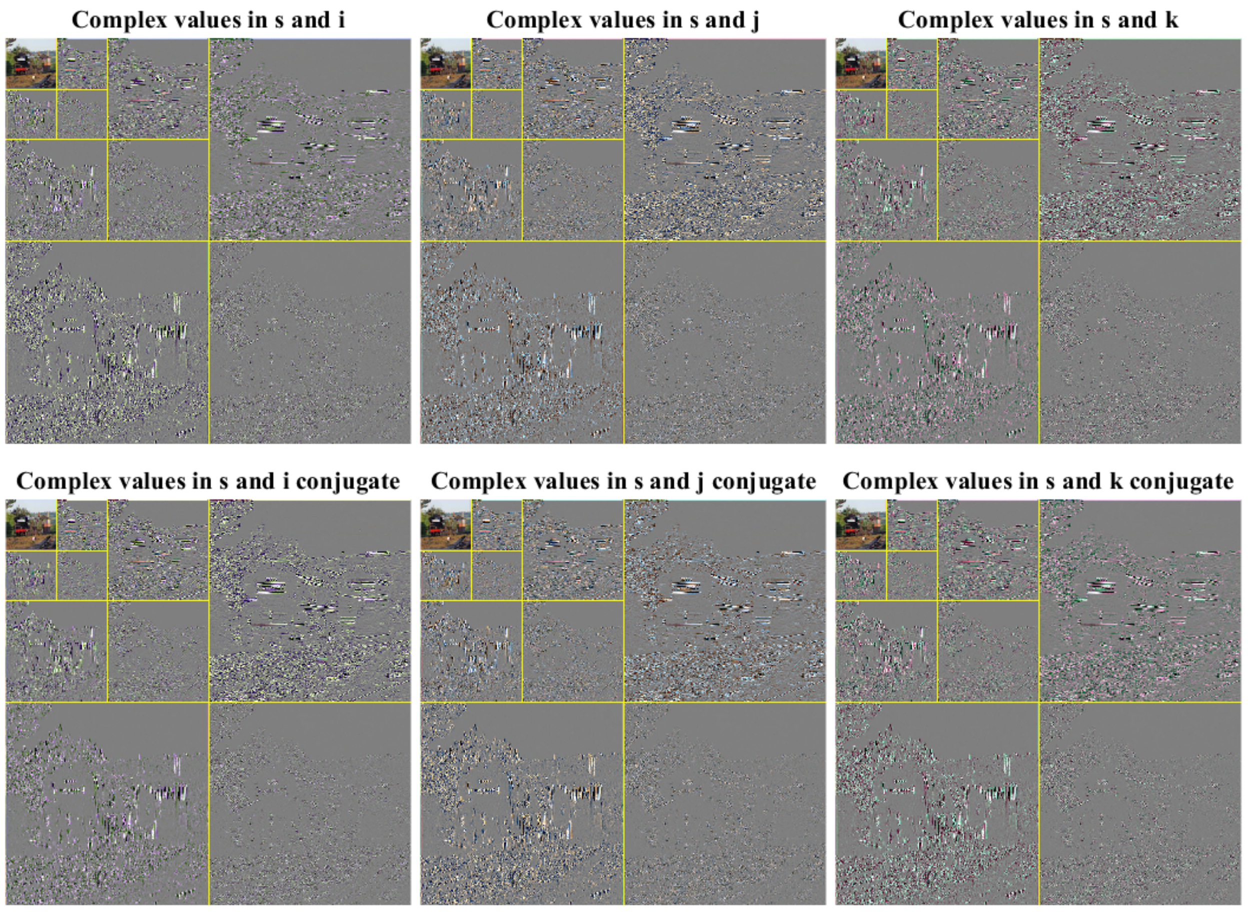

Complex values in s and i
Complex values in s and j
Complex values in s and k
Complex values in s and i conjugate
Complex values in s and j conjugate
Complex values in s and k conjugate

Figure 6.32: QWT of P & O as a colour vector image, using complex length 6 quaternion filters.

6.2.2 Clifford $Cl(1,1)$ Wavelet Transforms

In Subsubsection 6.2.2.1, Figures 6.33 to 6.40, we use our length 10 full Clifford $Cl(1,1)$ scaling and wavelet filters on each test image to find the first ever Clifford wavelet transforms (CWTs). We follow these in Subsubsection 6.2.2.2 with Figures 6.41 to 6.44, wherein we use length 6 complex and our Clifford $Cl(1,0)$ scaling and wavelet filter values in the appropriate parts of Clifford $Cl(1,1)$ scaling and wavelet filters.

6.2.2.1 Full Multiwavelets

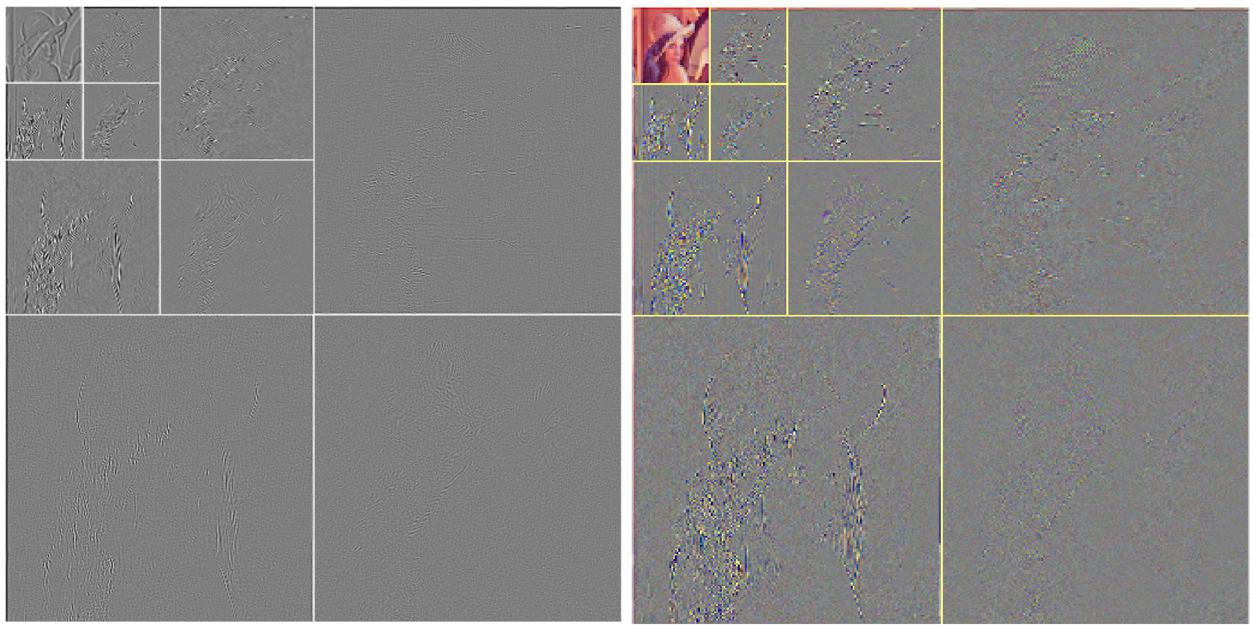


Figure 6.33: CWT of Lena as a colour vector image, using length 10 full $Cl(1,1)$ filters.

In the red component of the level one horizontal detail, $\sigma = 0.0094, \gamma_1 = 0.4832$ and $\beta_2 = 9.7104$.

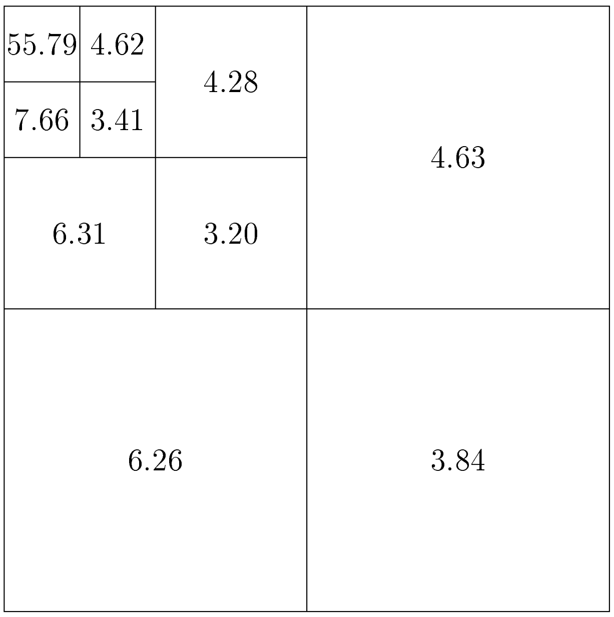


Figure 6.34: Analysis of extracted detail, 44.21% of that in the original image.

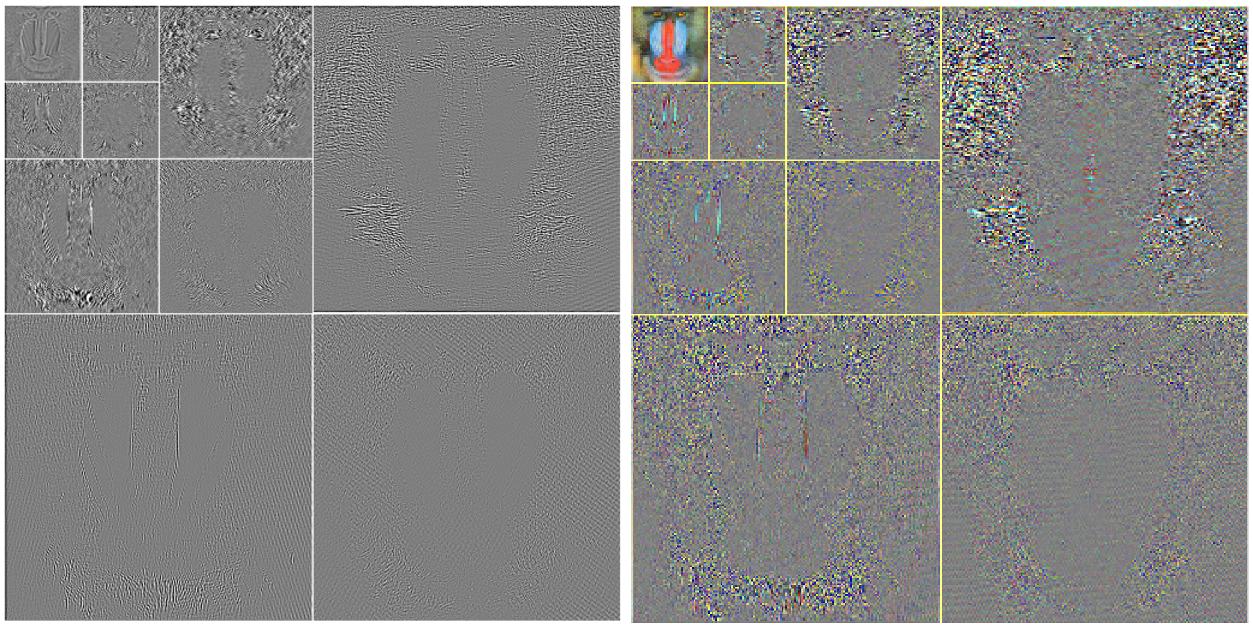


Figure 6.35: CWT of the mandrill as a colour vector image, using our length 10 full $Cl(1,1)$ filters.

In the red component of the level one horizontal detail, $\sigma = 0.0427, \gamma_1 = 0.0565$ and $\beta_2 = 6.4093$.

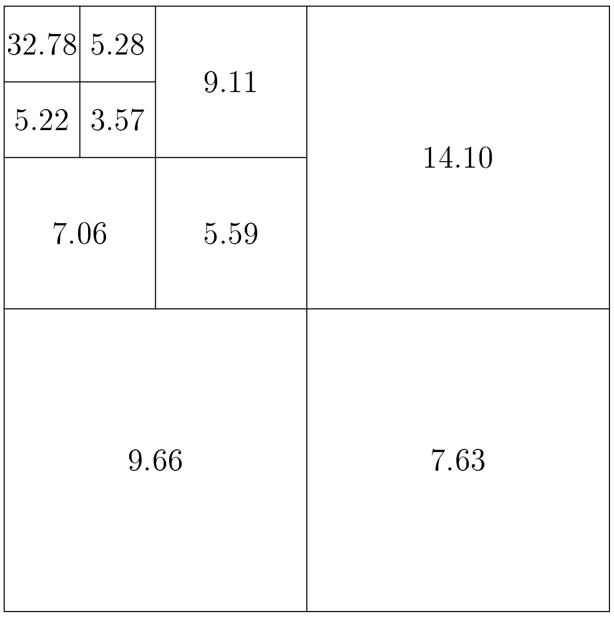


Figure 6.36: Analysis of extracted detail, 67.22% of that in the original image.

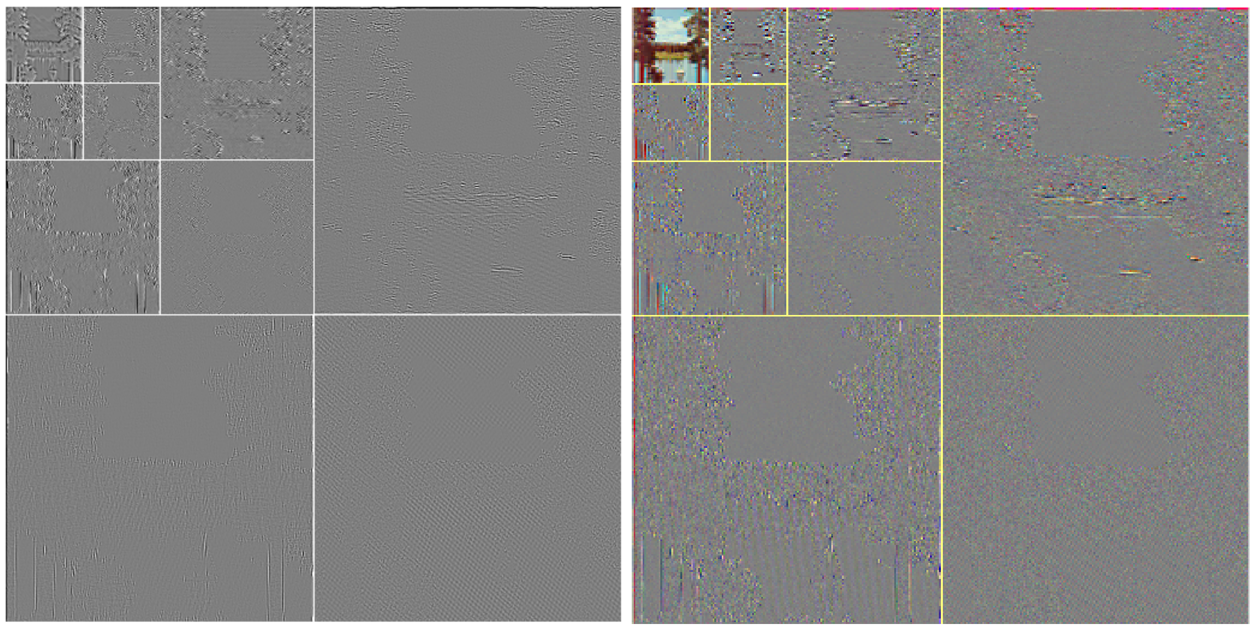


Figure 6.37: CWT of the yacht as a colour vector image, using our length 10 full $Cl(1,1)$ filters.

In the red component of the level one horizontal detail, $\sigma = 0.0240, \gamma_1 = 0.1656$ and $\beta_2 = 20.2576$.

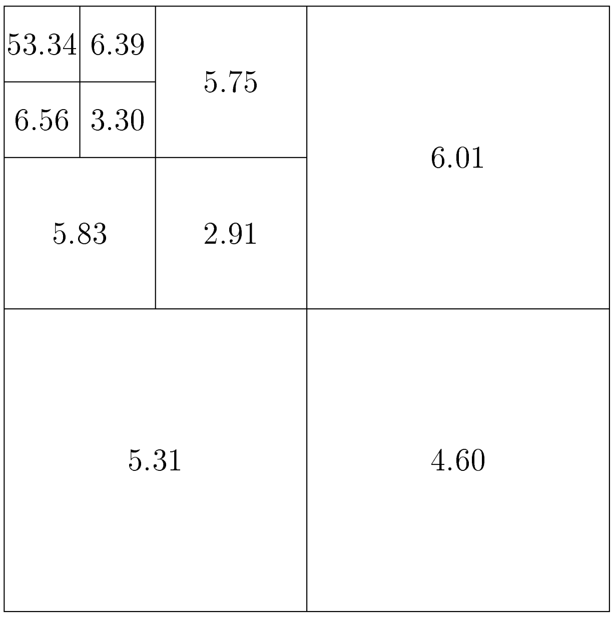


Figure 6.38: Analysis of extracted detail, 46.66% of that in the original image.

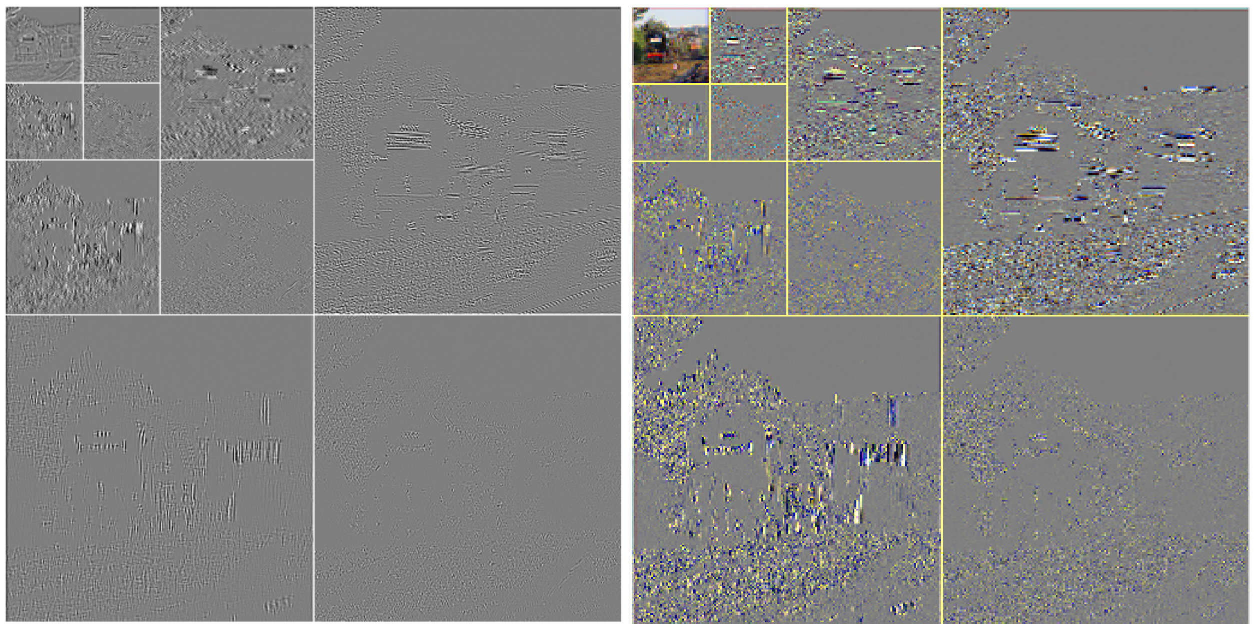


Figure 6.39: CWT of P & O as a colour vector image, using our length 10 full $Cl(1,1)$ filters.

In the red component of the level one horizontal detail, $\sigma = 0.0363, \gamma_1 = 0.0855$ and $\beta_2 = 11.0939$.

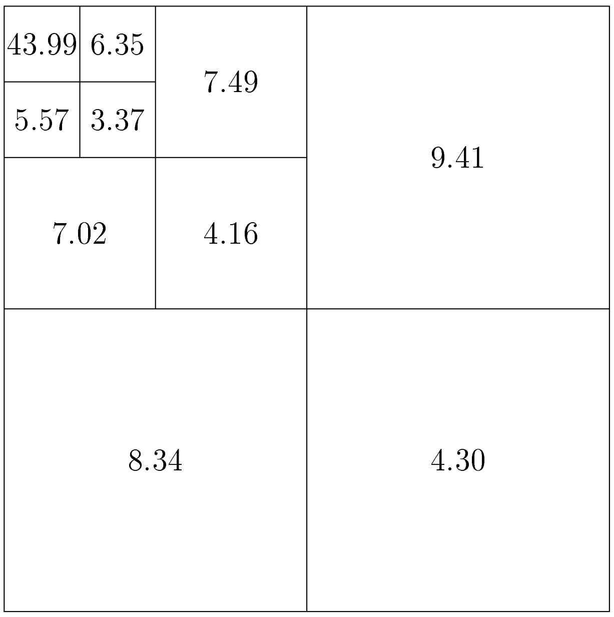


Figure 6.40: Analysis of extracted detail, 56.01% of that in the original image.

6.2.2.2 Complex and $Cl(1,0)$ Values in $Cl(1,1)$ Fliters

For our $Cl(1,1)$ filters in each of Figs. 6.41 to 6.44, from left to right we used the values from length 6 $Cl(1,0)$ filters in the $\mathbf{e}_0$ and $\mathbf{e}_1$ positions, the values from length 6 complex filters in the $\mathbf{e}_0$ and $\mathbf{e}_2$ positions and the values from length 6 $Cl(1,0)$ filters in the $\mathbf{e}_0$ and $\mathbf{e}_{12}$ positions.

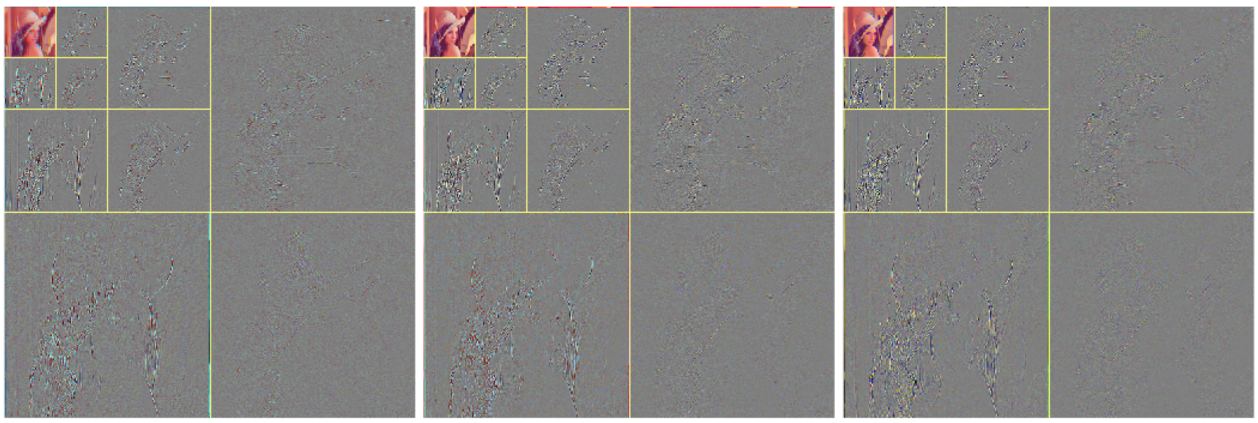


Figure 6.41: CWT of Lena as a colour vector image, using values from length 6 complex and $Cl(1,0)$ filters in $Cl(1,1)$ filters.

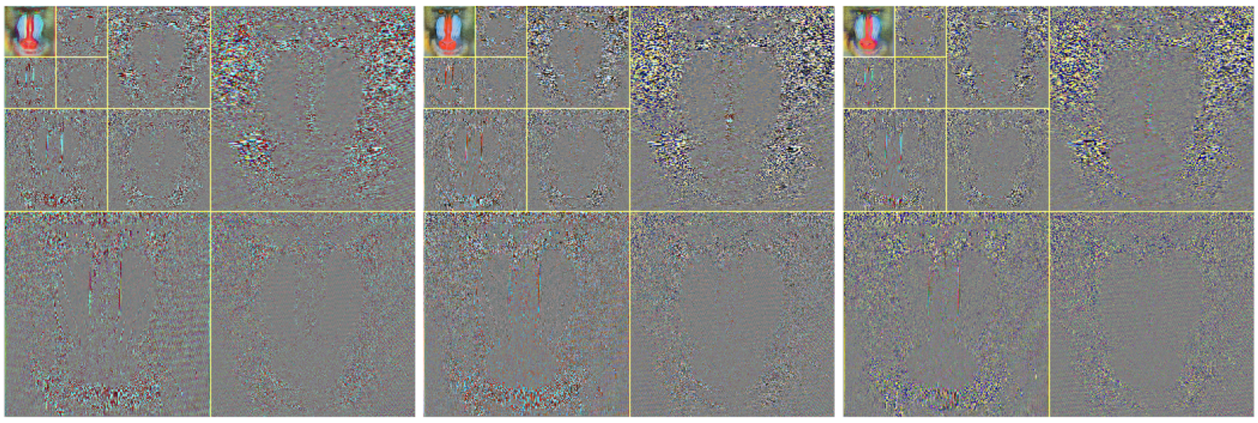


Figure 6.42: CWT of the mandrill as a colour vector image, using values from length 6 complex and $Cl(1,0)$ filters in $Cl(1,1)$ filters.

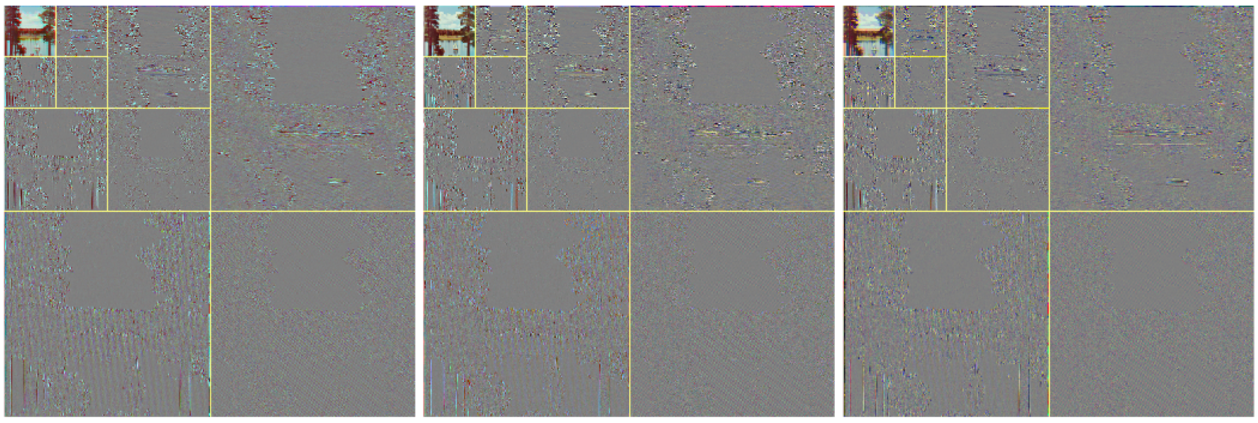


Figure 6.43: CWT of the yacht as a colour vector image, using values from length 6 complex and $Cl(1,0)$ filters in $Cl(1,1)$ filters.

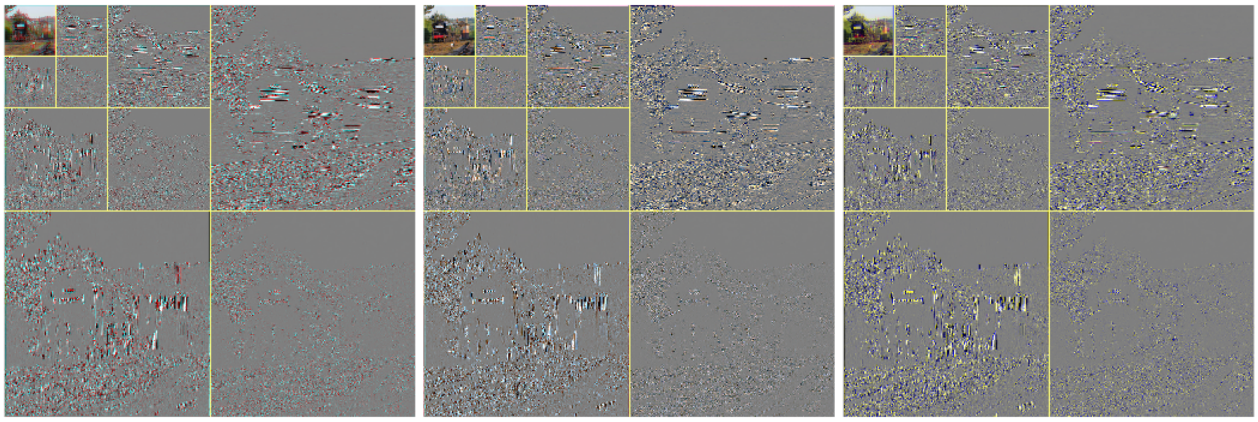


Figure 6.44: CWT of P & O as a colour vector image, using values from length 6 complex and $Cl(1,0)$ filters in $Cl(1,1)$ filters.

6.2.3 Clifford $Cl(2,0)$ Wavelet Transforms

In Subsubsection 6.2.3.1, Figures 6.45 to 6.52, we use our first length 10 full Clifford $Cl(2,0)$ scaling and wavelet filters on each test image. We follow these in Subsubsection 6.2.3.2 with Figures 6.53 to 6.56, wherein we use length 6 complex and our Clifford $Cl(1,0)$ scaling and wavelet filter values in the appropriate places of Clifford $Cl(2,0)$ scaling and wavelet filters.

6.2.3.1 Full Multiwavelets

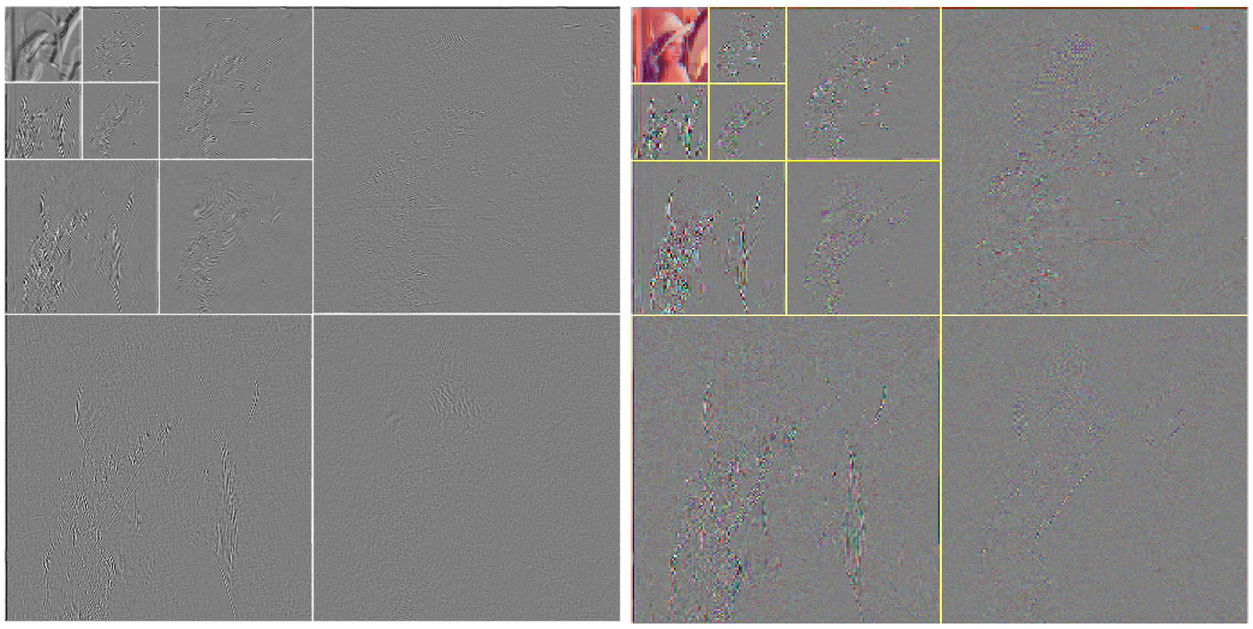


Figure 6.45: CWT of Lena as a colour vector image, using length our first length 10 full $Cl(2,0)$ filters.

In the red component of the level one horizontal detail, $\sigma = 0.0096, \gamma_1 = 0.4029$ and $\beta_2 = 10.9633$.

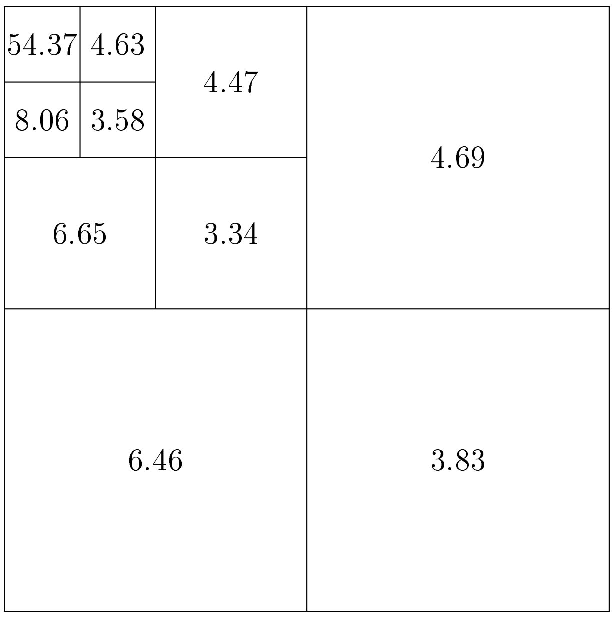


Figure 6.46: Analysis of extracted detail, 45.63% of that in the original image.

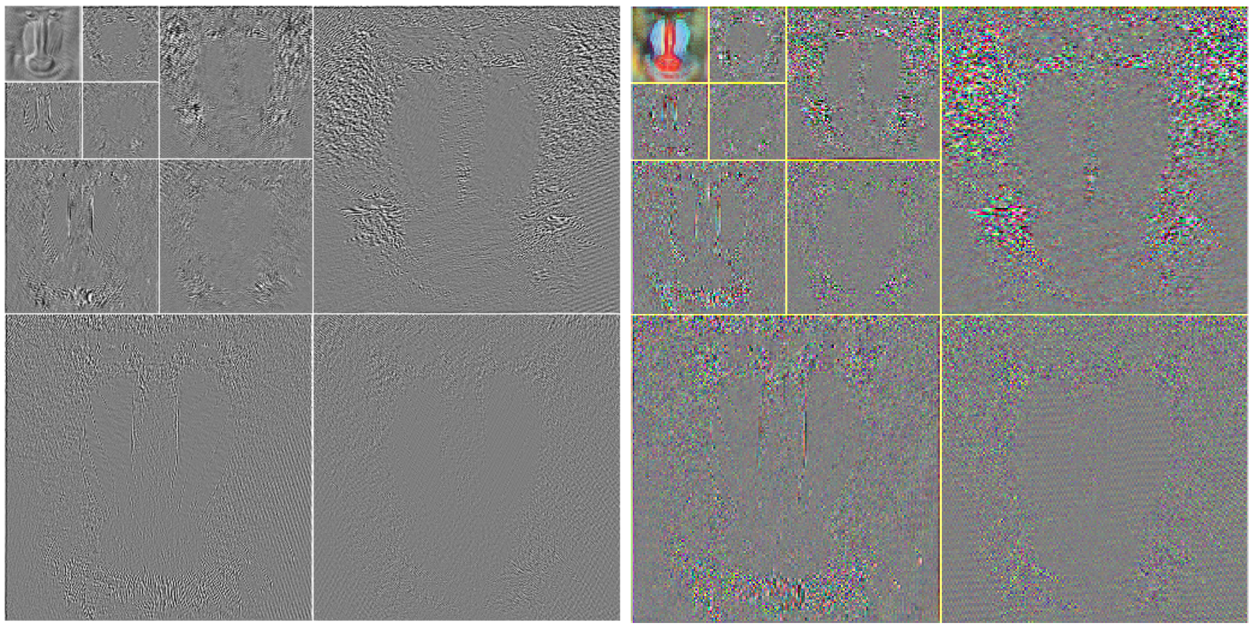


Figure 6.47: CWT of the mandrill as a colour vector image, using our first length 10 full $Cl(2,0)$ filters.

In the red component of the level one horizontal detail, $\sigma = 0.0457, \gamma_1 = 0.2904$ and $\beta_2 = 7.1101$.

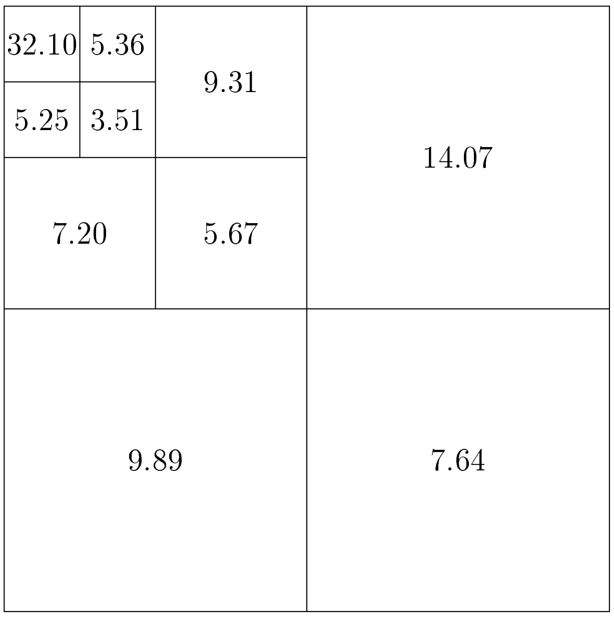


Figure 6.48: Analysis of extracted detail, 67.90% of that in the original image.

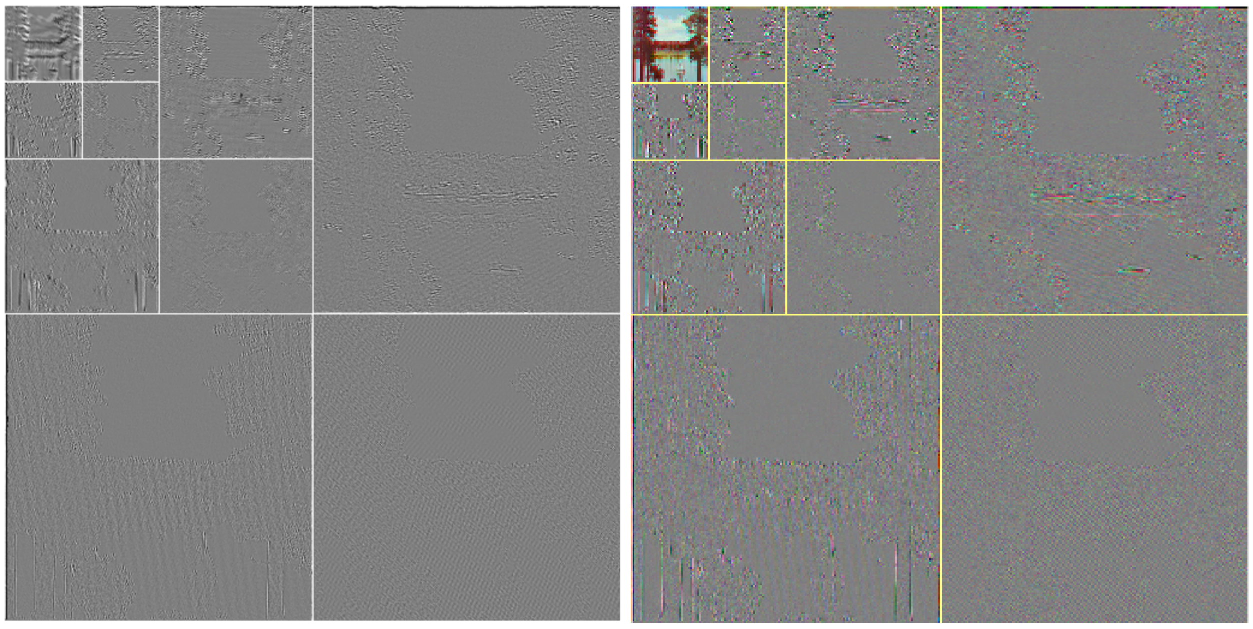


Figure 6.49: CWT of the yacht as a colour vector image, using our first length 10 full $Cl(2,0)$ filters.

In the red component of the level one horizontal detail, $\sigma = 0.0231, \gamma_1 = -1.0433$ and $\beta_2 = 16.5627$.

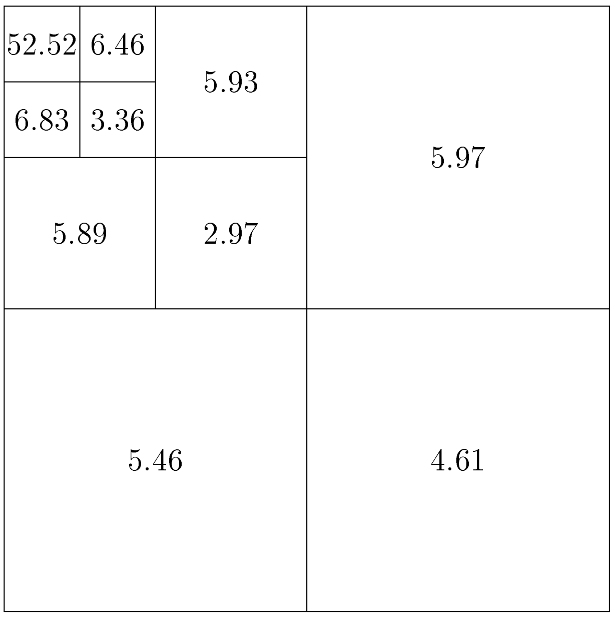


Figure 6.50: Analysis of extracted detail, 47.48% of that in the original image.

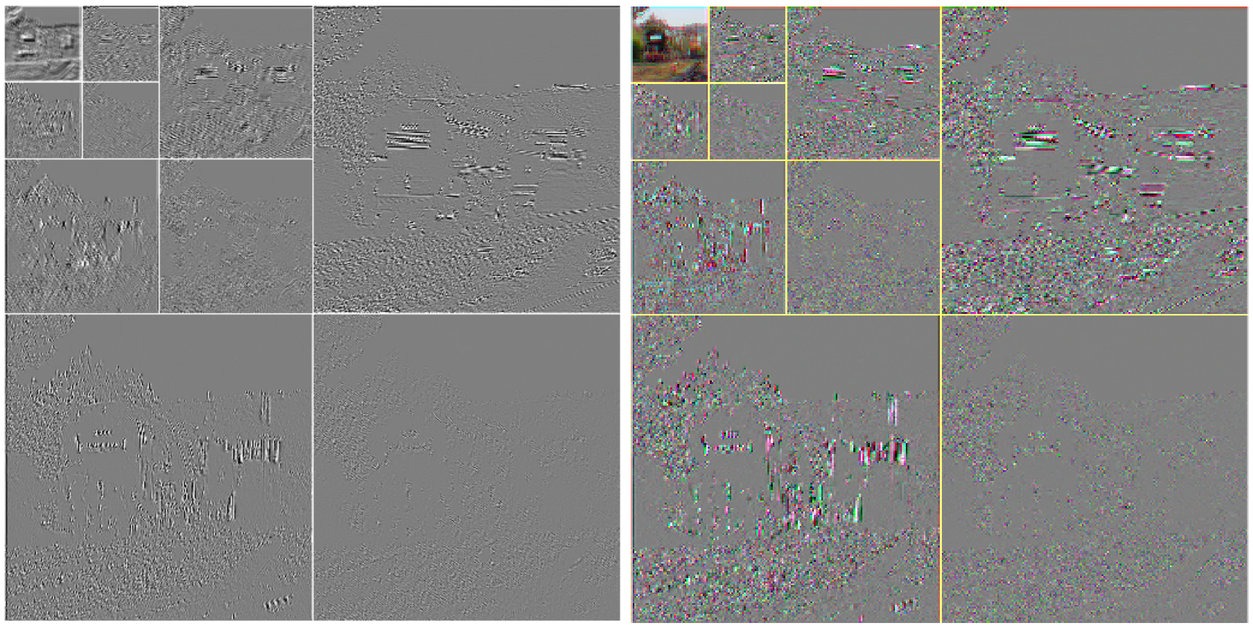


Figure 6.51: CWT of P & O as a colour vector image, using our first length 10 full $Cl(2,0)$ filters.

In the red component of the level one horizontal detail, $\sigma = 0.0411, \gamma_1 = 0.0359$ and $\beta_2 = 12.2651$.

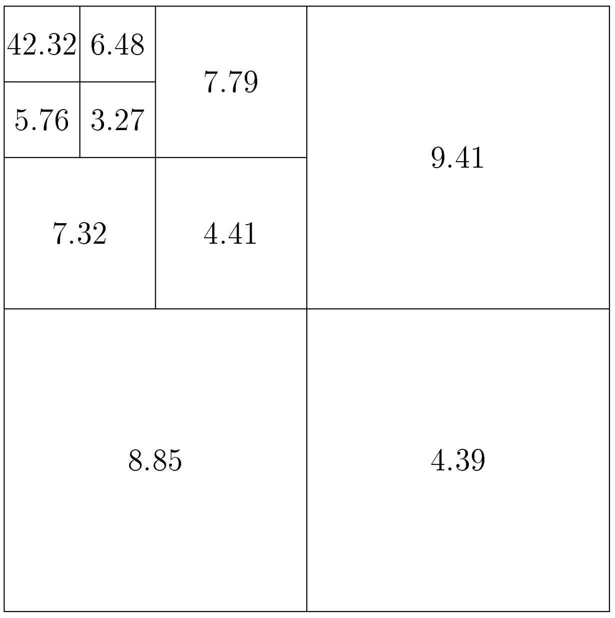


Figure 6.52: Analysis of extracted detail, 57.68% of that in the original image.

6.2.3.2 Complex and $Cl(1,0)$ Values in $Cl(2,0)$ Fliters

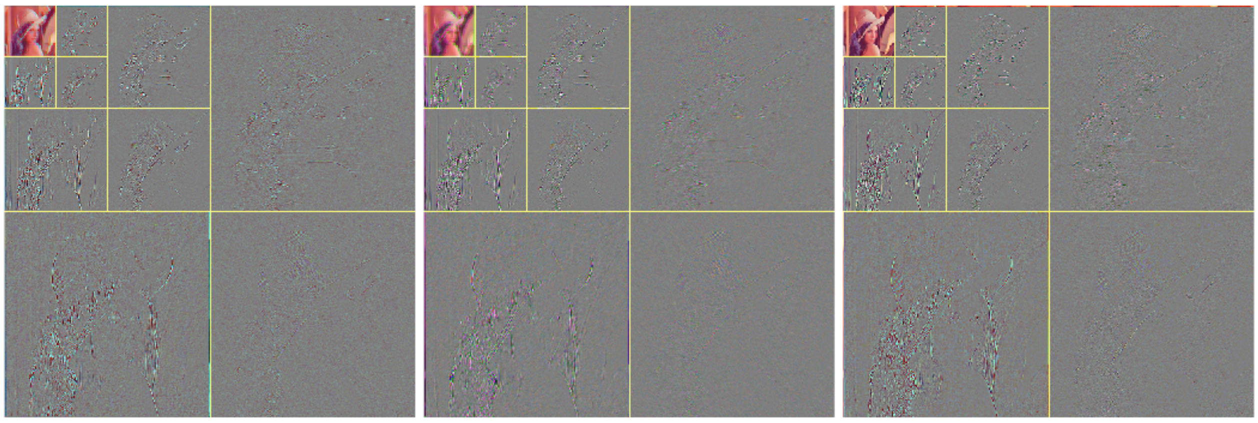


Figure 6.53: CWT of Lena as a colour vector image, using values from length 6 complex and $Cl(1,0)$ filters in a $Cl(1,1)$ filter.

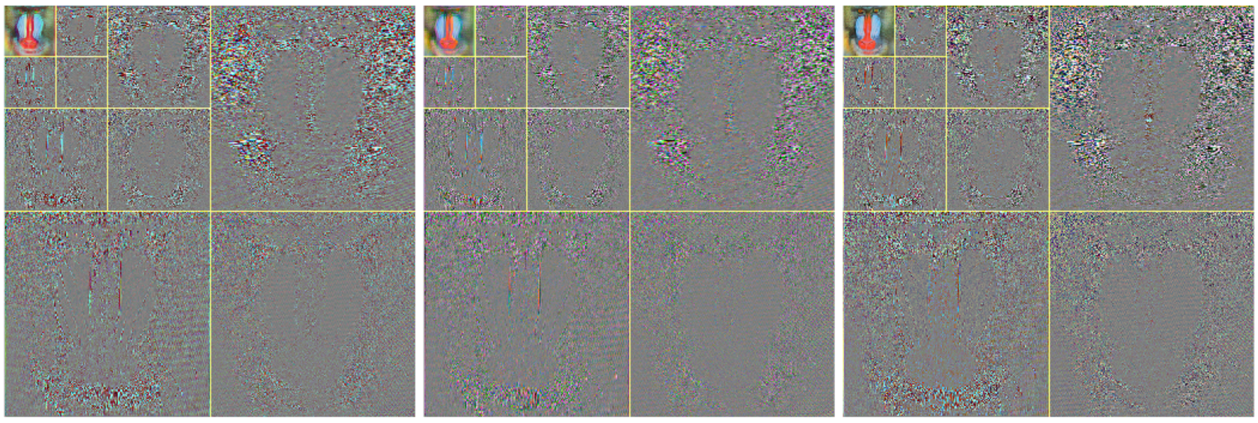


Figure 6.54: CWT of the mandrill as a colour vector image, using values from length 6 complex and $Cl(1,0)$ filters in a $Cl(1,1)$ filter.

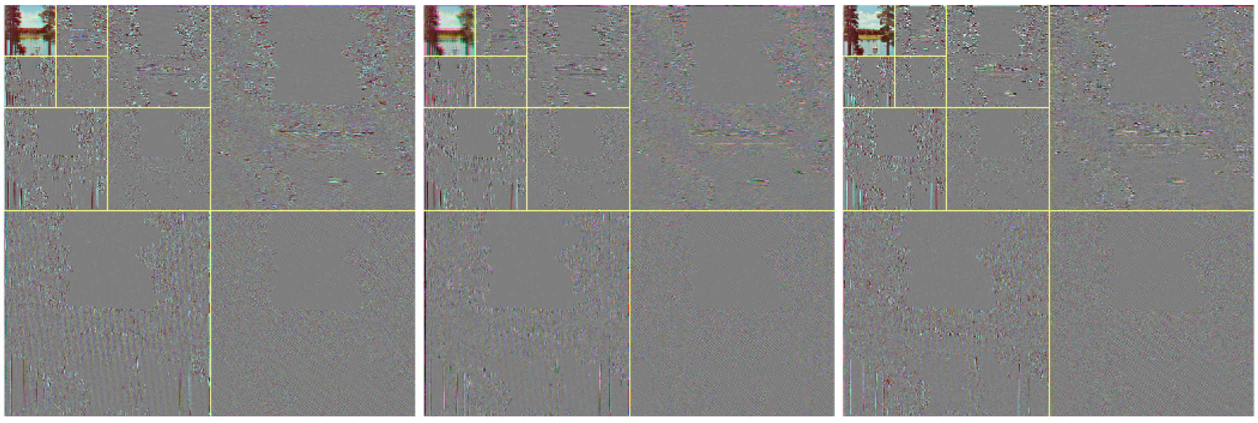


Figure 6.55: CWT of the yacht as a colour vector image, using values from length 6 complex and $Cl(1,0)$ filters in a $Cl(1,1)$ filter.

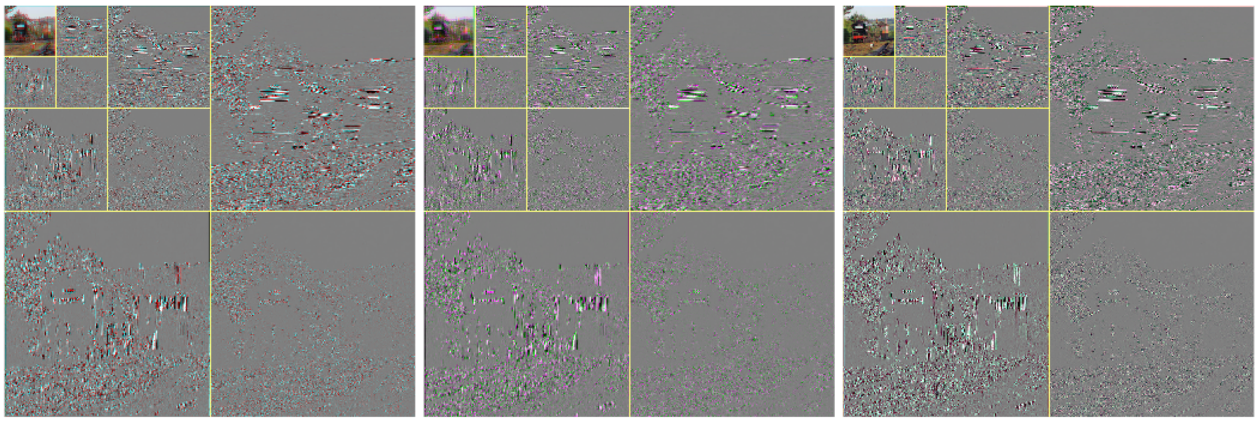


Figure 6.56: CWT of P & O as a colour vector image, using values from length 6 complex and $Cl(1,0)$ filters in a $Cl(1,1)$ filter.

6.2.4 Real Wavelet Transforms for Comparison

In this subsection, we use real Daubechies length 10 scaling and wavelet filters in the scalar parts of quaternion filters, leaving the imaginary parts zero, on each test image. We also do as we did with the results from full quaternion and Clifford filters and in each case give the standard deviation, skewness and kurtosis of the red component of the first level horizontal detail and an analysis of all the detail extracted from the original image.

We obtained the filters from MATLAB® by using

```
[LoD,HiD] = wfilters('db5')
```

`LoD` being the scaling filter, `HiD` the wavelet filter and `db5` the label for length 10 Daubechies filters.

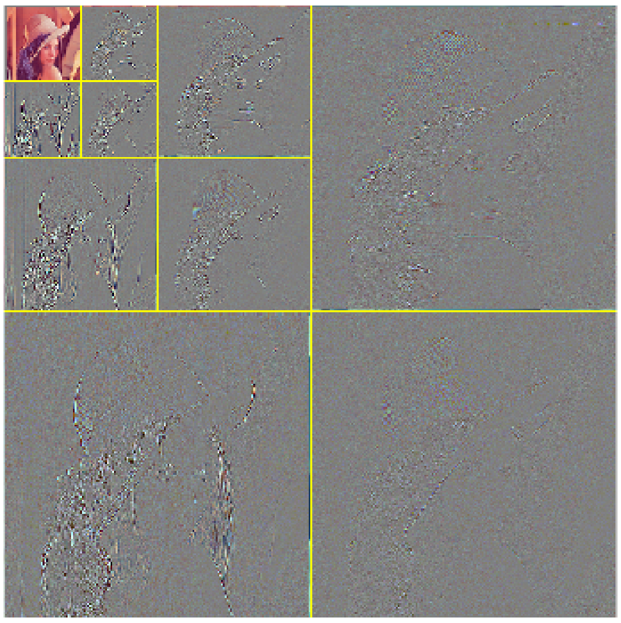


Figure 6.57: Wavelet transform of Lena, using Daubechies' real length 10 filters.

In the red component of the level one horizontal detail, $\sigma = 0.0081, \gamma_1 = -1.5921$ and $\beta_2 = 22.6430$.

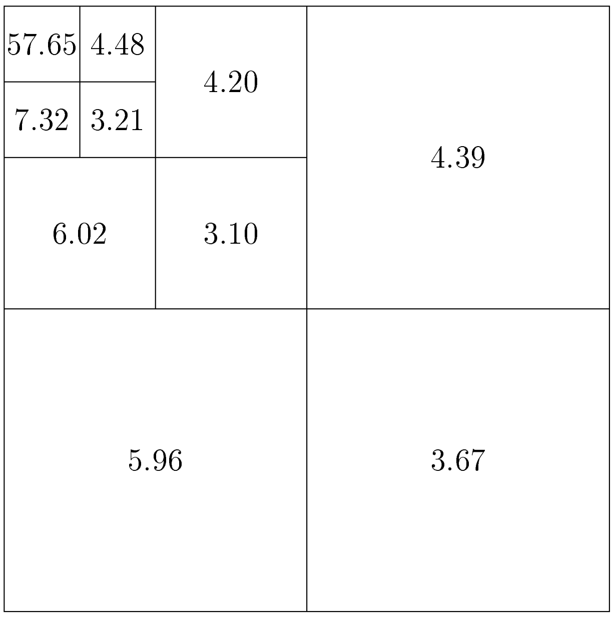


Figure 6.58: Analysis of extracted detail, 42.35% of that in the original image.

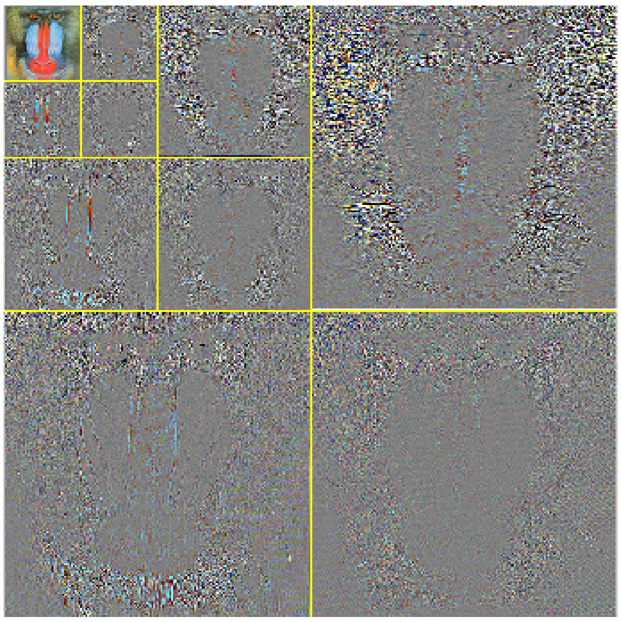


Figure 6.59: Wavelet transform of the mandrill, using Daubechies' real length 10 filters.

In the red component of the level one horizontal detail, $\sigma = 0.0472, \gamma_1 = 0.5014$ and $\beta_2 = 8.7681$.

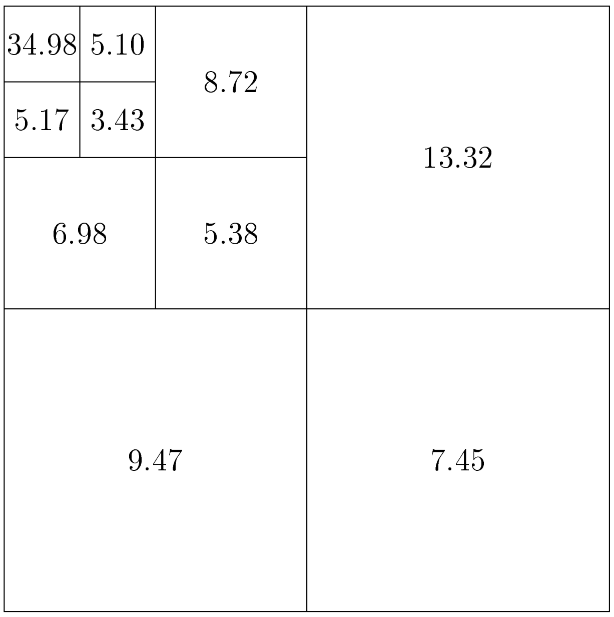


Figure 6.60: Analysis of extracted detail, 65.02% of that in the original image.

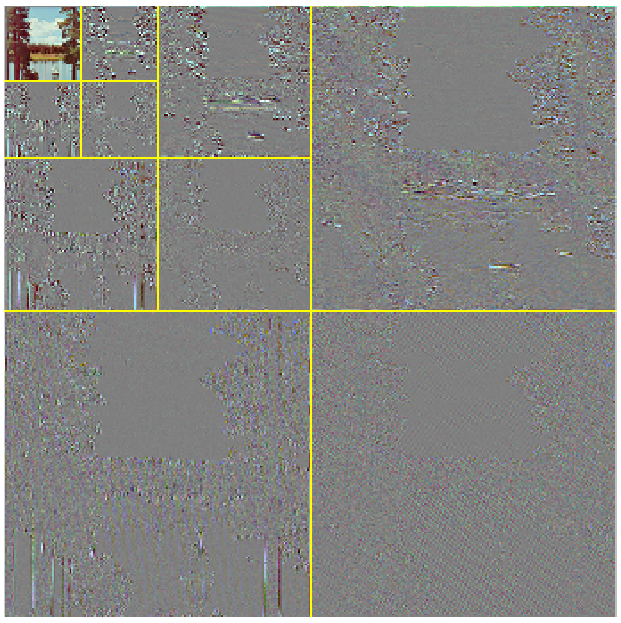


Figure 6.61: Wavelet transform of the yacht, using Daubechies' real length 10 filters.

In the red component of the level one horizontal detail, $\sigma = 0.0158, \gamma_1 = 0.0258$ and $\beta_2 = 7.5389$.

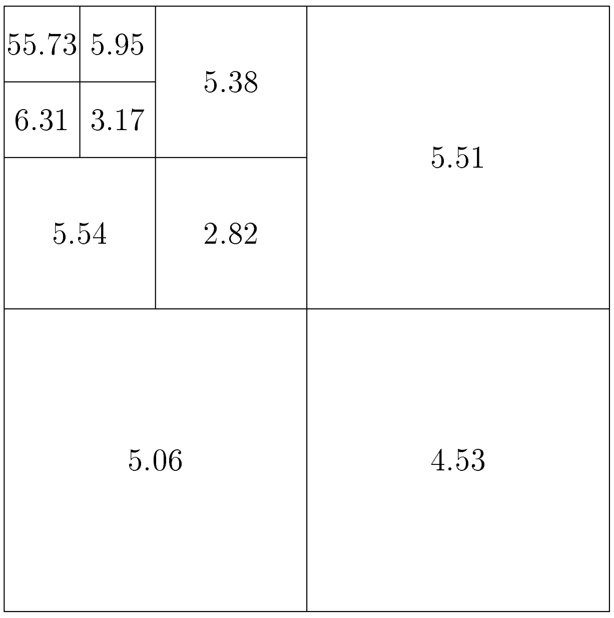


Figure 6.62: Analysis of extracted detail, 44.27% of that in the original image.

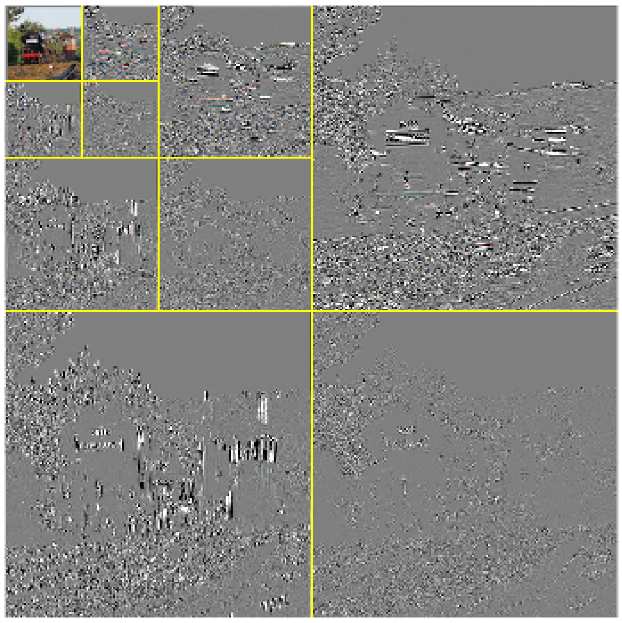


Figure 6.63: Wavelet transform of P & O, using Daubechies' real length 10 filters.

In the red component of the level one horizontal detail, $\sigma = 0.0453, \gamma_1 = 0.1010$ and $\beta_2 = 12.6881$.

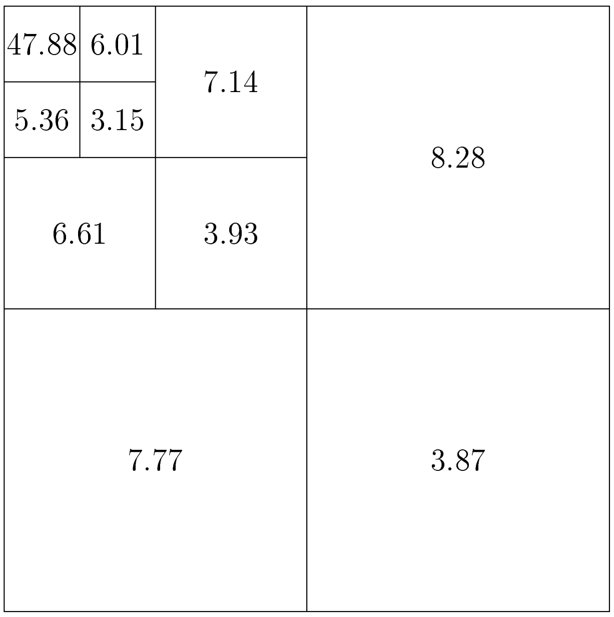


Figure 6.64: Analysis of extracted detail, 52.12% of that in the original image.

6.3 Discussion

All the wavelet transforms of each test image in turn are visually similar to each other. This is to be expected, as the real wavelet transform of an image does not vary greatly as the filters are changed for other real scaling and wavelet filters. Application of a (real, complex or hypercomplex) scaling filter to an image effectively finds a 'weighted average' of local pixels, the weights depending on the particular family and length of filter; the wavelet filter finds 'weighted differences' such that the sum of the results leaves just the original pixel. These sums and differences have their origins in the Haar wavelet, which we mentioned in Subsection 3.1. The differences in colours, between the wavelet transforms of each test image in turn, can be accounted for by the fact that full and partial conjugates of our chosen 'original filters' are also valid scaling and wavelet filters: there are no preferred 'original filters' and ours are just the first ones we found after making some arbitrary decisions, specifically whether to take the positive or negative square root when solving quadratic equations in our Maple™ scripts to find our 'original filters'. Then using the opposite sign gave a partial conjugate as another solution of Ginzberg's equations.

We first look closely at Figure 6.3, the wavelet transforms of Lena using our length 10 quaternion filters. In the scalar image horizontal detail, first level, we can identify Lena's eyes, nose, mouth and the top of her shoulder. There is a little evidence of the feathers and the top of the hat, but it is not very sharp. The second level is clearer. In the vertical detail, first level, we can see the feathers, her hair, the rim of the hat and part of the top of the hat, her eyes and the vertical part of her shoulder. The second level is much the same. In the diagonal detail, first level, we can just discern the rim of her hat and the edge of the feathers, but it is very blurred. At the second level, the rim and the feathers are much more obvious. The third level details appear similar to the second level. Our comments about the scalar detail also apply to the vector detail, except that the colours against the grey background make the details easier to see. The contrasts in colour in the background of the original image only seem to appear at the second level in the diagonal and vertical detail in both the scalar and vector images. A little of the background is apparent in all six third level detail parts.

We now look closely at Figure 6.5, the wavelet transforms of the mandrill, also using our length 10 quaternion filters. The texture of the fur is obvious, being brightest in the scalar image first level horizontal detail. Close inspection of the original image reveals that most fur in the upper half of the image, to the left and right of the eyes, is indeed horizontally aligned. The outline of the red snout is clearest in the scalar image vertical detail parts; in the vector image vertical detail parts, the colours towards the bottom merge somewhat with the surrounding colours, although the vertical edges of the snout are obvious. The vertical whiskers at the bottom of the original image stand out in the vertical detail parts, perhaps moreso in the vector one. The outline of the cheeks can be seen in all of the vector detail parts and the scalar horizontal and diagonal detail parts. They cannot be distinguished in the scalar vertical detail. A rippling effect is evident in all three first level scalar detail parts and which is clearest in the diagonal detail. Rippling can also be seen in the vector vertical detail only. We cannot definitively explain these effects. Could they somehow be the results of aliasing?

Next, we look closely at Figure 6.15, the wavelet transforms of the yacht using our first length 14 filters. The tree trunks are clear in both the scalar and vector vertical detail parts, moreso at the second and third level. The yacht can be identified in all the horizontal and vertical detail images, but it is barely perceptible in the two level one diagonal detail parts. The edge of the sky/treetops is clearly defined in all the detail images, but it is sharper in the horizontal and vertical detail parts. Evidence of the clouds can just be seen in all the scalar detail parts, except perhaps the second level diagonal detail part. They are virtually invisible in all the vector detail parts except the level one diagonal detail. The grass on the opposite side of the lake can be seen in the vector first and second level horizontal detail part and the line separating the grass from the trees is visible in the vertical vector detail part. This line is also visible in the scalar horizontal and vertical detail parts at levels one and two and in the second level diagonal detail part. A difference between the grass and the trees beyond can just be perceived in the two level one diagonal detail parts, but the grass and the trees are so similar that the horizontal dividing line is not sharp. There is some evidence of rippling again in the two level one diagonal detail parts and perhaps a little in the two level one vertical detail parts.

Finally, we look closely at Figure 6.25, the wavelet transforms of P & O using our length 18 filters. All the scalar and vector horizontal and vertical detail parts contain a lot of information, except perhaps that in the third level parts less detail is actually identifiable. The scalar image appears to be more or less simply a greyscale version of the vector image. The smokebox numberplate and the headboard are clear in all six horizontal detail parts. The crossbar at the top of the telegraph pole is really just clear at level one, but the signal box window, running plate and bottom of the tender can be seen at levels one and two. The bufferbeam is apparent in the scalar horizontal detail third level and to a lesser extent at levels two and one, but only the top edge is clear in the vector detail at level one. The windows of the signal box, top part of the telegraph pole, smokebox numberplate and headboard are clear in both the scalar and vector vertical detail levels one and two. The lamp at the top of the smokebox can also just be seen. The scalar and vector diagonal detail parts do not really highlight anything in particular. The fact there is some diagonal detail is obvious, but that is all we can say about it. In all the detail images, the sky is plain grey and is clearly delimitated.

We mentioned above that the colours vary as we take full and partial conjugates. In order to investigate this further, we worked our way through the full and all partial conjugates of Ginzberg's length 10 filters with the mandrill image in Figure 6.27 and similarly with our first length 14 filters and the P & O image in Figure 6.28. The most noticeable thing about the mandrill wavelet transforms is that there are apparently two versions of the first level horizontal detail, one predominantly purple and the other, yellow. When we talk of differences between the detail parts, we shall only be referring to the colours. The same appears to be true for the first level vertical detail, although these ones do not appear as bright. Similar differences are apparent with the second level detail. The third level detail is a bit small, but varies in a similar fashion. There appears to be slightly more variability between the colours of the detail images for the P & O wavelet transforms. There seems to be three main sorts of first level horizontal detail and either two or three sorts of first level vertical detail. At the second level, there are three clear sorts of horizontal detail and three or maybe four sorts of vertical detail. There are also differences between the third level detail images. The diagonal detail in all the cases seems more or less the same.

In Figures 6.29 to 6.32, we looked at length 6 complex filter values in quaternion filters. In these figures we can see a greater variation of colours between the horizontal detail parts of each image and likewise with the vertical and diagonal detail parts, as compared with Figures 6.27 and 6.28, but less colour variation within each detail part. In Figures 6.41 to 6.44 and 6.53 to 6.56, where we used length 6 complex and $Cl(1,0)$ filter values in $Cl(1,1)$ and $Cl(2,0)$ scaling and wavelet filters, we can also see markedly less variation in colours within each detail part.

Figures 6.57 to 6.63 (odd numbers), the wavelet transforms using real Daubechies length 10 filters, show even less colour variation within each detail part than those using complex or $Cl(1,0)$ values. In fact the detail parts of Figure 6.63 are almost greyscale apart from a few hints of colour in the second and third level horizontal detail.

We have attempted to show that our wavelet transforms do actually differ between different filters, for each test image, by finding the standard deviation (σ), skewness (γ_1) and kurtosis (β_2) of the red component in the first level horizontal detail where we have used full quaternion, Clifford and real filters; no two filters result in exactly the same three figures. We also ran our code in Appendix F using the filters whose wavelet transforms we have not presented, together with lengths 14 and 18 real Daubechies filters, and noted down these statistics. We give the ranges of the values we obtained in Table 6.2.

Table 6.2: Ranges of σ, γ_1 and β_2 in the red component of the first level horizontal detail, over all wavelet transforms from full filters.

	Lena	Mandril	Yacht	P & O
σ	$[0.0080, 0.0128]$	$[0.0351, 0.0472]$	$[0.0158, 0.0240]$	$[0.0285, 0.0453]$
γ_1	$[-1.5921, 8.9353]$	$[-0.0302, 0.5014]$	$[-3.4253, 3.2203]$	$[-0.0398, 0.3485]$
β_2	$[9.7104, 124.4129]$	$[6.3118, 8.7681]$	$[7.5389, 56.1722]$	$[10.3607, 15.8714]$

We could do the same for the blue and green in this region and all three primary colours in the other detail regions, but we believe that the fact that the values of these statistics for the red component differ in this region, between different filters, is sufficient to infer that our wavelet transforms do indeed differ between different filters.

What wavelet filter is optimal for a particular application? Ngui *et al.* [114] discuss this question and conclude that more work is needed to give a definitive answer. In our case we chose to look at the amount of detail extracted by each of our wavelets. If wavelet A extracts more detail from the starting image than wavelet B, it seems reasonable to claim that wavelet A is better than wavelet B *for this particular application, in this particular case*; wavelet B may be better than wavelet A for other signal processing tasks. When we ran our code in Appendix F a number of times (with different filters and images) in order to construct Table 6.2, we also noted the total detail extracted in each case; we give the ranges of the percentages in Table 6.3. We can see that the amount of detail extracted is strongly dependent on the actual test image. For each one, the ranges of detail extraction for the quaternion and Clifford filters overlap, but the quaternion filters are generally slightly better: but both are clearly better than Daubechies' real filters, assuming that our method of measuring the detail is valid.

Table 6.3: Ranges of detail extracted, expressed as percentages of the total detail present in each test image.

	Lena	Mandrill	Yacht	P & O
Quaternion filters	$[45.44, 46.29]$	$[67.61, 68.13]$	$[46.88, 48.01]$	$[56.51, 57.74]$
Clifford filters	$[44.21, 45.66]$	$[67.22, 67.90]$	$[46.66, 47.48]$	$[56.01, 57.68]$
Real filter	$[41.45, 42.35]$	$[63.43, 65.02]$	$[43.58, 44.27]$	$[52.12, 52.33]$

6.4 Chapter Summary

We began by considering how to apply vectors of filter coefficients to a matrix of pixels and introduced the pattern of banded matrix that we used. We then explained how we found the wavelet transform of each image, how the approximation and three detail images could be extracted from this transform and how we could display the results. We then went on to find wavelet transforms using some of our full filters on each of four test images in turn and did the same with quaternion, $Cl(1,1)$ and $Cl(2,0)$ filters constructed using the values from length 6 complex and $Cl(1,0)$ filters. We investigated differences between transforms found using full and partial conjugates and compared our hypercomplex wavelet transforms to real wavelet transforms of the same four test images. We showed that our hypercomplex filters appeared to extract more detail from the test images than real filters.

Chapter 7

Conclusions

7.1 Summary of Thesis

We began by introducing quaternions and Clifford algebras and then real wavelets and wavelet transforms. We went on to investigate how other authors have combined these two topics and looked at the different definitions of quaternion wavelets and quaternion wavelet transforms.

In the most-cited article, by Chan *et al.* [24], the authors used what was effectively a two dimensional short-time Fourier transform based on the quaternionic Gabor transform of Bülow [17]. This 'dual tree quaternion wavelet transform', as the authors called it, can be used to analyse a monochrome image only and quaternions enter into the frame simply as a means to separate an approximation to the original image (scalar or s component) from horizontal ($\mathbf{i}$ component), vertical ($\mathbf{j}$ component) and diagonal ($\mathbf{k}$ component) detail images. Thus we can see where the name 'quaternion wavelet transform' came from, but we would argue that the term is a misnomer because the quaternion s, $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$ here are merely labels: no properties of quaternions are used at all. The articles that have made use of this 'quaternion wavelet transform', as other authors have continued to call it, reported a number of different applications, with some appearing in several different articles as authors tweaked their and others' work.

A couple of authors attempted to generalise the basic equation for a continuous one dimensional wavelet transform to a continuous quaternion wavelet transform, but these were just examples of interesting mathematics and have no practical use.

Augereau and Carré [4] defined a 'hypercomplex polynomial-wavelet transform'. Using polynomials with quaternion coefficients to derive orthogonal basis functions is an unusual way to proceed and the resulting functions do not quite satisfy the conditions to be called 'wavelets', but the authors only claim that they are 'wavelet-like' and the fact they are quaternion-valued led us to include this article.

By solving certain matrix equations symbolically, using real matrix representations of quaternions, Ginzberg [61] was able to find discrete Daubechies quaternion scaling filters. He then used a numerical method to find associated wavelet filters. He did not actually do anything with his filters other than list the coefficients.

Ginzberg's filters were 10 coefficients in length. His symbolic method appeared to be unnecessarily complicated in our opinion. We took a simpler approach to solving his equations and as a result, discovered that the full and partial conjugates of his solution were also solutions. We extended our simplified method to other lengths and found further quaternion scaling filters. By using appropriate real matrix representations, we also found some Clifford $Cl(1,0)$, $Cl(1,1)$ and $Cl(2,0)$ scaling filters. In each case, Ginzberg's numerical method, adapted to cater for any type of scaling filter whose coefficients are real square matrices, gave us related wavelet filters. We found that the lengths of all of our filters are always twice an odd number. In order to check that our filters are likely to be genuine scaling and wavelet filters, we found a couple of real and complex scaling and wavelet filters using Ginzberg's method and confirmed that they do agree with published examples found by other methods. In addition, we found the frequency response of scalar, **i**, **j** and **k** and scalar, vector and bivector parts as if they were real filters and then by taking the square root of the sum of the squares of the two or four parts for each filter, we were able to demonstrate that the scaling filters are low pass and the wavelet filters are high pass. In fact the total frequency responses were virtually identical, visually, to those of real Daubechies scaling and wavelet filters of the same length.

We then used the cascade algorithm and found vectors of samples from complex, quaternion and Clifford $Cl(1,0)$, $Cl(1,1)$ and $Cl(2,0)$ scaling and wavelet functions. With real filters, the resulting functions are easily viewed in two dimensions and there is only one possible vantage point for each function. However, with a vector of four dimensional samples (or even just a two dimensional one), viewing each component separately cannot allow one to fully appreciate the complexity of the scaling or wavelet function. Hence we visualised our four dimensional functions using all possible projections onto two and three dimensions, giving 20 plots altogether for each one; for the two dimensional functions there were four plots each.

Finally, we demonstrated an application of our filters by finding hypercomplex wavelet transforms of four colour vector test images.

7.2 Contributions

We extended what Ginzberg did and found more quaternion and the first Clifford scaling and wavelet filters. Ours are ones that make use of the multiplicative properties of quaternions and Clifford algebras, rather than simply using the quaternion 1, **i**, **j** and **k** as mere labels, as the most popular 'quaternion wavelet transform' does. We discovered that filters whose coefficients are full and partial conjugates of other valid scaling and wavelet filters, are also valid scaling and wavelet filters. We showed that the frequency responses of our filters are virtually identical to those of real Daubechies filters of the same length, which is evidence that our filters really are indeed genuine hypercomplex Daubechies scaling and wavelet filters.

We found a way of visualising the functions, derived from our filters from using the cascade algorithm, that showed the complexity of these five dimensional objects; these dimensions are four spatial plus a fifth, that would be time if we used our filters to analyse a time-varying hypercomplex signal. We plotted these objects using all possible projections onto two and three dimensions, 20 altogether for each filter. The conventional way of visualising them would be to plot the scalar and three imaginary parts separately against time, only four plots in total. Our method revealed some interesting shapes that could be related to the symmetry or asymmetry, or a combination of the two, of the related filter. Most of the functions show a helical form, winding first in one sense and then unwinding in the opposite sense. All displayed finite support, as expected of scaling and wavelet functions.

Using our scaling and wavelet filters arranged in a certain way inside banded matrices, we found the very first hypercomplex wavelet transforms of four separate colour test images. We represented these as colour vector images, which are simply matrices of pure quaternions, or the Clifford equivalents, where the numbers multiplying the imaginary part are the red, green and blue values from the colour image. The product of our banded matrices with the colour vector images were matrices of full quaternions, from which we extracted the scalar and vector parts and then for each one, the approximation, horizontal, vertical and diagonal detail images. We displayed the scalar and vector results side by side.

The complex numbers and the Clifford $Cl(1, 0)$ algebra are isomorphic to certain two dimensional subalgebras of the quaternions and Clifford $Cl(1,1)$ and $Cl(2,0)$ algebras. This observation would allow us to generate many more hypercomplex scaling and wavelet filters with four dimensional filter coefficients, by using the values of complex and $Cl(1,0)$ filter coefficients in the scalar and one appropriate imaginary part of the four dimensional filter coefficients. We actually demonstrated wavelet transforms only using the values from length 6 complex and $Cl(1,0)$ filters, due to the large number of filters that resulted: to do the same with longer lengths would have resulted in an excessive number of images, many of which would be visually very similar to each other.

This thesis could have been written before the advent of the quaternion toolbox by Sangwine and Bihan [128] and the Clifford multivector toolbox by Sangwine and Hitzer [130], both for MATLAB®, in which case we would have had to use real matrix representations of hypercomplex numbers. An $n \times n$ pixel image (an $n \times n \times 3$ real array) would then be represented as a real $4n \times 4n$ matrix instead of a hypercomplex $n \times n$ matrix and our hypercomplex $n \times n$ banded matrices in Chapter 6 would then become $4n \times 4n$ banded matrices formed of the matrix-valued scaling and wavelet filters we found in Chapter 5. However, thanks to the two toolboxes, we avoided the additional complications that would have ensued.

7.3 Limitations

Solving Ginzberg's equations symbolically for length 18 quaternion and length 14 Clifford $Cl(1,0)$ scaling filters was not possible and we had to resort to a genetic algorithm for most of the nonlinear equations; see Appendix B, `eqn7` to `eqn11` and `f1` to `f5` in the Maple™ code and the fitness function in Appendix C for examples of the sort of nonlinear equations we are talking about. There is always a danger that with only sixteen digits of precision available in MATLAB®, that the values found for the unknown variables, when put back into the original nonlinear equation, will give a result close to zero, but not identically zero. Then we would not have true solutions to the equations. For example, the complex scaling filters of Zhang *et al.* [166] seem to come in pairs after the length 6 one and this suggests that perhaps $Cl(1,0)$ might also come in pairs: but we found three apparently genuine length 14 $Cl(1,0)$ scaling filters. With longer filters, the number of nonlinear equations to solve simultaneously increases and compounds the problem as the number of possible false 'solutions' also increases.

Comparing wavelet transforms of colour images, found using different lengths of filters, is perhaps not the best way to demonstrate the effects of varying the filters' lengths. As we saw, visually the wavelet transforms found using length 10 filters are virtually indistinguishable from the wavelet transform found using length 18 filters. Wavelets are ideally suited to analysing signals varying in time and frequency, neither of which are directly related to two dimensional arrays of static data, *i.e.* colour images. However, we are not aware of any quaternion signals, never mind ones that vary in time and frequency, so we are left with colour images to analyse.

7.4 Suggestions for Future Research

Going forward, we should like to explain why the lengths of all our filters were twice odd numbers. A number of articles on complex wavelet filters state that filter lengths are $2J+2$, $J \in 2\mathbb{Z}^+$, but do not explain why that is. A simple explanation is given by Sherlock and Kakad [134]: it is a direct result of finding complex solutions of the frequency response equation that Daubechies used. It may be shown, *e.g.* Smith and Barnwell [136] or Zhang *et al.* [166], that Equation (3.7) may be factored into

$$P(z) = C(1+z^{-1})^N \prod_{m=1}^{M} (z^{-1} - r_m)(z^{-1}) \prod_{j=1}^{J} (z^{-1} - c_j)(z^{-1} - c_j^{-1})(z^{-1} - \overline{c}_j)(z^{-1} - \overline{c}_j^{-1}),$$

where the r_m are the real zeros (which Daubechies was interested in), the c_j are the complex zeros and $2M + 4J = N - 2$. Using the notation of Section 3.5, we get $2N - 2$ values $z_i = 1 - 2y_i \pm \sqrt{(1-2y_i)^2 - 1}$. Of these, we keep the $(2N-2)/4$ roots that have $|z_i| > 1$ and positive real part and sort them in order of increasing real part. We take the conjugate of every second root and add the inverses to give a total of $N - 1$ roots. Finally, we add N roots at $z = -1$ to give a total of $2N - 1$. Then the $2N$ complex coefficients of the resulting polynomial are also those of a complex scaling filter. For $(2N-2)/4$ to be an integer, N must be odd. Hence complex scaling (and wavelet) filters must be twice an odd number in length. Daubechies was interested only in minimum phase and kept just the real solutions. Could a similar method, or some other method, be devised to find hypercomplex scaling filters from which the length restriction was obvious?

Are there tighter restrictions on lengths of higher dimensional full scaling filters, or are they all twice an odd number in length? Of course, putting complex filter values into higher dimensional filters ensures that the lengths are always twice an odd number, but is this always true if instead we have non-zero vectors, bivectors, trivectors and higher order multivector elements, according to the exact Clifford signature?

We know that the shortest real scaling and wavelet filters are those of the Haar wavelet, which is length 2, the shortest complex and $Cl(1,0)$ ones are length 6 and the shortest full quaternion, $Cl(1,1)$ and $Cl(2,0)$ ones are length 10. Would the shortest full $Cl(0,3)$, $Cl(1,2)$, $Cl(2,1)$ $Cl(3,0)$ scaling and wavelet filters be length 14 and for general $Cl(p,q)$, would the shortest full scaling and wavelet filters be length $4(p+q)+2$?

Can a deeper theory of hypercomplex wavelets be derived, akin to Daubechies' 'Ten Lectures on Wavelets' [29] for real wavelets, to answer the above questions?

We mentioned in Section 7.3 a drawback of using MATLAB®'s genetic algorithm function with only 16 digits of precision available. There is no genetic algorithm available in Maple™, but there is the Optimization Package, which we have not yet fully investigated. Maple™ can do calculations with many more than just 16 digits of precision and it would remove the need to duplicate everything in MATLAB®.

A programme that takes the Clifford signature and length of filter required as inputs and returns one or more hypercomplex scaling filters and associated wavelet filters as output(s) would be something to aim for. Exactly how it would work we cannot yet say, but it is something that could be a possibly long-term goal.

We have only used our filters to find wavelet transforms of colour vector images. What other uses can we find for them? Could we use them for one or more of the applications listed in Table 4.1? Would our filters actually be an improvement or would they make no difference?

The reason there are three primary colours is that the responses of the photoreceptors in the retina of the human eye peak at wavelengths of 564nm, 534nm and 420nm, *i.e.* red, green and blue. Could we use our filters to analyse signals outside the visible spectrum? It is not immediately obvious how three wavelengths could be chosen, nor what the wavelet transform of such a signal would tell us. If it were useful, perhaps full Clifford $Cl(p, q)$ filters where $p + q > 2$ might find a practical use.

Could octonion scaling and wavelet filters be found? Ginzberg's equations could not be used in matrix form, but simply using octonions with symbolic coefficients should be possible. The nonlinear equations would only involve products of two octonions, so it might be possible to find an octonion scaling filter. However, the method we have used to find quaternion and Clifford wavelet filters relies on the fact that the scaling filters can be represented as vectors of real matrices, so another way of finding wavelet filters would be required for the octonion case. We are assuming, of course, that Ginzberg's equations would actually be soluble.

Bibliography

[1] S. Agaian, C. M. Mosquera-Lopez and A. Greenblatt. Systems and methods for quantitative analysis of histopathology images using multiclassifier ensemble schemes, 2016. US Patent App. 15/028,314, https://patentimages.storage.googleapis.com/14/d9/81/e76bcfd169db26/US20160253466A1.pdf, accessed 27th July 2018.

[2] D. Alfsmann, H. G. Göckler, S. J. Sangwine and T. A. Ell. Hypercomplex algebras in digital signal processing: Benefits and drawbacks. In *Proceedings of the 15th European Signal Processing Conference (EUSIPCO 2007) held in Poznań, Poland*, pages 1322–1326. IEEE, 2007. http://www.eurasip.org/Proceedings/Eusipco/Eusipco2007/Papers/c2l-b01.pdf, accessed 27th July 2018.

[3] S. Arivazhagan, R. A. Priyadharshini and L. Sangeetha. Automatic target recognition in SAR images using quaternion wavelet transform and principal component analysis. *International Journal of Computational Vision and Robotics*, 7(3):314–334, 2017. https://doi.org/10.1504/IJCVR.2017.083449.

[4] B. Augereau and P. Carré. Hypercomplex polynomial wavelet-filter bank transform for color image. *Signal Processing*, 136:16–28, 2017. https://doi.org/10.1016/j.sigpro.2016.11.022.

[5] M. Bahri. Construction of quaternion-valued wavelets. *Matematika*, 26(1):107–114, 2010. https://doi.org/10.11113/matematika.v26.n.553.

[6] M. Bahri. Quaternion algebra-valued wavelet transform. *Applied Mathematical Sciences*, 5(71):3531–3540, 2011. http://www.m-hikari.com/ams/ams-2011/ams-69-72-2011/bahriAMS69-72-2011.pdf, accessed 27th July 2018.

[7] M. Bahri, R. Ashino and R. Vaillancourt. Two-dimensional quaternion wavelet transform. *Applied Mathematics and Computation*, 218(1):10–21, 2011. https://doi.org/10.1016/j.amc.2011.05.030.

[8] M. Bahri, R. Ashino and R. Vaillancourt. Two-dimensional quaternion Fourier transform of type II and quaternion wavelet transform. In *Proceedings of the 2012 Interna-*

tional Conference on Wavelet Analysis and Pattern Recognition (ICWAPR 2012), held in Xian, China, pages 359–364. IEEE, 2012. https://doi.org/10.1109/ICWAPR.2012.6294808.

[9] M. Bahri, R. Ashino and R. Vaillancourt. Continuous quaternion Fourier and wavelet transforms. *International Journal of Wavelets, Multiresolution and Information Processing*, 12(4):1460003, 2014. https://doi.org/10.1142/S0219691314600030.

[10] E. Bayro-Corrochano. Multi–resolution image analysis using the quaternion wavelet transform. *Numerical Algorithms*, 39(1–3):35–55, 2005. https://doi.org/10.1007/s11075-004-3619-8.

[11] E. Bayro-Corrochano. The theory and use of the quaternion wavelet transform. *Journal of Mathematical Imaging and Vision*, 24(1):19–35, 2006. https://doi.org/10.1007/s10851-005-3605-3.

[12] E. Bayro-Corrochano and M. de La Torre Gomora. Image processing using the quaternion wavelet transform. In *Progress in Pattern Recognition, Image Analysis and Applications: Proceedings of the 9th IberoAmerican Congress on Pattern Recognition (CIARP 2004), held in Puebla, Mexico*, volume 3287 of *Lecture Notes in Computer Science*, pages 613–620. Springer: Berlin and Heidelberg, 2004. https://doi.org/10.1007/978-3-540-30463-0_77.

[13] B. E. Blank. Book review of 'An Imaginary Tale: The Story of $\sqrt{-1}$'. *Notices of the American Mathematical Society*, 46(10):1233–1236, 1999. http://www.ams.org/journals/notices/199910/rev-blank.pdf, accessed 27th July 2018.

[14] C. Blatter. *Wavelets: A Primer.* A. K. Peters, Natick, MA, 1998. https://www.taylorfrancis.com/books/9781439863978, accessed 27th July 2018.

[15] S.-T. Bow. *Pattern Recognition and Image Preprocessing.* Signal Processing and Communications. Marcel Dekker, New York, 2nd edition, 2002. http://www.crcnetbase.com/isbn/9780203903896, accessed 27th July 2018.

[16] F. Brackx, N. D. Schepper and F. Sommen. The two-dimensional Clifford–Fourier transform. *Journal of Mathematical Imaging and Vision*, 26(1–2):5–18, 2006. https://doi.org/10.1007/s10851-006-3605-y.

[17] T. Bülow. *Hypercomplex Spectral Signal Representations for the Processing and Analysis of Images.* PhD thesis, Institut für Informatik und Praktische Mathematik der Christian-Albrechts-Universität, Kiel, Germany, 1999. http://www.uni-kiel.de/journals/receive/jportal_jparticle_00000190, accessed 27th July 2018.

[18] P. Carré and P. Denis. Quaternionic wavelet transform for colour images. *Proceedings*

of SPIE: Wavelet Applications in Industrial Processing IV, held in Boston, MA, 6383: 638301, 2006. https://doi.org/10.1117/12.685942.

[19] P. Chai, X. Luo and Z. Zhang. Image fusion using quaternion wavelet transform and multiple features. *IEEE Access*, 5:6724–6734, 2017. https://doi.org/10.1109/ACCESS.2017.2685178.

[20] W. L. Chan, H. Choi and R. Baraniuk. Quaternion wavelets for image analysis and processing. In *Proceedings of the 2004 IEEE International Conference on Image Processing (ICIP 2004), held in Singapore*, volume 5, pages 3057–3060. IEEE, 2004. https://doi.org/10.1109/ICIP.2004.1421758.

[21] W. L. Chan, H. Choi and R. Baraniuk. Directional hypercomplex wavelets for multidimensional signal analysis and processing. In *Proceedings of the 2004 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2004), held in Montreal, Canada*, volume III, pages 996–999. IEEE, 2004. https://doi.org/10.1109/ICASSP.2004.1326715.

[22] W. L. Chan, H. Choi and R. Baraniuk. Coherent image processing using quaternion wavelets. *Proceedings of SPIE: Wavelets XI, held in San Diego, CA*, 5914:59140Z, 2005. https://doi.org/10.1117/12.615393.

[23] W. L. Chan, H. Choi and R. G. Baranuik. Multiscale image disparity estimation using the quaternion wavelet transform. In *Proceedings of the 2006 IEEE International Conference on Image Processing (ICIP 2006), held in Atlanta, Georgia*, pages 1229–1232. IEEE, 2006. https://doi.org/10.1109/ICIP.2006.312547.

[24] W. L. Chan, H. Choi and R. G. Baraniuk. Coherent multiscale image processing using dual-tree quaternion wavelets. *IEEE Transactions on Image Processing*, 17(7): 1069–1082, 2008. https://doi.org/10.1109/TIP.2008.924282.

[25] J. M. Chappell, S. P. Drake, C. L. Seidel, L. J. Gunn, A. Iqbal, A. Allison and D. Abbott. Geometric algebra for electrical and electronic engineers. *Proceedings of the IEEE*, 102(9):1340–1363, 2014. https://doi.org/10.1109/JPROC.2014.2339299.

[26] W. K. Clifford. Applications of Grassmann's extensive algebra. *American Journal of Mathematics*, 1(4):350–358, 1878. https://doi.org/10.2307/2369379.

[27] W. K. Clifford. On the classification of geometric algebras. In R. Tucker and H. J. S. Smith, editors, *Mathematical Papers by William Kingdon Clifford*, chapter 43, pages 397–401. Macmillan and Co., London, 1882. https://archive.org/stream/mathematicalpap00smitgoog#page/n489/mode/2up.

[28] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Communications*

on Pure and Applied Mathematics, 41(7):909–996, 1988. https://doi.org/10.1002/cpa.3160410705.

[29] I. Daubechies. *Ten Lectures on Wavelets*. CBMS-NSF Regional Conference Series in Applied Mathematics. The Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992. https://doi.org/10.1137/1.9781611970104.

[30] I. Daubechies and J. C. Lagarias. Two-scale difference equations I. Existence and global regularity of solutions. *SIAM Journal on Mathematical Analysis*, 22(5):1388–1410, 1991. https://doi.org/10.1137/0522089 or https://pdfs.semanticscholar.org/e6e0/a58a11c2d81d4bf09c02322d88879cee05fd.pdf, accessed 6th May 2019.

[31] I. Daubechies and J. C. Lagarias. Two-scale difference equations II. Local regularity, infinite products of matrices and fractals. *SIAM Journal on Mathematical Analysis*, 23(4):1031–1079, 1992. https://doi.org/10.1137/0523059 or https://pdfs.semanticscholar.org/a65d/ec6715cc20f5331e96fc2719af73c2b3b7be.pdf, accessed 6th May 2019.

[32] P. Denis, P. Carre and C. Fernandez-Maloigne. Spatial and spectral quaternionic approaches for colour images. *Computer Vision and Image Understanding*, 107(1–2): 74–87, 2007. https://doi.org/10.1016/j.cviu.2006.11.019.

[33] A. Diek and R. Kantowski. Some Clifford Algebra History. In R. Ablamowicz and P. Lounesto, editors, *Clifford Algebras and Spinor Structures*, volume 321 of *Mathematics and Its Applications*, pages 3–12. Springer, Dordrecht, the Netherlands, 1995. https://doi.org/10.1007/978-94-015-8422-7_1.

[34] X. Ding, Y. Xu, L. Deng and X. Yang. Colorization using quaternion algebra with automatic scribble generation. In *Advances in Multimedia Modeling: Proceedings of the 18th International Conference on Multimedia Modeling (MMM 2012) held in Klagenfurt, Austria*, volume 7131 of *Lecture Notes in Computer Science*, pages 103–114. Springer, Berlin and Heidelberg, 2012. https://doi.org/10.1007/978-3-642-27355-1_12.

[35] T. A. Ell. Quaternion-Fourier transforms for analysis of 2-dimensional linear time-invariant partial-differential systems. In *Proceedings of the 32nd IEEE Conference on Decision and Control, held in San Antonio, TX*, volume 2, pages 1830–1841. IEEE, 1993. https://doi.org/10.1109/CDC.1993.325510.

[36] T. A. Ell and S. J. Sangwine. Hypercomplex Fourier transforms of color images. *IEEE Transactions on Image Processing*, 16(1):22–35, 2007. https://doi.org/10.1109/TIP.2006.884955.

[37] T. A. Ell, N. L. Bihan and S. J. Sangwine. *Quaternion Fourier Transforms for Signal and Image Processing*. Digital and Image Processing Series. ISTE Ltd, London, 2014.

https://doi.org/10.1002/9781118930908.

[38] J. C. Familton. *Quaternions: A history of complex noncommutative rotation groups in theoretical physics.* PhD thesis, Columbia University, 2015. https://doi.org/10.7916/D8FB521P.

[39] R. W. Farebrother, J. Groß and S.-O. Troschke. Matrix representations of quaternions. *Linear Algebra and its Applications*, 362:251–255, 2003. https://doi.org/10.1016/S0024-3795(02)00535-9.

[40] J. Fauvel. *Early Mathematics.* The Open University, Milton Keynes, UK, 1987. Unit 1 of the second level module MA290 Topics in the History of Mathematics.

[41] D. Fearnley-Sander. Hermann Grassmann and the creation of linear algebra. *The American Mathematical Monthly*, 86(10):809–817, 1979. https://doi.org/10.2307/2320145.

[42] G. Flegg. *From the Greeks to the Renaissance.* The Open University, Milton Keynes, UK, 1987. Unit 5 of the second level module MA290 Topics in the History of Mathematics.

[43] P. Fletcher. Discrete wavelets with quaternion and Clifford coefficients. *Advances in Applied Clifford Algebras*, 28(3):59, 2018. https://doi.org/10.1007/s00006-018-0876-5.

[44] P. Fletcher and S. J. Sangwine. The development of the quaternion wavelet transform. *Signal Processing*, 136:2–15, 2017. https://doi.org/10.1016/j.sigpro.2016.12.025.

[45] J. Friberg. *Unexpected Links between Egyptian and Babylonian Mathematics*, chapter 2, Hieratic Mathematical Papyri and Cuneiform Mathematical Texts, pages 25–104. World Scientific, Singapore, 2005. https://doi.org/10.1142/9789812701121_0002.

[46] R. Fröberg. *An Introduction to Gröbner Bases.* John Wiley & Sons, 1997. https://www.wiley.com/en-gb/An+Introduction+to+Gr%26ouml%3Bbner+Bases-p-9780470866207, accessed 27th July 2018.

[47] D. Gabor. Theory of communication Part 1 - The analysis of information. *The Journal of the Institute of Electrical Engineers - Part III: Radio and Communication Engineering*, 93(26):429–441, 1946. https://doi.org/10.1049/ji-3-2.1946.0074.

[48] D. Gabor. Theory of communication Part 2 - The analysis of hearing. *The Journal of the Institute of Electrical Engineers - Part III: Radio and Communication Engineering*, 93(26):442–445, 1946. https://doi.org/10.1049/ji-3-2.1946.0075.

[49] S. Gai. New banknote defect detection algorithm using quaternion wavelet transform. *Neurocomputing*, 196:133–139, 2016. https://doi.org/10.1016/j.neucom.2015.12.112.

[50] S. Gai and L. Luo. Visual objects tracking and identification based on reduced quaternion wavelet transform. *Signal, Image and Video Processing*, 8(Supplement 1):75—-84, 2014. https://doi.org/10.1007/s11760-014-0636-5.

[51] S. Gai and L. Luo. Image denoising using normal inverse Gaussian model in quaternion wavelet domain. *Multimedia Tools and Applications*, 74(3):1107–1124, 2015. https://doi.org/10.1007/s11042-013-1812-2.

[52] S. Gai, P. Liu, J. Liu and X. Tang. Banknote image retrieval using rotated quaternion wavelet filters. *International Journal of Computational Intelligence Systems*, 4(2):268–276, 2011. https://doi.org/10.1080/18756891.2011.9727782.

[53] S. Gai, G. Yang and S. Zhang. Banknote classification by using contourlet and quaternion wavelet features. *ICIC Express Letters*, 6(9):2295–2300, 2012.

[54] S. Gai, G. Yang and M. Wan. Employing quaternion wavelet transform for banknote classification. *Neurocomputing*, 118:171–178, 2013. https://doi.org/10.1016/j.neucom.2013.02.029.

[55] S. Gai, G. Yang and S. Zhang. New feature extraction approach for bank note classification using quaternion wavelets. *Journal of Intelligent and Fuzzy Systems*, 25(3):685–694, 2013. https://doi.org/10.3233/IFS-120675.

[56] S. Gai, G. Yang and S. Zhang. Multiscale texture classification using reduced quaternion wavelet transform. *International Journal of Electronics and Communications (AEÜ)*, 67(3):233–241, 2013. https://doi.org/10.1016/j.aeue.2012.08.004.

[57] S. Gai, G. Yang, M. Wan and L. Wang. Hidden Markov tree model of images using quaternion wavelet transform. *Computers and Electrical Engineering*, 40(3):819–832, 2014. https://doi.org/10.1016/j.compeleceng.2014.02.009.

[58] R. X. Gao and R. Yan. From Fourier transform to wavelet transform: A historical perspective. In *Wavelets: Theory and Applications for Manufacturing*, chapter 2, pages 17–32. Springer US, 2011. https://doi.org/10.1007/978-1-4419-1545-0_2.

[59] P. Geng, X. Su and T. Xu. Medical image fusion based on quaternion wavelet transform and visibility feature. *International Journal of Applied Mathematics and Machine Learning*, 2(1):9–26, 2015. https://pdfs.semanticscholar.org/28fd/0a4d7bc349f16c6be3f7a15b35a4cdf763a5.pdf, accessed 27th July 2018.

[60] P. Geng, X. Sun and J. Liu. Adopting quaternion wavelet transform to fuse multi-

modal medical images. *Journal of Medical and Biological Engineering*, 37(2):230––239, 2017. https://doi.org/10.1007/s40846-016-0200-6.

[61] P. Ginzberg. *Quaternion Matrices: Statistical Properties and Applications to Signal Processing and Wavelets.* PhD thesis, Imperial College Department of Mathematics, London, UK, 2013. http://hdl.handle.net/10044/1/18975, accessed 27th July 2018.

[62] P. Ginzberg and A. T. Walden. Matrix-valued and quaternion wavelets. *IEEE Transactions on Signal Processing*, 61(6):1357–1367, 2013. https://doi.org/10.1109/TSP.2012.2235434.

[63] J. C. Goswami and A. K. Chan. *Fundamentals of Wavelets: Theory, Algorithms and Applications.* Microwave and Optical Engineering. John Wiley and Sons, 2nd edition, 2011. https://doi.org/10.1002/9780470926994.

[64] H. Grassmann. Sur les différents genres de multiplication. *Journal für die reine und angewandte Mathematik ('Crelle's Journal')*, 49(2):123–141, 1855. https://gdz.sub.uni-goettingen.de/id/PPN243919689_0049?tify={%22pages%22:[127],%22view%22:%22info%22}, accessed 13th June 2018.

[65] H. Grassmann. *Die lineale Ausdehnungslehre.* Cambridge University Press, 2013. Reprint of the original 1844 book, as updated in 1862. https://doi.org/10.1017/CBO9781107325081.

[66] A. Greenblatt, C. Mosquera-Lopez and S. Agaian. Quaternion neural networks applied to prostate cancer Gleason grading. In *Proceedings of the 2013 IEEE International Conference on Systems, Man and Cybernetics (SMC 2013), held in Manchester, UK*, pages 1144–1149. IEEE, 2013. https://doi.org/10.1109/SMC.2013.199.

[67] L. Guilbeau. The history of the solution of the cubic equation. *Mathematics News Letter*, 5(4):8–12, 1930. https://doi.org/10.2307/3027812.

[68] L. Guo, M. Dai and M. Zhu. Multifocus color image fusion based on quaternion curvelet transform. *Optics Express*, 20(17):18846–18860, 2012. https://doi.org/10.1364/OE.20.018846.

[69] A. Haar. Zur Theorie der orthogonalen Funktionensysteme. *Mathematische Annalen*, 69(3):331–371, 1910. https://doi.org/10.1007/BF01456326.

[70] W. R. Hamilton. Theory of conjugate functions, or algebraic couples; with a preliminary and elementary essay on algebra as the science of pure time. *The Transactions of the Royal Irish Academy*, 17:293–423, 1833. http://www.jstor.org/stable/30078796.

[71] S. Han, J. Yang, R. Wang and G. Jia. A novel color image watermarking algorithm based on QWT and DCT. In J. Yang, Q. Hu, M.-M. Cheng, L. Wang, Q. Liu, X. Bai and D. Meng, editors, *Computer Vision: Proceedings of the Second CCF Chinese Conference on Computer Vision (CCCV 2017), held in Tianjin, China*, volume 771 of *Communications in Computer and Information Science*, pages 428–438. Springer, Singapore, 2017. https://doi.org/10.1007/978-981-10-7299-4_35.

[72] S. Han, J. Yang, R. Wang and G. Jia. A robust color image watermarking algorithm against rotation attacks. *Optoelectronics Letters*, 14(1):61–66, 2018. https://doi.org/10.1007/s11801-018-7212-0.

[73] J.-X. He and B. Yu. Wavelet analysis of quaternion–valued time–series. *International Journal of Wavelets, Multiresolution and Information Processing*, 3(2):233–246, 2005. https://doi.org/10.1142/S0219691305000804.

[74] E. Hitzer and S. J. Sangwine. Multivector and multivector matrix inverses in real Clifford algebras. *Applied Mathematics and Computation*, 311:375–389, 2017. https://doi.org/10.1016/j.amc.2017.05.027.

[75] J. A. Hogan and A. J. Morris. Quaternionic wavelets. *Numerical Functional Analysis and Optimization*, 33(7-9):1031–1062, 2012. https://doi.org/10.1080/01630563.2012.682140.

[76] J. Jin, Y. Liu, Q. Wang and S. Yi. Ultrasonic speckle reduction based on soft thresholding in quaternion wavelet domain. In *Proceedings of the 2012 IEEE International Conference on Instrumentation and Measurement Technology (I2MTC 2012), held in Graz, Austria*. IEEE, 2012. https://doi.org/10.1109/I2MTC.2012.6365367.

[77] M. Kadiri, M. Djebbouri and P. Carré. Image denoising using selective neighboring quaternionic wavelet coefficients. In *Proceedings of the 24th International Conference on Microelectronics (ICM 2012), held in Algiers, Algeria*. IEEE, 2012. https://doi.org/10.1109/ICM.2012.6471420.

[78] M. Kadiri, M. Djebbouri and P. Carré. Magnitude-phase of the dual-tree quaternionic wavelet transform for multispectral satellite image denoising. *EURASIP Journal on Image and Video Processing*, 41, 2014. https://doi.org/10.1186/1687-5281-2014-41.

[79] A. Katunin. Identification of stiff inclusion in circular composite plate based on quaternion wavelet analysis of modal shapes. *Journal of Vibroengineering*, 16(5):2545–2551, 2014. https://www.jvejournals.com/article/15211, accessed 27th July 2018.

[80] V. J. Katz. *A History of Mathematics, Brief Version*. China Machine Press, English reprint edition, 2004.

[81] F. Keinert. *Wavelets and Multiwavelets*. Studies in Advanced Mathematics. Chapman and Hall/CRC, Boca Raton, Florida, USA, 2003. https://doi.org/10.1201/9780203011591.

[82] F. Keinert. Multiwavelet Toolbox for MATLAB®, 2004. http://orion.math.iastate.edu/keinert/book.html, accessed 27th July 2018.

[83] M. Khelifi, A. M. Lakhdar and I. Elawady. Optimization of channel coding for transmitted image using quincunx wavelets transforms compression. *International Journal of Advanced Computer Science and Applications*, 7(5):419–424, 2016. https://doi.org/10.14569/IJACSA.2016.070556.

[84] J. Kovačević, V. K. Goyal and M. Vetterli. *Fourier and Wavelet Signal Processing*. Self-published online, 2013. http://fourierandwavelets.org/FWSP_a3.2_2013.pdf, accessed 27th July 2018.

[85] S. Kumar, S. Kumar, N. Sukavanam and B. Raman. Dual tree fractional quaternion wavelet transform for disparity estimation. *ISA Transactions*, 53(2):547–559, 2014. https://doi.org/10.1016/j.isatra.2013.12.001.

[86] V. R. V. Kumar, A. Vidya, M. Sharumathy and R. Kanizohi. Super resolution enhancement of medical image using quaternion wavelet transform with SVD. In *Proceedings of the 4th International Conference on Signal Processing, Communications and Networking (ICSCN 2017), held in Chennai, India*. IEEE, 2017. https://doi.org/10.1109/ICSCN.2017.8085687.

[87] W. Lawton. Applications of complex valued wavelet transforms to subband decomposition. *IEEE Transactions on Signal Processing*, 41(12):3566–3568, 1993. https://doi.org/10.1109/78.258098.

[88] A. C. Le Ngo, K. Li-Minn Ang, G. Qiu and J. Kah-Phooi Seng. Wavelet-based scale saliency. *ArXiv e-prints*, 2013. http://arxiv.org/abs/1301.2884v1, accessed 27th July 2018.

[89] A. C. Le Ngo, K. Li-Minn Ang, G. Qiu and J. Kah-Phooi Seng. Information-based scale saliency methods with wavelet sub-band energy density descriptors. In *Proceedings of the 5th Asian Conference on Intelligent Information and Database Systems (ACIIDS 2013), held in Kuala Lumpur, Malaysia*, volume 7803 of *Lecture Notes in Computer Science*, pages 366–376. Springer, Berlin and Heidelberg, 2013. https://doi.org/10.1007/978-3-642-36543-0_38.

[90] B. Lei, D. Ni, S. Chen, T. Wang and F. Zhou. Optimal image watermarking scheme based on chaotic map and quaternion wavelet transform. *Nonlinear Dynamics*, 78(4):2897–2907, 2014. https://doi.org/10.1007/s11071-014-1634-4.

[91] B. Lei, F. Zhou, E.-L. Tan, D. Ni, H. Lei, S. Chen and T. Wang. Optimal and secure audio watermarking scheme based on self-adaptive particle swarm optimization and quaternion wavelet transform. *Signal Processing*, 113:80–94, 2015. https://doi.org/10.1016/j.sigpro.2014.11.007.

[92] P. Lévy. *Processus Stochastiques et Mouvement Brownien.* Gautier-Villars, Paris, 1948.

[93] C. Li, J. Li and B. Fu. Magnitude-phase of quaternion wavelet transform for texture representation using multilevel copula. *IEEE Signal Processing Letters*, 20(8):799–802, 2013. https://doi.org/10.1109/LSP.2013.2247596.

[94] C.-R. Li, J.-P. Li, X.-C. Yang and Z.-W. Liang. Gait recognition using the magnitude and phase of quaternion wavelet transform. In *Proceedings of the 2012 International Conference on Wavelet Active Media Technology and Information Processing (ICWAMTIP 2012), held in Chengdu, China*, pages 322–324. IEEE, 2012. https://doi.org/10.1109/ICWAMTIP.2012.6413504.

[95] J. M. Lilly. Modulated oscillations in three dimensions. *IEEE Transactions on Image Processing*, 59(12):5930–5943, 2011. https://doi.org/10.1109/TSP.2011.2164914.

[96] J.-M. Lina and M. Mayrand. Parameterizations for complex Daubechies wavelets. In *Proceedings of SPIE: Wavelet Applications, held in Orlando, FL*, volume 2242, pages 868–877. SPIE, 1994. https://doi.org/10.1117/12.170087.

[97] H. Liu, C. Wang, J. Lu, Z. Tang and J. Yang. Maximum likelihood estimation of monocular optical flow field for mobile robot ego-motion. *International Journal of Advanced Robotic Systems*, 13(1):12, 2016. https://doi.org/10.5772/62157.

[98] Y. Liu and W. Du. The image sharpness metric via Gaussian mixture modeling of the quaternion wavelet transform phase coefficients with applications. In R. Lee, editor, *Applied Computing & Information Technology: Proceedings of the 3rd International Conference on Applied Computing and Information Technology (ACIT 2015), held in Okayama, Japan*, volume 619 of *Studies in Computational Intelligence*, pages 167–178. Springer, 2016. https://doi.org/10.1007/978-3-319-26396-0_13.

[99] Y. Liu, J. Jin, Q. Wang and Y. Shen. Phase-preserving speckle reduction based on soft thresholding in quaternion wavelet domain. *Journal of Electronic Imaging*, 21(4): 043009, 2012. https://doi.org/10.1117/1.JEI.21.4.043009.

[100] Y. Liu, J. Jin, Q. Wang and S. Yi. Ultrasound extended-field-of-view imaging based on motion estimation using quaternion wavelet. In *Proceedings of the 2012 IEEE International Conference on Instrumentation and Measurement Technology (I2MTC 2012), held in Graz, Austria.* IEEE, 2012. https://doi.org/10.1109/I2MTC.2012.6365366.

[101] Y. Liu, J. Jin, Q. Wang and Y. Shen. Phases measure of image sharpness based on quaternion wavelet. *Pattern Recognition Letters*, 34(9):1063–1070, 2013. https://doi.org/10.1016/j.patrec.2013.03.002.

[102] Y. Liu, J. Jin, Q. Wang, Y. Shen and X. Dong. Novel focus region detection method for multifocus image fusion using quaternion wavelet. *Journal of Electronic Imaging*, 22(2):023017, 2013. https://doi.org/10.1117/1.JEI.22.2.023017.

[103] Y. Liu, J. Jin, Q. Wang, Y. Shen and X. Dong. Region level based multi-focus image fusion using quaternion wavelet and normalized cut. *Signal Processing*, 97:9–30, 2014. https://doi.org/10.1016/j.sigpro.2013.10.010.

[104] Y.-P. Liu, W. Du, J. Jin, H. Wang and R. Liang. Boost image denoising via noise level estimation in quaternion wavelet domain. *AEU - International Journal of Electronics and Communications*, 70(5):584–591, 2016. https://doi.org/10.1016/j.aeue.2016.01.014.

[105] P. Lounesto. *Clifford Algebras and Spinors*. London Mathematical Society Lecture Note Series, vol. 286. Cambridge University Press, 2$^{\text{nd}}$ edition, 2001. https://doi.org/10.1017/CBO9780511526022.

[106] C. Madhu and E. Anant Shankar. Image compression using quaternion wavelet transform. *Helix*, 8(1):2691–2695, 2018. https://doi.org/10.29042/2018-2691-2695.

[107] S. G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7): 674–693, 1989. https://doi.org/10.1109/34.192463.

[108] N. Malleswari and C. Madhu. Image denoising based on quaternion wavelet transform. *International Journal of Recent Trends in VLSI,Embedded Systems and Signal Processing*, 2(4):17–20, 2016. Journal appears to be defunct.

[109] Y. Meyer. *Wavelets and operators*, volume 37 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 1992 English edition. https://doi.org/10.1017/CBO9780511623820.

[110] M. Mitrea. *Clifford Wavelets, Singular Integrals and Hardy Spaces*, volume 1575 of *Lecture Notes in Mathematics*. Springer Berlin Heidelberg, 1994. https://doi.org/10.1007/BFb0073556.

[111] C. Mosquera-Lopez, S. Agaian and A. Velez-Hoyoz. The development of a multi-stage learning scheme using new tissue descriptors for automatic grading of prostatic carcinoma. In *Proceedings of the 2014 International Conference on Acoustics, Speech and Signal Processing (ICASSP 2014), held in Florence, Italy*, pages 3586–3590. IEEE,

2014. https://doi.org/10.1109/ICASSP.2014.6854269.

[112] E. U. Moya-Sánchez and E. Bayro-Corrochano. Quaternion atomic function wavelet for applications in image processing. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: Proceedings of the 15th IberoAmerican Congress on Pattern Recognition (CIARP 2010), held in São Paulo, Brazil*, volume 6419 of *Lecture Notes in Computer Science*, pages 346–353. Springer Berlin Heidelberg, 2010. https://doi.org/10.1007/978-3-642-16687-7_47.

[113] M. Naouai, A. Hamouda, A. Akkari and C. Weber. New approach for road extraction from high resolution remotely sensed images using the quaternionic wavelet. In *Pattern Recognition and Image Analysis: Proceedings of the 5th Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA 2011) held in Las Palmas de Gran Canaria, Spain*, volume 6669 of *Lecture Notes in Computer Science*, pages 452–459. Springer Berlin Heidelberg, 2011. https://doi.org/10.1007/978-3-642-21257-4_56.

[114] W. K. Ngui, M. S. Leong, L. M. Hee and A. M. Abdelrhman. Wavelet analysis: Mother wavelet selection methods. *Applied Mechanics and Materials*, 393:953–958, 2013. https://doi.org/10.4028/www.scientific.net/AMM.393.953.

[115] J. J. O'Connor and E. F. Robertson. MacTutor biography of William Rowan Hamilton, 1998. http://www-groups.dcs.st-and.ac.uk/~history/Biographies/Hamilton.html [last accessed 6th June 2016].

[116] J. J. O'Connor and E. F. Robertson. MacTutor biography of Caspar Wessel, 1998. http://www-history.mcs.st-andrews.ac.uk/Biographies/Wessel.html [last accessed 7th June 2016].

[117] H. Pang, M. Zhu and L. Guo. Multifocus color image fusion using quaternion wavelet transform. In *Proceedings of the 2012 5th International Congress on Image and Signal Processing (CISP 2012), held in Chongqing, China*, pages 543–546. IEEE, 2012. https://doi.org/10.1109/CISP.2012.6469884.

[118] L. Peng and J. Zhao. Quaternion-valued smooth orthogonal wavelets with short support and symmetry. In *Advances in Analysis and Geometry*, Trends in Mathematics, pages 365–376. Birkhäuser Basel, 2004. https://doi.org/10.1007/978-3-0348-7838-8_18.

[119] M. A. Pinsky. Brownian continuity modulus via series expansion. *Journal of Theoretical Probability*, 14(1):261–266, 2001. https://doi.org/10.1023/A:1007837502471.

[120] R. Poli, W. B. Langdon and N. F. McPhee. *A Field Guide to Genetic Programming.* Self-publishd online, 2008. http://www.gp-field-guide.org.uk/.

[121] R. A. Priyadharshini and S. Arivazhagan. A quaternionic wavelet transform-based approach for object recognition. *Defence Science Journal*, 64(4):350–357, 2014. https://doi.org/10.14429/dsj.64.4503.

[122] T. Ren, M. Hui, X. Wang, Z. Zhang and J. Liang. Structural state inspection using dual-tree quaternion wavelet transform. *Journal of Vibroengineering*, 20(1):138–151, 2018. https://doi.org/10.21595/jve.2017.18403.

[123] S. Sangeetha, C. M. Sujatha and D. Manamalli. Anisotropic analysis of trabecular architecture in human femur bone radiographs using quaternion wavelet transforms. In *Proceedings of the 2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC 2014), held in Chicago, IL*, pages 5603–5606. IEEE, 2014. https://doi.org/10.1109/EMBC.2014.6944897.

[124] S. J. Sangwine. Fourier transforms of colour images using quaternion, or hypercomplex, numbers. *Electronics Letters*, 32(21):1979–1980, 1996. https://doi.org/10.1049/el:19961331.

[125] S. J. Sangwine. Colour image edge detector based on quaternion convolution. *Electronics Letters*, 34(10):969–971, 1998. https://doi.org/10.1049/el:19980697.

[126] S. J. Sangwine. Perspectives on color image processing by linear vector methods using projective geometric transformations. In *Advances in Imaging and Electron Physics*, volume 175, chapter 6, pages 283–307. Academic Press: Elsevier BV, Amsterdam, 2013. https://doi.org/10.1016/B978-0-12-407670-9.00006-8.

[127] S. J. Sangwine. On harmonic analysis of vector-valued signals. *Mathematical Methods in the Applied Sciences*, 40(1):22–30, 2016. https://doi.org/10.1002/mma.3938.

[128] S. J. Sangwine and N. L. Bihan. Quaternion Toolbox for MATLAB®, 2005. Software library available at: http://qtfm.sourceforge.net/.

[129] S. J. Sangwine and T. A. Ell. The discrete Fourier transform of a colour image. In J. M. Blackledge and M. J. Turner, editors, *Image Processing II: Mathematical Methods, Algorithms and Applications: Proceedings of the Second IMA Conference on Image Processing, held at De Montfort University, Leicester in 1998*, pages 430–441, Chichester, UK, 2000. Horwood.

[130] S. J. Sangwine and E. Hitzer. Clifford Multivector Toolbox for MATLAB®, 2015. Software library available at: http://clifford-multivector-toolbox.sourceforge.net/.

[131] S. J. Sangwine and E. Hitzer. Clifford multivector toolbox (for MATLAB). *Advances in Applied Clifford Algebras*, 27(1):539—558, 2017. https://doi.org/10.

1007/s00006-016-0666-x.

[132] B. Sathyabama, P. Chitra, V. G. Devi, S. Raju and V. Abhaikumar. Quaternion wavelets based rotation, scale and translation invariant texture classification and retrieval. *Journal of Scientific and Industrial Research*, 70(4):256–263, 2011. http://nopr.niscair.res.in/handle/123456789/11583, accessed 27th July 2018.

[133] Y. Shen, H. Feng, Q. Wang, Y. Liu and Z. He. QWT enhanced SVM for hyperspectral image classification. In *Proceedings of the 2014 IEEE International Conference on Instrumentation and Measurement Technology (I2MTC 2014), held in Montevideo, Uruguay*, pages 1454–1458. IEEE, 2014. https://doi.org/10.1109/I2MTC.2014.6860986.

[134] B. G. Sherlock and Y. P. Kakad. MATLAB programs for generating orthonormal wavelets. In *Proceedings of the WSEAS International Conferences (ICONEMC 2002, ICOSMO 2002, ICOSSIP 2002, ICOMIV 2002, ICRODIC 2002), held in Skiathos, Greece*, pages 1461–1465. World Scientific and Engineering Society Press, 2002. http://www.wseas.us/e-library/conferences/skiathos2002/papers/447-146.pdf, accessed 19th Sept. 2018.

[135] L. Shi. Exploration in quaternion colour. MSc dissertation, Simon Fraser University, Burnaby, Canada, 2005. http://summit.sfu.ca/item/9829.

[136] M. Smith and T. Barnwell. Exact reconstruction techniques for tree-structured subband coders. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(3): 434–441, 1986. https://doi.org/10.1109/TASSP.1986.1164832.

[137] R. Soulard and P. Carré. Quaternionic wavelets for texture classification. In *Proceedings of the 2010 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2010), held in Dallas, USA*, pages 4134–4137. IEEE, 2010. https://doi.org/10.1109/ICASSP.2010.5495732.

[138] R. Soulard and P. Carré. Quaternionic wavelets for image coding. In *Proceedings of the 18th European Signal Processing Conference (EUSIPCO 2010), held in Aalborg, Denmark*, pages 125–129. European Association for Signal, Speech, and Image Processing, 2010. https://hal.archives-ouvertes.fr/hal-00514423.

[139] R. Soulard and P. Carré. Quaternionic wavelets for texture classification. *Pattern Recognition Letters*, 32(13):1669–1678, 2011. https://doi.org/10.1016/j.patrec.2011.06.028.

[140] G. Strang. Fast transforms: Banded matrices with banded inverses. *Proceedings of the National Academy of Sciences of the United States of America*, 107(28):12413–12416, 2010. https://doi.org/10.1073/pnas.1005493107.

[141] G. Strang and T. Nguyen. *Wavelets and Filter Banks.* Wellesley-Cambridge Press, revised edition edition, 1997.

[142] J.-O. Strömberg. A modified Franklin system and higher order spline systems on $\mathbb{R}^n$ as unconditional bases for Hardy spaces. In *Fundamental Papers in Wavelet Theory*, pages 197–215. Princeton University Press, 2006. Original paper from the Proceedings of the Conference on Harminic Analysis in Honor of Antoni Zygmund, held in Chicago, IL, in 1983. http://www.jstor.org/stable/j.ctt7rt5v.34.

[143] L. Tang, L. Li, K. Sun, Z. Xia, K. Gu and J. Qian. An efficient and effective blind camera image quality metric via modeling quaternion wavelet coefficients. *Journal of Visual Communication and Image Representation*, 49:204–212, 2017. https://doi.org/10.1016/j.jvcir.2017.09.010.

[144] A. Traoré, P. Carré and C. Olivier. Reduced-reference metric based on the quaternionic wavelet coefficients modeling by information criteria. In *Proceedings of the 2014 IEEE International Conference on Image Processing (ICIP 2014), held in Paris*, pages 526–530. IEEE, 2014. https://doi.org/10.1109/ICIP.2014.7025105.

[145] A. Traoré, P. Carré and C. Olivier. Quaternionic wavelet coefficients modeling for a reduced-reference metric. *Signal Processing: Image Communication*, 36:127—139, 2015. https://doi.org/10.1016/j.image.2015.06.003.

[146] L. Traversoni. Quaternions on wavelets problems. In *Approximation Theory VIII*, volume 2: Wavelets and Multilevel Approximation of *Approximations and Decompositions vol. 6*, pages 391–397. World Scientific, Singapore, 1995. Not available online.

[147] L. Traversoni. Quaternion wavelets for moving volume representation. In *Proceedings of the Fifth International Conference on Information Visualisation (IV 2001), held in London, UK*, pages 459–463. IEEE, 2001. https://doi.org/10.1109/IV.2001.942096.

[148] L. Traversoni. Image analysis using quaternion wavelets. In *Geometric Algebra with Applications in Science and Engineering*, chapter 16, pages 326–345. Birkhäuser, Boston, MA, 2001. https://doi.org/10.1007/978-1-4612-0159-5_16.

[149] L. Traversoni and Y. Xu. Velocity and object detection using quaternion wavelets. In *Proceedings of the 2007 International Conference on Numerical Analysis and Applied Mathematics (ICNAAM 2007), held in Corfu, Greece*, volume 936 of *AIP Conference Proceedings*, pages 775–778. American Institute of Physics, 2007. https://doi.org/10.1063/1.2790268.

[150] A. K. Umam and M. Yunus. Quaternion wavelet transform for image denoising. *Journal of Physics: Conference Series*, 974(1):012006, 2018. https://doi.org/10.1088/

1742-6596/974/1/012006.

[151] J. Vaz Jr. and R. da Rocha Jr. *An Introduction to Clifford Algebras and Spinors*. Oxford University Press, 2016. https://doi.org/10.1093/acprof:oso/9780198782926.001.0001.

[152] A. T. Walden and A. Serroukh. Wavelet analysis of matrix-valued time-series. *Proceedings of the Royal Society A*, 458(2017):157—179, 2002. https://doi.org/10.1098/rspa.2001.0866.

[153] C. Wang, C. Zhao and J. Yang. Monocular odometry in country roads based on phase-derived optical flow and 4-DOF ego-motion model. *Industrial Robot: An International Journal*, 38(5):509–520, 2011. https://doi.org/10.1108/01439911111154081.

[154] C. Wang, C. Zhao, B. Fan and J. Ding. Hyper-complex wave phase coherent based image motion estimation. In *Proceedings of the 32nd Chinese Control Conference (CCC 2013) held in Xi'an*, pages 3645–3650. IEEE, 2013. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6640054&tag=1, accessed 19th Sept. 2018.

[155] C. Wang, X. Wang, C. Zhang and Z. Xia. Geometric correction based color image watermarking using fuzzy least squares support vector machine and Bessel K form distribution. *Signal Processing*, 134:197–208, 2017. https://doi.org/10.1016/j.sigpro.2016.12.010.

[156] J. Wang, T. Li, Y.-Q. Shi, S. Lian and J. Ye. Forensics feature analysis in quaternion wavelet domain for distinguishing photographic images and computer graphics. *Multimedia Tools and Applications*, 76(22):23721—23737, 2017. https://doi.org/10.1007/s11042-016-4153-0.

[157] T. Wilkinson. A cubic quincentenary. *Mathematics Today*, 54(3):102–105, 2018. https://drive.google.com/file/d/1talKSI5gAQ_7AZDxXmR0ge2M_E35EqaP/view, accessed 27th July 2018.

[158] S. Wu, Q. Zhu and Y. Xie. Evaluation of various speckle reduction filters on medical ultrasound images. In *Proceedings of the 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC 2013), held in Osaka, Japan*, pages 1148–1151. IEEE, 2013. https://doi.org/10.1109/EMBC.2013.6609709.

[159] X.-G. Xia and B. W. Suter. Vector-valued wavelets and vector filter banks. *IEEE Transactions on Signal Processing*, 44(3):508–518, 1996. https://doi.org/10.1109/78.489024.

[160] Y. Xu, J. Zhou and G. T. Zhai. 2-D phase-based matching in uncalibrated images. In

Proceedings of the 2005 IEEE Workshop on Signal Processing Systems (SiPS 2005), held in Athens, Greece, pages 325–330. IEEE, 2005. https://doi.org/10.1109/SIPS.2005.1579887.

[161] Y. Xu, X. Yang, P. Zhang, L. Song and L. Traversoni. Cooperative stereo matching using quaternion wavelets and top–down segmentation. In *Proceedings of the 2007 IEEE International Conference on Multimedia and Expo (ICME 2007), held in Beijing, China*, pages 1954–1957. IEEE, 2007. https://doi.org/10.1109/ICME.2007.4285060.

[162] M. Yin, W. Liu, J. Shui and J. Wu. Quaternion wavelet analysis and application in image denoising. *Mathematical Problems in Engineering*, 2012:493976, 2012. https://doi.org/10.1155/2012/493976.

[163] M. Yin, J. Wu, B. Chu, R. Kong and Z. Wang. Multi-focus image fusion based on quaternion wavelet. *Journal of Information and Computational Science*, 11(9):3187–3198, 2014. https://doi.org/10.12733/jics20103882.

[164] F. Yu, X. Liu and L. Sun. Image denoising based on quaternion wavelet Q-HMT model. In *Proceedings of the 2014 33rd Chinese Control Conference (CCC 2014), held in Nanjing, China*, pages 6279–6282. IEEE, 2014. https://doi.org/10.1109/ChiCC.2014.6896020.

[165] X. Zhang. Image denoising via modified multiple-step local wiener filter and quaternion wavelet transform. In Q. Liang, J. Mu, W. Wang and B. Zhang, editors, *Proceedings of the 4th International Conference on Communications, Signal Processing, and Systems (CSPS 2015), held in Chengdu, China*, volume 386 of *Lecture Notes in Electrical Engineering*, pages 733–739. Springer, 2016. https://doi.org/10.1007/978-3-662-49831-6_76.

[166] X.-P. Zhang, M. D. Desai and Y.-N. Peng. Orthogonal complex filter banks and wavelets: some properties and design. *IEEE Transactions on Signal Processing*, 47(4):1039–1048, 1999. https://doi.org/10.1109/78.752601.

[167] J. Zhao and L. Peng. Quaternion-valued admissible wavelets and orthogonal decomposition of $L^2(IG(2), \mathbb{H})$. *Frontiers of Mathematics in China*, 2(3):491–499, 2007. https://doi.org/10.1007/s11464-007-0030-5.

[168] X.-N. Zheng, X.-Q. Luo, Z.-C. Zhang and X.-J. Wu. Multi-focus image fusion using quaternion wavelet transform. In *Proceedings of the 23rd International Conference on Pattern Recognition (ICPR 2016), held in Cancún, Mexico*, pages 883–888. IEEE, 2016. https://doi.org/10.1109/ICPR.2016.7899747.

[169] J. Zhou, Y. Xu and X. Yang. Quaternion wavelet phase based stereo matching for

uncalibrated images. *Pattern Recognition Letters*, 28(12):1509–1522, 2007. https://doi.org/10.1016/j.patrec.2007.03.009.

Appendix A

MATLAB® for the Edge Detection Filter of Denis *et al.*

MATLAB® for Figure 2.2b:

```
% Code for the edge detector of Denis et al.

qlena = cast(imreadq('Lena.tiff'),'double');

mu = unit(quaternion(1,1,1));

R = exp(mu.*pi./2);

% Left and right horizontal masks

left = [quaternion(ones(1,3));quaternion(zeros(1,3));R R R];
right = conj(left);

% Left and right diagonal masks

diagleft = [quaternion(0) quaternion(1) quaternion(1);R...
    quaternion(0) quaternion(1); R R quaternion(0)];
diagright = conj(diagleft);

% Convolve the image with each pair of masks; "h" for
   horozontal,
% "v" for vertical, "l" for left, "r" for right

fqlenah = conv2({left,right},qlena);
```

```
fqlenav = conv2({left',right'},qlena);
fqlenadl = conv2({diagleft,diagright},qlena);
fqlenadr = conv2({diagleft',diagright'},qlena);

% Find the saturations

dqlenah = 0.5.*abs(fqlenah+mu.*fqlenah.*mu);
dqlenav = 0.5.*abs(fqlenav+mu.*fqlenav.*mu);
dqlenadl = 0.5.*abs(fqlenadl+mu.*fqlenadl.*mu);
dqlenadr = 0.5.*abs(fqlenadr+mu.*fqlenadr.*mu);

% Use maximum saturations for image

im1 = max(dqlenah,dqlenav);
im2 = max(im1,dqlenadl);
im = max(im2,dqlenadr);

% Display filtered image

image(im)

% remove tickmarks and axis labels

set(gca,'XTickLabel',[]);
set(gca,'YTickLabel',[]);
set(gca,'XTick',[]);
set(gca,'YTick',[]);
```

Appendix B

Example of Maple™ for Scaling Filters

An example of our Maple™ code, in this case for our length 10 quaternion scaling filter:

```
> restart;
> interface(rtablesize=100):
> Digits:=16:
> with(LinearAlgebra):
################################################################################
>
Real 4×4 matrix representations of the quaternion s, i, j and k
>
################################################################################
> Is:=<1,0,0,0;0,1,0,0;0,0,1,0;0,0,0,1>:
> Ii:=<0,-1,0,0;1,0,0,0;0,0,0,-1;0,0,1,0>:
> Ij:=<0,0,-1,0;0,0,0,1;1,0,0,0;0,-1,0,0>:
> Ik:=<0,0,0,-1;0,0,-1,0;0,1,0,0;1,0,0,0>:
################################################################################
>
The ten quaternion scaling filter coefficients we are after, each represented as a 4 × 4 matrix
>
We assume symmetry and also that the coefficients of j and k in G[0] and the coefficient of j in
G[1] are zero
>
################################################################################
> G[0]:=A0*Is+B0*Ii:
> G[1]:=A1*Is+B1*Ii+D1*Ik:
> G[2]:=A2*Is+B2*Ii+C2*Ij+D2*Ik:
> G[3]:=A3*Is+B3*Ii+C3*Ij+D3*Ik:
> G[4]:=A4*Is+B4*Ii+C4*Ij+D4*Ik:
```

```
> G[5]:=G[4]:
> G[6]:=G[3]:
> G[7]:=G[2]:
> G[8]:=G[1]:
> G[9]:=G[0]:
```
##
>
Ginzberg's equations for a length 10 filter
>
##
```
> eqn1:=sum(G[k],k=0..9)-sqrt(2)*Is:
> eqn2:=sum((-1)^k*G[k],k=0..9):
> eqn3:=sum((-1)^k*k*G[k],k=1..9):
> eqn4:=sum((-1)^k*k^2*G[k],k=1..9):
> eqn5:=sum((-1)^k*k^3*G[k],k=1..9):
> eqn6:=sum((-1)^k*k^4*G[k],k=1..9):
> eqn7:=G[0].Transpose(G[2])+G[1].Transpose(G[3])
+G[2].Transpose(G[4])+G[3].Transpose(G[5])+G[4].Transpose(G[6])
+G[5].Transpose(G[7])+G[6].Transpose(G[8])+G[7].Transpose(G[9]):
> eqn8:=G[0].Transpose(G[4])+G[1].Transpose(G[5])
+G[2].Transpose(G[6])+G[3].Transpose(G[7])+G[4].Transpose(G[8])
+G[5].Transpose(G[9]):
> eqn9:=G[0].Transpose(G[6])+G[1].Transpose(G[7])
+G[2].Transpose(G[8])+G[3].Transpose(G[9]):
> eqn10:=G[0].Transpose(G[8])+G[1].Transpose(G[9]):
> eqn11:=G[0].Transpose(G[0])+G[1].Transpose(G[1])
+G[2].Transpose(G[2])+G[3].Transpose(G[3])+G[4].Transpose(G[4])
+G[5].Transpose(G[5])+G[6].Transpose(G[6])+G[7].Transpose(G[7])
+G[8].Transpose(G[8])+G[9].Transpose(G[9])-Is:
```
##
>
We solve the linear equations first: because of the symmetry, some of the equations are identically zero
>
##
```
> A0:=solve(eqn1[1,1]=0,A0);
```
$$A0 := -A1 - A2 - A3 - A4 + \frac{1}{2}\sqrt{2}$$
```
> B0:=solve(eqn1[2,1]=0,B0);
```
$$B0 := -B1 - B2 - B3 - B4$$
```
> C4:=solve(eqn1[3,1]=0,C4);
```
$$C4 := -C2 - C3$$
```
> D4:=solve(eqn1[4,1]=0,D4);
```
$$D4 := -D1 - D2 - D3$$
```
> eqn2[1..4,1];
```

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

```
> A1:=solve(eqn3[1,1]=0,A1);
```

$$A1 := -\frac{1}{4} A2 - \frac{3}{4} A3 - \frac{1}{2} A4 + \frac{9}{32}\sqrt{2}$$

```
> B1:=solve(eqn3[2,1]=0,B1);
```

$$B1 := -\frac{1}{4} B2 - \frac{3}{4} B3 - \frac{1}{2} B4$$

```
> C3:=solve(eqn3[3,1]=0,C3);
```

$$C3 := C2$$

```
> D1:=solve(eqn3[4,1]=0,D1);
```

$$D1 := \frac{1}{2} D2 - \frac{1}{2} D3$$

```
> eqn4[1..4,1];
```

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

```
> A2:=solve(eqn5[1,1]=0,A2);
```

$$A2 := \frac{1}{7} A3 - \frac{4}{7} A4 + \frac{3}{16}\sqrt{2}$$

```
> B2:=solve(eqn5[2,1]=0,B2);
```

$$B2 := \frac{1}{7} B3 - \frac{4}{7} B4$$

```
> C2:=solve(eqn5[3,1]=0,C2);
```

$$C2 := 0$$

```
> D2:=solve(eqn5[4,1]=0,D2);
```

$$D2 := 3\,D3$$

```
> eqn6[1..4,1];
```

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

```
> f1:=expand(eqn7[1,1]);
```

$$f1 := -\frac{82}{49} A3^2 + \frac{96}{49} A3\,A4 + \frac{5}{14} A3\sqrt{2} - \frac{52}{49} A4^2 + \frac{29}{112} A4\sqrt{2} + \frac{15}{256} - \frac{82}{49} B3^2 + \frac{96}{49} B3\,B4 - \frac{52}{49} B4^2 - 38\,D3^2$$

```
> eqn7[2..4,1];
```

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

```
> f2:=expand(eqn8[1,1]);
```

$$f2 := -\frac{24}{7} A3\,A4 - \frac{6}{7} A4^2 + \frac{5}{8} A4\sqrt{2} + \frac{2}{7} A3^2 + \frac{3}{8} A3\sqrt{2} - \frac{24}{7} B3\,B4 - \frac{6}{7} B4^2 + \frac{2}{7} B3^2 - 4\,D3^2$$

```
> eqn8[2..4,1];
```

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

```
> f3:=expand(eqn9[1,1]);
```

$$f3 := -\frac{46}{49} A3^2 + \frac{32}{49} A3\,A4 - \frac{1}{14} A3\sqrt{2} + \frac{20}{49} A4^2 - \frac{45}{112} A4\sqrt{2} + \frac{45}{256} - \frac{46}{49} B3^2 + \frac{32}{49} B3\,B4 + \frac{20}{49} B4^2 + 6\,D3^2$$

```
> eqn9[2..4,1];
```

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

```
> f4:=expand(eqn10[1,1]);
```

$$f4 := \frac{55}{98} A3^2 + \frac{18}{49} A3\,A4 - \frac{65}{224} A3\sqrt{2} + \frac{5}{98} A4^2 - \frac{5}{56} A4\sqrt{2} + \frac{75}{1024} + \frac{55}{98} B3^2 + \frac{18}{49} B3\,B4 + \frac{5}{98} B4^2$$

```
> eqn10[2..4,1];
```

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

```
> f5:=expand(eqn11[1,1]);
```

$$f5 := -\frac{315}{512} + \frac{44}{49} A3\,A4 + \frac{44}{49} B3\,B4 + 72\,D3^2 + \frac{173}{49} A3^2 - \frac{83}{112} A3\sqrt{2} + \frac{143}{49} A4^2 - \frac{11}{14} A4\sqrt{2} + \frac{173}{49} B3^2 + \frac{143}{49} B4^2$$

##
>
To solve the nonlinear equations, we simplify the task by finding the Gröbner basis
>
##

```
> F:=[f1,f2,f3,f4,f5]:
> J:=Groebner:-Basis(F,plex):
> J[1];
```

$$768\,A3 + 512\,A4 - 315\sqrt{2}$$

```
> A3:=solve(J[1]=0,A3);
```

$$A3 := -\frac{2}{3}\,A4 + \frac{105}{256}\sqrt{2}$$

```
> normal(J[5]);
```

$$\frac{131072}{3}\,B3\,A4^2 - \frac{94720}{3}\,B3\,A4\sqrt{2} + 9660\,B3 + \frac{262144}{9}\,B4\,A4^2 - 23040\,B4\,A4\sqrt{2} \\ + 8400\,B4$$

```
> B3:=solve(J[5]=0,B3);
```

$$B3 := -\frac{4}{3}\,\frac{B4\left(16384\,A4^2 - 12960\,A4\sqrt{2} + 4725\right)}{32768\,A4^2 - 23680\,A4\sqrt{2} + 7245}$$

```
> normal(J[2]);
```

$$\frac{28}{9}\,\frac{1}{\left(32768\,A4^2 - 23680\,A4\sqrt{2} + 7245\right)^2}\left(17592186044416\,A4^6\right. \\ - 39341900431360\,A4^5\sqrt{2} + 17592186044416\,A4^4\,B4^2 \\ - 28346784153600\,A4^3\,B4^2\sqrt{2} + 71452149678080\,A4^4 - 33642381312000\,A4^3\sqrt{2} \\ + 33484638781440\,A4^2\,B4^2 - 8576276889600\,A4\,B4^2\sqrt{2} + 17296234905600\,A4^2 \\ \left. - 2301527844000\,A4\sqrt{2} + 1609445376000\,B4^2 + 248015368125\right)$$

```
> solve(32768*A4^2-23680*A4*sqrt(2)+7245=0,A4);
```

$$\frac{185}{512}\sqrt{2} + \frac{1}{512}\sqrt{10490},\ \frac{185}{512}\sqrt{2} - \frac{1}{512}\sqrt{10490}$$

```
> A4:=(185/512)*sqrt(2);
```

$$A4 := \frac{185}{512}\sqrt{2}$$

```
> normal(J[3]);
```

$$\frac{313145}{9} - \frac{3111176241152}{9903609}\,B4^2$$

```
> s1:=solve(J[3]=0,B4);
```

$$s1 := -\frac{1}{194048}\sqrt{4170519790},\ \frac{1}{194048}\sqrt{4170519790}$$

##
>
Case 1: negative B4 and positive D3
>
##

```
> B4:=s1[1];
```

$$B4 := -\frac{1}{194048}\sqrt{4170519790}$$

```
> normal(J[4]);
```

$$-\frac{2590}{1137}+8192\,D3^2$$

```
> s2:=solve(J[4]=0,D3);
```

$$s2 := \frac{1}{72768}\sqrt{1472415},\ -\frac{1}{72768}\sqrt{1472415}$$

```
> D3:=s2[1];
```

$$D3 := \frac{1}{72768}\sqrt{1472415}$$

```
> eqn1[1..4,1],eqn2[1..4,1],eqn3[1..4,1],eqn4[1..4,1],
eqn5[1..4,1],eqn6[1..4,1],eqn7[1..4,1],eqn8[1..4,1],
eqn9[1..4,1],eqn10[1..4,1],eqn11[1..4,1];
```

$$\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\end{bmatrix}$$

```
> G[0][1..4,1],G[1][1..4,1],G[2][1..4,1],G[3][1..4,1],
G[4][1..4,1],G[5][1..4,1],G[6][1..4,1],G[7][1..4,1],
G[8][1..4,1],G[9][1..4,1];
```

$$\left[\begin{array}{c} -\frac{25}{3072}\sqrt{2} \\ -\frac{1}{3222528}\sqrt{4170519790} \\ 0 \\ 0 \end{array}\right], \left[\begin{array}{c} -\frac{85}{3072}\sqrt{2} \\ \frac{425}{1221338112}\sqrt{4170519790} \\ 0 \\ \frac{1}{72768}\sqrt{1472415} \end{array}\right],$$

$$\left[\begin{array}{c} \frac{1}{192}\sqrt{2} \\ \frac{491}{152667264}\sqrt{4170519790} \\ 0 \\ \frac{1}{24256}\sqrt{1472415} \end{array}\right], \left[\begin{array}{c} \frac{65}{384}\sqrt{2} \\ \frac{145}{76333632}\sqrt{4170519790} \\ 0 \\ \frac{1}{72768}\sqrt{1472415} \end{array}\right],$$

$$\left[\begin{array}{c} \frac{185}{512}\sqrt{2} \\ -\frac{1}{194048}\sqrt{4170519790} \\ 0 \\ -\frac{5}{72768}\sqrt{1472415} \end{array}\right], \left[\begin{array}{c} \frac{185}{512}\sqrt{2} \\ -\frac{1}{194048}\sqrt{4170519790} \\ 0 \\ -\frac{5}{72768}\sqrt{1472415} \end{array}\right],$$

$$\left[\begin{array}{c} \frac{65}{384}\sqrt{2} \\ \frac{145}{76333632}\sqrt{4170519790} \\ 0 \\ \frac{1}{72768}\sqrt{1472415} \end{array}\right], \left[\begin{array}{c} \frac{1}{192}\sqrt{2} \\ \frac{491}{152667264}\sqrt{4170519790} \\ 0 \\ \frac{1}{24256}\sqrt{1472415} \end{array}\right],$$

$$\left[\begin{array}{c} -\frac{85}{3072}\sqrt{2} \\ \frac{425}{1221338112}\sqrt{4170519790} \\ 0 \\ \frac{1}{72768}\sqrt{1472415} \end{array}\right], \left[\begin{array}{c} -\frac{25}{3072}\sqrt{2} \\ -\frac{1}{3222528}\sqrt{4170519790} \\ 0 \\ 0 \end{array}\right]$$

```
> map(evalf,G[0][1..4,1]),map(evalf,G[1][1..4,1]),
map(evalf,G[2][1..4,1]),map(evalf,G[3][1..4,1]),
map(evalf,G[4][1..4,1]),map(evalf,G[5][1..4,1]),
map(evalf,G[6][1..4,1]),map(evalf,G[7][1..4,1]),
map(evalf,G[8][1..4,1]),map(evalf,G[9][1..4,1]);
```

$$\begin{bmatrix} -0.01150889943337480 \\ -0.02004003121304868 \\ 0. \\ 0. \end{bmatrix}, \begin{bmatrix} -0.03913025807347430 \\ 0.02247233051595169 \\ 0. \\ 0.01667533923985632 \end{bmatrix}, \begin{bmatrix} 0.007365695637359869 \\ 0.2076972100391958 \\ 0. \\ 0.05002601771956897 \end{bmatrix},$$

$$\begin{bmatrix} 0.2393851082141957 \\ 0.1226724865811951 \\ 0. \\ 0.01667533923985632 \end{bmatrix}, \begin{bmatrix} 0.5109951348418410 \\ -0.3328019959232939 \\ 0. \\ -0.08337669619928162 \end{bmatrix}, \begin{bmatrix} 0.5109951348418410 \\ -0.3328019959232939 \\ 0. \\ -0.08337669619928162 \end{bmatrix},$$

$$\begin{bmatrix} 0.2393851082141957 \\ 0.1226724865811951 \\ 0. \\ 0.01667533923985632 \end{bmatrix}, \begin{bmatrix} 0.007365695637359869 \\ 0.2076972100391958 \\ 0. \\ 0.05002601771956897 \end{bmatrix}, \begin{bmatrix} -0.03913025807347430 \\ 0.02247233051595169 \\ 0. \\ 0.01667533923985632 \end{bmatrix},$$

$$\begin{bmatrix} -0.01150889943337480 \\ -0.02004003121304868 \\ 0. \\ 0. \end{bmatrix}$$

```
######################################################################################
>
```
Case 2: negative B4 and negative D3
```
>
######################################################################################
> B4:=s1[1];
```

$$B4 := -\frac{1}{194048}\sqrt{4170519790}$$

```
> D3:=s2[2];
```

$$D3 := -\frac{1}{72768}\sqrt{1472415}$$

```
> eqn1[1..4,1],eqn2[1..4,1],eqn3[1..4,1],eqn4[1..4,1],
eqn5[1..4,1],eqn6[1..4,1],eqn7[1..4,1],eqn8[1..4,1],
eqn9[1..4,1],eqn10[1..4,1],eqn11[1..4,1];
```

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

```
> G[0][1..4,1],G[1][1..4,1],G[2][1..4,1],G[3][1..4,1],
G[4][1..4,1],G[5][1..4,1],G[6][1..4,1],G[7][1..4,1],
G[8][1..4,1],G[9][1..4,1];
```

$$\begin{bmatrix} -\frac{25}{3072}\sqrt{2} \\ -\frac{1}{3222528}\sqrt{4170519790} \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -\frac{85}{3072}\sqrt{2} \\ \frac{425}{1221338112}\sqrt{4170519790} \\ 0 \\ -\frac{1}{72768}\sqrt{1472415} \end{bmatrix},$$

$$\begin{bmatrix} \frac{1}{192}\sqrt{2} \\ \frac{491}{152667264}\sqrt{4170519790} \\ 0 \\ -\frac{1}{24256}\sqrt{1472415} \end{bmatrix}, \begin{bmatrix} \frac{65}{384}\sqrt{2} \\ \frac{145}{76333632}\sqrt{4170519790} \\ 0 \\ -\frac{1}{72768}\sqrt{1472415} \end{bmatrix},$$

$$\begin{bmatrix} \frac{185}{512}\sqrt{2} \\ -\frac{1}{194048}\sqrt{4170519790} \\ 0 \\ \frac{5}{72768}\sqrt{1472415} \end{bmatrix}, \begin{bmatrix} \frac{185}{512}\sqrt{2} \\ -\frac{1}{194048}\sqrt{4170519790} \\ 0 \\ \frac{5}{72768}\sqrt{1472415} \end{bmatrix},$$

$$\begin{bmatrix} \frac{65}{384}\sqrt{2} \\ \frac{145}{76333632}\sqrt{4170519790} \\ 0 \\ -\frac{1}{72768}\sqrt{1472415} \end{bmatrix}, \begin{bmatrix} \frac{1}{192}\sqrt{2} \\ \frac{491}{152667264}\sqrt{4170519790} \\ 0 \\ -\frac{1}{24256}\sqrt{1472415} \end{bmatrix},$$

$$\begin{bmatrix} -\frac{85}{3072}\sqrt{2} \\ \frac{425}{1221338112}\sqrt{4170519790} \\ 0 \\ -\frac{1}{72768}\sqrt{1472415} \end{bmatrix}, \begin{bmatrix} -\frac{25}{3072}\sqrt{2} \\ -\frac{1}{3222528}\sqrt{4170519790} \\ 0 \\ 0 \end{bmatrix}$$

```
> map(evalf,G[0][1..4,1]),map(evalf,G[1][1..4,1]),
map(evalf,G[2][1..4,1]),map(evalf,G[3][1..4,1]),
map(evalf,G[4][1..4,1]),map(evalf,G[5][1..4,1]),
map(evalf,G[6][1..4,1]),map(evalf,G[7][1..4,1]),
map(evalf,G[8][1..4,1]),map(evalf,G[9][1..4,1]);
```

$$\begin{bmatrix}-0.01150889943337480\\-0.02004003121304868\\0.\\0.\end{bmatrix},\begin{bmatrix}-0.03913025807347430\\0.02247233051595169\\0.\\-0.01667533923985632\end{bmatrix},\begin{bmatrix}0.007365695637359869\\0.2076972100391958\\0.\\-0.05002601771956897\end{bmatrix},$$

$$\begin{bmatrix}0.2393851082141957\\0.1226724865811951\\0.\\-0.01667533923985632\end{bmatrix},\begin{bmatrix}0.5109951348418410\\-0.3328019959232939\\0.\\0.08337669619928162\end{bmatrix},\begin{bmatrix}0.5109951348418410\\-0.3328019959232939\\0.\\0.08337669619928162\end{bmatrix},$$

$$\begin{bmatrix}0.2393851082141957\\0.1226724865811951\\0.\\-0.01667533923985632\end{bmatrix},\begin{bmatrix}0.007365695637359869\\0.2076972100391958\\0.\\-0.05002601771956897\end{bmatrix},\begin{bmatrix}-0.03913025807347430\\0.02247233051595169\\0.\\-0.01667533923985632\end{bmatrix},$$

$$\begin{bmatrix}-0.01150889943337480\\-0.02004003121304868\\0.\\0.\end{bmatrix}$$

##
>
Case 3: positive B4 and positive D3
>
##

```
> B4:=s1[2];
```

$$B4 := \frac{1}{194048}\sqrt{4170519790}$$

```
> D3:=s2[1];
```

$$D3 := \frac{1}{72768}\sqrt{1472415}$$

```
> eqn1[1..4,1],eqn2[1..4,1],eqn3[1..4,1],eqn4[1..4,1],
eqn5[1..4,1],eqn6[1..4,1],eqn7[1..4,1],eqn8[1..4,1],
eqn9[1..4,1],eqn10[1..4,1],eqn11[1..4,1];
```

$$\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\0\\0\end{bmatrix}$$

```
> G[0][1..4,1],G[1][1..4,1],G[2][1..4,1],G[3][1..4,1],
G[4][1..4,1],G[5][1..4,1],G[6][1..4,1],G[7][1..4,1],
G[8][1..4,1],G[9][1..4,1];
```

$$\begin{bmatrix} -\frac{25}{3072}\sqrt{2} \\ \frac{1}{3222528}\sqrt{4170519790} \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -\frac{85}{3072}\sqrt{2} \\ -\frac{425}{1221338112}\sqrt{4170519790} \\ 0 \\ \frac{1}{72768}\sqrt{1472415} \end{bmatrix},$$

$$\begin{bmatrix} \frac{1}{192}\sqrt{2} \\ -\frac{491}{152667264}\sqrt{4170519790} \\ 0 \\ \frac{1}{24256}\sqrt{1472415} \end{bmatrix}, \begin{bmatrix} \frac{65}{384}\sqrt{2} \\ -\frac{145}{76333632}\sqrt{4170519790} \\ 0 \\ \frac{1}{72768}\sqrt{1472415} \end{bmatrix},$$

$$\begin{bmatrix} \frac{185}{512}\sqrt{2} \\ \frac{1}{194048}\sqrt{4170519790} \\ 0 \\ -\frac{5}{72768}\sqrt{1472415} \end{bmatrix}, \begin{bmatrix} \frac{185}{512}\sqrt{2} \\ \frac{1}{194048}\sqrt{4170519790} \\ 0 \\ -\frac{5}{72768}\sqrt{1472415} \end{bmatrix},$$

$$\begin{bmatrix} \frac{65}{384}\sqrt{2} \\ -\frac{145}{76333632}\sqrt{4170519790} \\ 0 \\ \frac{1}{72768}\sqrt{1472415} \end{bmatrix}, \begin{bmatrix} \frac{1}{192}\sqrt{2} \\ -\frac{491}{152667264}\sqrt{4170519790} \\ 0 \\ \frac{1}{24256}\sqrt{1472415} \end{bmatrix},$$

$$\begin{bmatrix} -\frac{85}{3072}\sqrt{2} \\ -\frac{425}{1221338112}\sqrt{4170519790} \\ 0 \\ \frac{1}{72768}\sqrt{1472415} \end{bmatrix}, \begin{bmatrix} -\frac{25}{3072}\sqrt{2} \\ \frac{1}{3222528}\sqrt{4170519790} \\ 0 \\ 0 \end{bmatrix}$$

```
> map(evalf,G[0][1..4,1]),map(evalf,G[1][1..4,1]),
map(evalf,G[2][1..4,1]),map(evalf,G[3][1..4,1]),
map(evalf,G[4][1..4,1]),map(evalf,G[5][1..4,1]),
map(evalf,G[6][1..4,1]),map(evalf,G[7][1..4,1]),
map(evalf,G[8][1..4,1]),map(evalf,G[9][1..4,1]);
```

$$\begin{bmatrix} -0.01150889943337480 \\ 0.02004003121304868 \\ 0. \\ 0. \end{bmatrix}, \begin{bmatrix} -0.03913025807347430 \\ -0.02247233051595169 \\ 0. \\ 0.01667533923985632 \end{bmatrix}, \begin{bmatrix} 0.007365695637359869 \\ -0.2076972100391958 \\ 0. \\ 0.05002601771956897 \end{bmatrix},$$

$$\begin{bmatrix} 0.2393851082141957 \\ -0.1226724865811951 \\ 0. \\ 0.01667533923985632 \end{bmatrix}, \begin{bmatrix} 0.5109951348418410 \\ 0.3328019959232939 \\ 0. \\ -0.08337669619928162 \end{bmatrix}, \begin{bmatrix} 0.5109951348418410 \\ 0.3328019959232939 \\ 0. \\ -0.08337669619928162 \end{bmatrix},$$

$$\begin{bmatrix} 0.2393851082141957 \\ -0.1226724865811951 \\ 0. \\ 0.01667533923985632 \end{bmatrix}, \begin{bmatrix} 0.007365695637359869 \\ -0.2076972100391958 \\ 0. \\ 0.05002601771956897 \end{bmatrix}, \begin{bmatrix} -0.03913025807347430 \\ -0.02247233051595169 \\ 0. \\ 0.01667533923985632 \end{bmatrix},$$

$$\begin{bmatrix} -0.01150889943337480 \\ 0.02004003121304868 \\ 0. \\ 0. \end{bmatrix}$$

```
####################################################################################
>
```

Case 4: positive B4 and negative D3

```
>
####################################################################################
> B4:=s1[2];
```

$$B4 := \frac{1}{194048}\sqrt{4170519790}$$

```
> D3:=s2[2];
```

$$D3 := -\frac{1}{72768}\sqrt{1472415}$$

```
> eqn1[1..4,1],eqn2[1..4,1],eqn3[1..4,1],eqn4[1..4,1],
eqn5[1..4,1],eqn6[1..4,1],eqn7[1..4,1],eqn8[1..4,1],
eqn9[1..4,1],eqn10[1..4,1],eqn11[1..4,1];
```

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

```
> G[0][1..4,1],G[1][1..4,1],G[2][1..4,1],G[3][1..4,1],
G[4][1..4,1],G[5][1..4,1],G[6][1..4,1],G[7][1..4,1],
G[8][1..4,1],G[9][1..4,1];
```

$$\begin{bmatrix} -\frac{25}{3072}\sqrt{2} \\ \frac{1}{3222528}\sqrt{4170519790} \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -\frac{85}{3072}\sqrt{2} \\ -\frac{425}{1221338112}\sqrt{4170519790} \\ 0 \\ -\frac{1}{72768}\sqrt{1472415} \end{bmatrix},$$

$$\begin{bmatrix} \frac{1}{192}\sqrt{2} \\ -\frac{491}{152667264}\sqrt{4170519790} \\ 0 \\ -\frac{1}{24256}\sqrt{1472415} \end{bmatrix}, \begin{bmatrix} \frac{65}{384}\sqrt{2} \\ -\frac{145}{76333632}\sqrt{4170519790} \\ 0 \\ -\frac{1}{72768}\sqrt{1472415} \end{bmatrix},$$

$$\begin{bmatrix} \frac{185}{512}\sqrt{2} \\ \frac{1}{194048}\sqrt{4170519790} \\ 0 \\ \frac{5}{72768}\sqrt{1472415} \end{bmatrix}, \begin{bmatrix} \frac{185}{512}\sqrt{2} \\ \frac{1}{194048}\sqrt{4170519790} \\ 0 \\ \frac{5}{72768}\sqrt{1472415} \end{bmatrix},$$

$$\begin{bmatrix} \frac{65}{384}\sqrt{2} \\ -\frac{145}{76333632}\sqrt{4170519790} \\ 0 \\ -\frac{1}{72768}\sqrt{1472415} \end{bmatrix}, \begin{bmatrix} \frac{1}{192}\sqrt{2} \\ -\frac{491}{152667264}\sqrt{4170519790} \\ 0 \\ -\frac{1}{24256}\sqrt{1472415} \end{bmatrix},$$

$$\begin{bmatrix} -\frac{85}{3072}\sqrt{2} \\ -\frac{425}{1221338112}\sqrt{4170519790} \\ 0 \\ -\frac{1}{72768}\sqrt{1472415} \end{bmatrix}, \begin{bmatrix} -\frac{25}{3072}\sqrt{2} \\ \frac{1}{3222528}\sqrt{4170519790} \\ 0 \\ 0 \end{bmatrix}$$

```
> map(evalf,G[0][1..4,1]),map(evalf,G[1][1..4,1]),
map(evalf,G[2][1..4,1]),map(evalf,G[3][1..4,1]),
map(evalf,G[4][1..4,1]),map(evalf,G[5][1..4,1]),
map(evalf,G[6][1..4,1]),map(evalf,G[7][1..4,1]),
map(evalf,G[8][1..4,1]),map(evalf,G[9][1..4,1]);
```

$$\begin{bmatrix} -0.01150889943337480 \\ 0.02004003121304868 \\ 0. \\ 0. \end{bmatrix}, \begin{bmatrix} -0.03913025807347430 \\ -0.02247233051595169 \\ 0. \\ -0.01667533923985632 \end{bmatrix}, \begin{bmatrix} 0.007365695637359869 \\ -0.2076972100391958 \\ 0. \\ -0.05002601771956897 \end{bmatrix},$$

$$\begin{bmatrix} 0.2393851082141957 \\ -0.1226724865811951 \\ 0. \\ -0.01667533923985632 \end{bmatrix}, \begin{bmatrix} 0.5109951348418410 \\ 0.3328019959232939 \\ 0. \\ 0.08337669619928162 \end{bmatrix}, \begin{bmatrix} 0.5109951348418410 \\ 0.3328019959232939 \\ 0. \\ 0.08337669619928162 \end{bmatrix},$$

$$\begin{bmatrix} 0.2393851082141957 \\ -0.1226724865811951 \\ 0. \\ -0.01667533923985632 \end{bmatrix}, \begin{bmatrix} 0.007365695637359869 \\ -0.2076972100391958 \\ 0. \\ -0.05002601771956897 \end{bmatrix}, \begin{bmatrix} -0.03913025807347430 \\ -0.02247233051595169 \\ 0. \\ -0.01667533923985632 \end{bmatrix},$$

$$\begin{bmatrix} -0.01150889943337480 \\ 0.02004003121304868 \\ 0. \\ 0. \end{bmatrix}$$

```
>
```

Appendix C

Example of our Genetic Algorithm in MATLAB® for Scaling Filters

An example of our genetic algorithm code in MATLAB® code, in this case for our length 10 Clifford $Cl(1,1)$ scaling filter:

```
close all
clear all

% Choose seed for random number generator

rng(2171201)

% Random start for seven unknowns, which may lie between
% -1 and +1

x = 2.*rand(1,7)-1;

% The fitness function of the seven unknowns that we want to
% minimise

fit = @(x) Cl_1_1_fit10(x);

% Options for the ga function

options = gaoptimset('PlotFcns', @gaplotbestf,'generations',...
    1000,'crossoverFcn', {@crossoverheuristic, 1.3},...
    'InitialPopulation',x,'InitialScores',[fit(x)],...
    'MutationFcn',{@mutationadaptfeasible,0.01,0.6},...
    'populationsize',50000,'elitecount',10,'tolfun',1e-60,...
    'UseParallel',true);

% The actual ga function

[r,fval,exitflag,output,population,scores] = ga(fit,7,...
    [],[],[],[],-1,1,[],options);

% Termination condition

if exitflag == 1
     disp('Terminated due to tolerance?')
end

% The 'best' values

best = find(scores==min(scores))
x = population(best(1),:)
fit(x)

% Now put x back into the equations from Maple and check what
% Ginzberg's equations evaluate to

% Assign x to unknowns from Maple

A4 = x(1);
B3 = x(2);
B4 = x(3);
C3 = x(4);
C4 = x(5);
D3 = x(6);
D4 = x(7);

% The first nonlinear equation, which we could not do anything
   with

A3 = -(1/5376*(1792*A4*sqrt(2)+73728*B3^2-196608*B3*B4...
    +131072*B4^2+73728*D3^2...
    -196608*D3*D4+131072*D4^2-2205))*sqrt(2);

% The solutions of the linear equations from Maple, eliminating
% one unknown at a time by expressing it in terms of other
% unknowns

D2 = (15/7)*D3-(20/7)*D4;
```

```
B2 = (15/7)*B3-(20/7)*B4;
C2 = (1/7)*C3-(4/7)*C4;
A2 = (1/7)*A3-(4/7)*A4+(3/16)*sqrt(2);
D1 = (7/4)*D2-(9/4)*D3+(5/2)*D4;
B1 = (7/4)*B2-(9/4)*B3+(5/2)*B4;
C1 = -(1/4)*C2-(3/4)*C3-(1/2)*C4;
A1 = -(1/4)*A2-(3/4)*A3-(1/2)*A4+(9/32)*sqrt(2);
D0 = D1-D2+D3-D4;
B0 = B1-B2+B3-B4;
C0 = -C1-C2-C3-C4;
A0 = -A1-A2-A3-A4+(1/2)*sqrt(2);

% Matrix basis elements

m0 = [1,0,0,0;0,1,0,0;0,0,1,0;0,0,0,1];
m1 = [0,1,0,0;1,0,0,0;0,0,0,1;0,0,1,0];
m2 = [0,0,-1,0;0,0,0,1;1,0,0,0;0,-1,0,0];
m12 = m1*m2;

% Matrices to use in Ginzberg's equations

G0 = A0*m0+B0*m1+C0*m2+D0*m12;
G1 = A1*m0+B1*m1+C1*m2+D1*m12;
G2 = A2*m0+B2*m1+C2*m2+D2*m12;
G3 = A3*m0+B3*m1+C3*m2+D3*m12;
G4 = A4*m0+B4*m1+C4*m2+D4*m12;
G5 = A4*m0-B4*m1+C4*m2-D4*m12;
G6 = A3*m0-B3*m1+C3*m2-D3*m12;
G7 = A2*m0-B2*m1+C2*m2-D2*m12;
G8 = A1*m0-B1*m1+C1*m2-D1*m12;
G9 = A0*m0-B0*m1+C0*m2-D0*m12;

% Ginzberg's equations

eqn1 = G0+G1+G2+G3+G4+G5+G6+G7+G8+G9-sqrt(2)*m0;
eqn2 = G0-G1+G2-G3+G4-G5+G6-G7+G8-G9;
eqn3 = -G1+2*G2-3*G3+4*G4-5*G5+6*G6-7*G7+8*G8-9*G9;
eqn4 = -G1+4*G2-9*G3+16*G4-25*G5+36*G6-49*G7+64*G8-81*G9;
eqn5 = -G1+8*G2-27*G3+64*G4-125*G5+216*G6-343*G7+512*G8-729*G9;
eqn6 = -G1+16*G2-81*G3+256*G4-625*G5+1296*G6-2401*G7+4096*G8...
    -6561*G9;
eqn7 = G0*G2'+G1*G3'+G2*G4'+G3*G5'+G4*G6'+G5*G7'+G6*G8'+G7*G9';
eqn8 = G0*G4'+G1*G5'+G2*G6'+G3*G7'+G4*G8'+G5*G9';
eqn9 = G0*G6'+G1*G7'+G2*G8'+G3*G9';
```

```
eqn10 = G0*G8'+G1*G9';
eqn11 = G0*G0'+G1*G1'+G2*G2'+G3*G3'+G4*G4'+G5*G5'+G6*G6'...
   +G7*G7'+G8*G8'+G9*G9'-m0;

% Now display the values in Ginzberg's equations: they should
% all be 'small' (theoretically zero if we had the exact roots)

eqn1(1:4,1),eqn2(1:4,1),eqn3(1:4,1),eqn4(1:4,1),eqn5(1:4,1),...
   eqn6(1:4,1),eqn7(1:4,1),eqn8(1:4,1),eqn9(1:4,1),...
   eqn10(1:4,1),eqn11(1:4,1)
```

The fitness function whose value we want to minimise:

```
function y = Cl_1_1_fit10(x)

% Assign x to unknowns from Maple

A4 = x(1);
B3 = x(2);
B4 = x(3);
C3 = x(4);
C4 = x(5);
D3 = x(6);
D4 = x(7);

% The first nonlinear equation, which we could not do anything
% with

A3 = -(1/5376*(1792*A4*sqrt(2)+73728*B3^2-196608*B3*B4...
   +131072*B4^2+73728*D3^2...
   -196608*D3*D4+131072*D4^2-2205))*sqrt(2);

% The solutions of the linear equations from Maple, eliminating
% one unknown at a time by expressing it in terms of other
% unknowns

D2 = (15/7)*D3-(20/7)*D4;
B2 = (15/7)*B3-(20/7)*B4;
C2 = (1/7)*C3-(4/7)*C4;
A2 = (1/7)*A3-(4/7)*A4+(3/16)*sqrt(2);
D1 = (7/4)*D2-(9/4)*D3+(5/2)*D4;
B1 = (7/4)*B2-(9/4)*B3+(5/2)*B4;
C1 = -(1/4)*C2-(3/4)*C3-(1/2)*C4;
```

```
A1 = -(1/4)*A2-(3/4)*A3-(1/2)*A4+(9/32)*sqrt(2);
D0 = D1-D2+D3-D4;
B0 = B1-B2+B3-B4;
C0 = -C1-C2-C3-C4;
A0 = -A1-A2-A3-A4+(1/2)*sqrt(2);

% Equations that we want to be zero, from the first columns of
% equations eqn7 to eqn11

f1 = -(82/49)*C3^2-(52/49)*C4^2-(82/49)*A3^2+(222/49)*B3^2...
    +(222/49)*D3^2-(52/49)*A4^2-(100/49)*B4^2-(100/49)*D4^2...
    +(5/14)*A3*sqrt(2)+(29/112)*A4*sqrt(2)+15/256...
    +(96/49)*A3*A4-(368/49)*B3*B4+(96/49)*C3*C4-(368/49)*D3*D4;
f2 = (2/7)*C3^2-(6/7)*C4^2+(2/7)*A3^2-(30/7)*B3^2...
    -(30/7)*D3^2-(6/7)*A4^2+(26/7)*B4^2+(26/7)*D4^2...
    +(3/8)*A3*sqrt(2)+(5/8)*A4*sqrt(2)-(24/7)*A3*A4...
    +(24/7)*B3*B4-(24/7)*C3*C4+(24/7)*D3*D4;
f3 = -(46/49)*A3^2+(32/49)*A3*A4-(1/14)*A3*sqrt(2)...
    +(20/49)*A4^2-(45/112)*A4*sqrt(2)+45/256-(50/7)*B3^2...
    +(144/7)*B3*B4-(100/7)*B4^2-(46/49)*C3^2+(32/49)*C3*C4...
    +(20/49)*C4^2-(50/7)*D3^2+(144/7)*D3*D4-(100/7)*D4^2;
f4 = (55/98)*A3^2+(18/49)*A3*A4-(65/224)*A3*sqrt(2)...
    +(5/98)*A4^2-(5/56)*A4*sqrt(2)+75/1024-(15/14)*B3^2...
    +(26/7)*B3*B4-(45/14)*B4^2+(55/98)*C3^2+(18/49)*C3*C4...
    +(5/98)*C4^2-(15/14)*D3^2+(26/7)*D3*D4-(45/14)*D4^2;
f5 = (173/49)*C3^2+(143/49)*C4^2+(173/49)*A3^2+(781/49)*B3^2...
    +(781/49)*D3^2+(143/49)*A4^2+(1551/49)*B4^2...
    +(1551/49)*D4^2-315/512-(83/112)*A3*sqrt(2)...
    -(11/14)*A4*sqrt(2)+(44/49)*A3*A4-(1980/49)*B3*B4...
    +(44/49)*C3*C4-(1980/49)*D3*D4;

f6 = 2*A0+2*A1+2*A2+2*A3+2*A4-sqrt(2);
f7 = 2*C0+2*C1+2*C2+2*C3+2*C4;
f8 = 2*B0-2*B1+2*B2-2*B3+2*B4;
f9 = 2*D0-2*D1+2*D2-2*D3+2*D4;
f10 = 7*A1-5*A2+3*A3-A4-9*A0;
f11 = -9*B1+9*B2-9*B3+9*B4+9*B0;
f12 = 7*C1-5*C2+3*C3-C4-9*C0;
f13 = -9*D1+9*D2-9*D3+9*D4+9*D0;
f14 = 63*A1-45*A2+27*A3-9*A4-81*A0;
f15 = -65*B1+53*B2-45*B3+41*B4+81*B0;
f16 = 63*C1-45*C2+27*C3-9*C4-81*C0;
f17 = -65*D1+53*D2-45*D3+41*D4+81*D0;
f18 = 511*A1-335*A2+189*A3-61*A4-729*A0;
```

```
f19 = -513*B1+351*B2-243*B3+189*B4+729*B0;
f20 = 511*C1-335*C2+189*C3-61*C4-729*C0;
f21 = -513*D1+351*D2-243*D3+189*D4+729*D0;
f22 = 4095*A1-2385*A2+1215*A3-369*A4-6561*A0;
f23 = -4097*B1+2417*B2-1377*B3+881*B4+6561*B0;
f24 = 4095*C1-2385*C2+1215*C3-369*C4-6561*C0;
f25 = -4097*D1+2417*D2-1377*D3+881*D4+6561*D0;

% Since some equations above could be negative, we add the
% squares of each one

y = f1^2+f2^2+f3^2+f4^2+f5^2+f6^2+f7^2+f8^2+f9^2+f10^2+f11^2...
   +f12^2+f13^2+f14^2+f15^2+f16^2+f17^2+f18^2+f19^2+f20^2...
   +f21^2+f22^2+f23^2+f24^2+f25^2;

end
```

Appendix D

MATLAB® for a Wavelet Filter from a Scaling Filter

Our version of Ginzberg's MATLAB® script, in this case to find wavelet filter coefficients corresponding to our length 10 scaling filter:

```
close all
clear all

% Script for running the rewritten G2GH function of Ginzberg
% 15/07/2017

% Length of filter

L = 10;

% My new length 10 quaternion scaling filter coefficients
% (filter is symmetric, so only one half)

a{1} = [-(25/3072)*sqrt(2),-(1/3222528)*sqrt(4170519790),0,0];
a{2} = [-(85/3072)*sqrt(2),...
  (425/1221338112)*sqrt(4170519790),0,(1/72768)*sqrt(1472415)];
a{3} = [(1/192)*sqrt(2),(491/152667264)*sqrt(4170519790),0,...
  (1/24256)*sqrt(1472415)];
a{4} = [(65/384)*sqrt(2),(145/76333632)*sqrt(4170519790),0,...
  (1/72768)*sqrt(1472415)];
a{5} = [(185/512)*sqrt(2),-(1/194048)*sqrt(4170519790),0,...
  -(5/72768)*sqrt(1472415)];

```

```
% Basis quaternions as real 4 x 4 matrices

s = eye(4);
i = [0,-1,0,0;1,0,0,0;0,0,0,-1;0,0,1,0];
j = [0,0,-1,0;0,0,0,1;1,0,0,0;0,-1,0,0];
k = [0,0,0,-1;0,0,-1,0;0,1,0,0;1,0,0,0];

% 3-D array of zeros to take the scaling filter coefficients

S = repmat(0,[4,4,L]);

% Fill S with the scaling filter coefficients

for m = 1:L/2
    v = a{m};
    M = v(1)*s+v(2)*i+v(3)*j+v(4)*k;
    S(1:4,1:4,m) = M(1:4,1:4);
    S(1:4,1:4,L+1-m) = M(1:4,1:4);
end

% Find the wavelet filter coefficients: the result will be L
% (4 x 4) matrices

[S,W] = G2GH(S);

% Display W

format long

W
```

Our version of Ginzberg's `G2GH`, `G2polyphase` and `polyphase2GH` functions.

```
function [G,H] = G2GH(G)

% INPUT: a matrix-valued scaling filter G as an (n x n x L) 3-D
% array

% OUTPUT: A matrix-valued scaling filter G and a corresponding
% matrix-valued wavelet filter H as (n x n x L) 3-D arrays

[G,H] = polyphase2GH(G2polyphase(G));

end
```

```



function P = G2polyphase(G)

% This uses function from Keinert's mw toolbox, available from
% http://orion.math.iastate.edu/keinert/book.html

% INPUT: a matrix-valued scaling filter G as an (n x n x L) 3-D
% array

% OUTPUT: a (2n x 2n) polyphase matrix P of class mpoly

[n,m,L] = size(G);

% Write the (n x 2n) top half of the polyphase matrix as an
% mpoly object P1

Pg = zeros(n,2*n,L/2);

for k = 1:L/2
    Pg(1:n,1:2*m,k) = [G(1:n,1:m,2*k-1),G(1:n,1:m,2*k)];
end

P1 = mpoly(Pg,0,'polyphase',2,2);

% Find the projection factorisation of the polyphase matrix

F = projection_factorization(P1);

% Find the full (2n x 2n) polyphase matrix from the
% factorisation by unitary completion of the constant
% coefficient.

lf = length(F);

P = [eye(n),eye(n);-eye(n),eye(n)]/sqrt(2);

for k = 2:lf
    P = P*F{k}; % This is mpoly multiplication, not ordinary
                % MATLAB multiplication
end

% This last P will be a (2n x 2n x L/2) 3-D array of class
% mpoly

end



function [G,H] = polyphase2GH(P)

% INPUT: a polyphase matrix P as a (2n x 2n x L/2) 3-D array

% OUTPUT: an (n x n) matrix-valued scaling filter G and wavelet
% filter H as (n x n x L) arrays

P = P.coef;

L = 2*size(P,3);

n = size(P,1)/2;

G = zeros(n,n,L);

H = G;

for k = 1:L/2
    G(:,:,2*k-1) = P(1:n,1:n,k);
    G(:,:,2*k) = P(1:n,n+1:2*n,k);
    H(:,:,2*k-1) = P(n+1:2*n,1:n,k);
    H(:,:,2*k) = P(n+1:2*n,n+1:2*n,k);
end

end
```

Appendix E

Example of MATLAB® for Scaling and Wavelet Functions

An example of our MATLAB® code for using the cascade algorithm to calculate quaternion scaling and wavelet functions and display them using all possible projections onto two and three dimensions.

```
close all
clear all

% Scaling filter

s1 = quaternion(-(25/3072)*sqrt(2),...
  -(1/3222528)*sqrt(4170519790),0,0);
s2 = quaternion(-(85/3072)*sqrt(2),...
  (425/1221338112)*sqrt(4170519790),0,(1/72768)*sqrt(1472415));
s3 = quaternion((1/192)*sqrt(2),...
  (491/152667264)*sqrt(4170519790),0,(1/24256)*sqrt(1472415));
s4 = quaternion((65/384)*sqrt(2),...
  (145/76333632)*sqrt(4170519790),0,(1/72768)*sqrt(1472415));
s5 = quaternion((185/512)*sqrt(2),...
  -(1/194048)*sqrt(4170519790),0,-(5/72768)*sqrt(1472415));

s = [s1,s2,s3,s4,s5,s5,s4,s3,s2,s1];
v = s;

s = cascade(17,s,v,v);
S = s;
```

```
sz = 12; % size of each circle

u = length(s);
U = u;
max_s = max(abs(s.s));
max_i = max(abs(s.x));
max_j = max(abs(s.y));
max_k = max(abs(s.z));
max_a = max(max_s,max_i);
max_b = max(max_j,max_k);
m = max(max_a,max_b);
m1 = m;
t = 1:u;

figure
scatter(t,s.s,sz,'b','filled'); % 1
axis square
xlim([0,u]);
los = min(s.s);
his = max(s.s);
ylim([los,his]);
xticks([0,u]);
xticklabels({});
yticks([]);

figure
scatter(t,s.x,sz,'b','filled'); % 2
axis square
xlim([0,u]);
lox = min(s.x);
hix = max(s.x);
ylim([lox,hix]);
xticks([0,u]);
xticklabels({});
yticks([]);

figure
scatter(t,s.y,sz,'b','filled'); % 3
axis square
xlim([0,u]);
loy = min(s.y);
hiy = max(s.y);
ylim([loy,hiy]);
xticks([0,u]);
```

```
xticklabels({});
yticks([]);

figure
scatter(t,s.z,sz,'b','filled'); % 4
axis square
xlim([0,u]);
loz = min(s.z);
hiz = max(s.z);
ylim([loz,hiz]);
xticks([0,u]);
xticklabels({});
yticks([]);

figure
scatter(s.s,s.x,sz,'b','filled'); % 5
axis square
xlim([los,his]);
ylim([lox,hix]);
xticks([los,his]);
yticks([lox,hix]);
xticklabels({});
yticklabels({});

figure
scatter(s.s,s.y,sz,'b','filled'); % 6
axis square
xlim([los,his]);
ylim([loy,hiy]);
xticks([los,his]);
yticks([loy,hiy]);
xticklabels({});
yticklabels({});

figure
scatter(s.s,s.z,sz,'b','filled'); % 7
axis square
xlim([los,his]);
ylim([loz,hiz]);
xticks([los,his]);
yticks([loz,hiz]);
xticklabels({});
yticklabels({});

```

```
figure
scatter(s.x,s.y,sz,'b','filled'); % 8
axis square
xlim([lox,hix]);
ylim([loy,hiy]);
xticks([lox,hix]);
yticks([loy,hiy]);
xticklabels({});
yticklabels({});

figure
scatter(s.x,s.z,sz,'b','filled'); % 9
axis square
xlim([lox,hix]);
ylim([loz,hiz]);
xticks([lox,hix]);
yticks([loz,hiz]);
xticklabels({});
yticklabels({});

figure
scatter(s.y,s.z,sz,'b','filled'); % 10
axis square
xlim([loy,hiy]);
ylim([loz,hiz]);
xticks([loy,hiy]);
yticks([loz,hiz]);
xticklabels({});
yticklabels({});

figure
scatter3(s.s,s.x,t,sz,'b','filled'); % 11
xlim([-m,m]);
ylim([-m,m]);
zlim([0,u]);
xticks([-m,0,m]);
xticklabels({'','',''});
yticks([-m,0,m]);
yticklabels({'','',''});
zticks([0 u/3 2*u/3 u]);
zticklabels({'','','',''});
ax=gca;
ax.GridLineStyle = '-';
ax.GridColor = 'k';
ax.GridAlpha=0.25;

figure
scatter3(s.s,s.y,t,sz,'b','filled'); % 12
xlim([-m,m]);
ylim([-m,m]);
zlim([0,u]);
xticks([-m,0,m]);
xticklabels({'','',''});
yticks([-m,0,m]);
yticklabels({'','',''});
zticks([0 u/3 2*u/3 u]);
zticklabels({'','','',''});
ax=gca;
ax.GridLineStyle = '-';
ax.GridColor = 'k';
ax.GridAlpha=0.25;

figure
scatter3(s.s,s.z,t,sz,'b','filled'); % 13
xlim([-m,m]);
ylim([-m,m]);
zlim([0,u]);
xticks([-m,0,m]);
xticklabels({'','',''});
yticks([-m,0,m]);
yticklabels({'','',''});
zticks([0 u/3 2*u/3 u]);
zticklabels({'','','',''});
ax=gca;
ax.GridLineStyle = '-';
ax.GridColor = 'k';
ax.GridAlpha=0.25;

figure
scatter3(s.x,s.y,t,sz,'b','filled'); % 14
xlim([-m,m]);
ylim([-m,m]);
zlim([0,u]);
xticks([-m,0,m]);
xticklabels({'','',''});
yticks([-m,0,m]);
yticklabels({'','',''});
zticks([0 u/3 2*u/3 u]);
```

```
zticklabels({'','','',''});
ax=gca;
ax.GridLineStyle = '-';
ax.GridColor = 'k';
ax.GridAlpha=0.25;

figure
scatter3(s.x,s.z,t,sz,'b','filled'); % 15
xlim([-m,m]);
ylim([-m,m]);
zlim([0,u]);
xticks([-m,0,m]);
xticklabels({'','',''});
yticks([-m,0,m]);
yticklabels({'','',''});
zticks([0 u/3 2*u/3 u]);
zticklabels({'','','',''});
ax=gca;
ax.GridLineStyle = '-';
ax.GridColor = 'k';
ax.GridAlpha=0.25;

figure
scatter3(s.y,s.z,t,sz,'b','filled'); % 16
xlim([-m,m]);
ylim([-m,m]);
zlim([0,u]);
xticks([-m,0,m]);
xticklabels({'','',''});
yticks([-m,0,m]);
yticklabels({'','',''});
zticks([0 u/3 2*u/3 u]);
zticklabels({'','','',''});
ax=gca;
ax.GridLineStyle = '-';
ax.GridColor = 'k';
ax.GridAlpha=0.25;

figure
scatter3(s.s,s.x,s.y,sz,'b','filled'); % 17
xlim([-m,m]);
ylim([-m,m]);
zlim([-m,m]);
xticks([-m,0,m]);
```

```
xticklabels({'','',''});
yticks([-m,0,m]);
yticklabels({'','',''});
zticks([-m,0,m]);
zticklabels({'','',''});
ax=gca;
ax.GridLineStyle = '-';
ax.GridColor = 'k';
ax.GridAlpha=0.25;

figure
scatter3(s.s,s.x,s.z,sz,'b','filled'); % 18
xlim([-m,m]);
ylim([-m,m]);
zlim([-m,m]);
xticks([-m,0,m]);
xticklabels({'','',''});
yticks([-m,0,m]);
yticklabels({'','',''});
zticks([-m,0,m]);
zticklabels({'','',''});
ax=gca;
ax.GridLineStyle = '-';
ax.GridColor = 'k';
ax.GridAlpha=0.25;

figure
scatter3(s.s,s.y,s.z,sz,'b','filled'); % 19
xlim([-m,m]);
ylim([-m,m]);
zlim([-m,m]);
xticks([-m,0,m]);
xticklabels({'','',''});
yticks([-m,0,m]);
yticklabels({'','',''});
zticks([-m,0,m]);
zticklabels({'','',''});
ax=gca;
ax.GridLineStyle = '-';
ax.GridColor = 'k';
ax.GridAlpha=0.25;

figure
scatter3(s.x,s.y,s.z,sz,'b','filled'); % 20
```

```
xlim([-m,m]);
ylim([-m,m]);
zlim([-m,m]);
xticks([-m,0,m]);
xticklabels({'','',''});
yticks([-m,0,m]);
yticklabels({'','',''});
zticks([-m,0,m]);
zticklabels({'','',''});
ax=gca;
ax.GridLineStyle = '-';
ax.GridColor = 'k';
ax.GridAlpha=0.25;
```

For the wavelet function, we repeat the same code, but using the wavelet filter, and choose `r` in place of `b` as a colour option in the `scatter` and `scatter3` functions.

The cascade function used in our script above:

```
function s = cascade (n,t,c,x)

% The cascade function as written by John Burkardt in 2011, as
% modified by us: in the code below, x has to be defined as a
% quaternion array before we can do anything with it, hence the
% extra input x

% The original function (accessed 20/08/2018):
% http://people.sc.fsu.edu/~jburkardt/m_src/wavelet/cascade.m

s = t;

for i = 1 : n
    nx = length(s)*2-1;
    x(1:2:nx) = s;
    x(2:2:nx-1) = 0;
    s = conv(x,c);
end

%  Force S to be a column vector.

s = s(:);
```

```

return

end
```

With the $Cl(1,1)$ scaling and wavelet functions, we modify our quaternion script thus:

```
close all
clear all

clifford_signature(1,1);

% Scaling filter

s1 = -0.033422438260461*e0-0.001075680435047*e1...
   -0.039029301042422*e2-0.045881706741888*e12;
s2 = -0.059428506929067*e0-0.005811186899868*e1...
   -0.117412928715765*e2-0.143031714180422*e12;
s3 = 0.099865720860175*e0-0.012920727973642*e1...
   -0.079033678850327*e2-0.026932969773814*e12;
s4 = 0.328654553494030*e0-0.014651866872188*e1...
   +0.077733576496353*e2+0.318574308342227*e12;
s5 = 0.371437452021874*e0-0.006466645363360*e1...
   +0.157742332112162*e2+0.248357270677511*e12;
s6 = 0.371437452021871*e0+0.006466645363363*e1...
   +0.157742332112161*e2-0.248357270677505*e12;
s7 = 0.328654553494028*e0+0.014651866872180*e1...
   +0.077733576496354*e2-0.318574308342237*e12;
s8 = 0.099865720860176*e0+0.012920727973644*e1...
   -0.079033678850328*e2+0.026932969773813*e12;
s9 = -0.059428506929068*e0+0.005811186899869*e1...
   -0.117412928715766*e2+0.143031714180426*e12;
s10 = -0.033422438260462*e0+0.001075680435049*e1...
   -0.039029301042421*e2+0.045881706741886*e12;

s = [s1,s2,s3,s4,s5,s6,s7,s8,s9,s10];
v = s;
SZ = 12;

s = cascade_Cl_p_q(15,s,v,v);

ss = part(s,1);
sx = part(s,2);
sy = part(s,3);
```

```
sz = part(s,4);

u = length(s);
max_e0 = max(abs(ss));
max_e1 = max(abs(sx));
max_e2 = max(abs(sy));
max_e12 = max(abs(sz));
m1 = max(max_e0,max_e1);
m2 = max(max_e2,max_e12);
m = max(m1,m2);
t = 1:u;

figure
scatter(t,ss,SZ,'b','filled'); % 1
axis square
xlim([0,u]);
los = min(ss);
his = max(ss);
ylim([los,his]);
xticks([0,u]);
xticklabels({});
yticks([]);
```

...and so on.

Our revised cascade function `cascade_Cl_p_q` makes use of a very slightly edited version of the quaternion `conv` function in the `qtfm` toolbox, which we have called `convc`: this replaces lines 76 to 80 of `conv` with

```
if LRS(1)==1
     C = zeros(1,m+n-1)*e0; % L or R or both is/are a row vector
else
     C = zeros(m+n-1,1)*e0; % L or R or both is/are a column vector
end
```

where the two `*e0` cause `C` to be a zero multivector.

Appendix F

Example of MATLAB® for Wavelet Transforms

Code for finding quaternion wavelet transforms of colour vector images, in this case using our length 10 quaternion filters:

```
close all
clear all

% Test image

L = cast(imreadq('Lena.tiff'),'double')./255;
%L = cast(imreadq('Mandrill.tiff'),'double')./255;
%L = cast(imreadq('Yacht.tiff'),'double')./255;
%L = cast(imreadq('PO.PNG'),'double')./255;

N = size(L,1);

% Original image:

figure
image(L)
axis off
axis square

A0 = L;

% A number to scale up the results slightly, to make them a bit
% brighter

sd = max(max(std2(L.x),std2(L.y)),std2(L.z));

% Move the original to the centre of the unit 4-D cube

L = quaternion(L.s-0.5,L.x-0.5,L.y-0.5,L.z-0.5);

% Scaling filter

s1 = quaternion(-(25/3072)*sqrt(2),...
  -(1/3222528)*sqrt(4170519790),0,0);
s2 = quaternion(-(85/3072)*sqrt(2),...
  (425/1221338112)*sqrt(4170519790),0,(1/72768)*sqrt(1472415));
s3 = quaternion((1/192)*sqrt(2),...
  (491/152667264)*sqrt(4170519790),0,(1/24256)*sqrt(1472415));
s4 = quaternion((65/384)*sqrt(2),...
  (145/76333632)*sqrt(4170519790),0,(1/72768)*sqrt(1472415));
s5 = quaternion((185/512)*sqrt(2),...
  -(1/194048)*sqrt(4170519790),0,-(5/72768)*sqrt(1472415));

s = [s1,s2,s3,s4,s5,s5,s4,s3,s2,s1];

% Wavelet filter

w1 = quaternion(0.011508899433375,-0.019328607701528,...
  0.000785610562363,-0.005233602142379);
w2 = quaternion(-0.039130258073474,-0.026029448073557,...
  -0.003928052811814,0.009492671472039);
w3 = quaternion(-0.007365695637360,0.213388598131364,...
  0.006284884498903,0.008157200580536);
w4 = quaternion(0.239385108214196,-0.122672486581195,0,...
  -0.016675339239856);
w5 = quaternion(-0.510995134841841,-0.342761925084588,...
  -0.010998547873079,-0.010106266205973);

w = [w1,w2,w3,w4,w5,-w5,-w4,-w3,-w2,-w1];

% The (2 x 2n) submatrix for the leading diagonal of the banded
% matrix

P = [s;w];

% Four levels of quaternion wavelet transform analysis
```

```

[App1,Horiz1,Vert1,Diag1] = QWT_2D_total(L,P,N);

[App2,Horiz2,Vert2,Diag2] = QWT_2D_total(App1,P,N/2);

[App3,Horiz3,Vert3,Diag3] = QWT_2D_total(App2,P,N/4);

[App4,Horiz4,Vert4,Diag4] = QWT_2D_total(App3,P,N/8);

% Approximations

A1 = App1;A2 = App2;A3 = App3;A4 = App4;

% Greyscale approximations

A1s = A1.s+1;A2s = A2.s+1;A3s = A3.s+1;A4s = A4.s+1;
SA1 = quaternion(A1s,A1s,A1s);
SA2 = quaternion(A2s,A2s,A2s);
SA3 = quaternion(A3s,A3s,A3s);
SA4 = quaternion(A4s,A4s,A4s);

% Colour approximations

A1 = quaternion(A1.x+0.5,A1.y+0.5,A1.z+0.5);
A2 = quaternion(A2.x+0.5,A2.y+0.5,A2.z+0.5);
A3 = quaternion(A3.x+0.5,A3.y+0.5,A3.z+0.5);
A4 = quaternion(A4.x+0.5,A4.y+0.5,A4.z+0.5);

% Horizontal detail

H1 = Horiz1;H2 = Horiz2;H3 = Horiz3;H4 = Horiz4;
H1 = H1./sd;
H2 = H2./sd;
H3 = H3./sd;
H4 = H4./sd;

% Greyscale horizontal detail

H1s = H1.s+0.5;H2s = H2.s+0.5;H3s = H3.s+0.5;H4s = H4.s+0.5;
SH1 = quaternion(H1s,H1s,H1s);
SH2 = quaternion(H2s,H2s,H2s);
SH3 = quaternion(H3s,H3s,H3s);
SH4 = quaternion(H4s,H4s,H4s);

```

```
% Colour horizontal detail

H1 = quaternion(H1.x+0.5,H1.y+0.5,H1.z+0.5);
H2 = quaternion(H2.x+0.5,H2.y+0.5,H2.z+0.5);
H3 = quaternion(H3.x+0.5,H3.y+0.5,H3.z+0.5);
H4 = quaternion(H4.x+0.5,H4.y+0.5,H4.z+0.5);

% Vertical detail

V1 = Vert1;V2 = Vert2;V3 = Vert3;V4 = Vert4;
V1 = V1./sd;
V2 = V2./sd;
V3 = V3./sd;
V4 = V4./sd;

% Greyscale vertical detail

V1s = V1.s+0.5;V2s = V2.s+0.5;V3s = V3.s+0.5;V4s = V4.s+0.5;
SV1 = quaternion(V1s,V1s,V1s);
SV2 = quaternion(V2s,V2s,V2s);
SV3 = quaternion(V3s,V3s,V3s);
SV4 = quaternion(V4s,V4s,V4s);

% Colour vertical detail

V1 = quaternion(V1.x+0.5,V1.y+0.5,V1.z+0.5);
V2 = quaternion(V2.x+0.5,V2.y+0.5,V2.z+0.5);
V3 = quaternion(V3.x+0.5,V3.y+0.5,V3.z+0.5);
V4 = quaternion(V4.x+0.5,V4.y+0.5,V4.z+0.5);

% Diagonal detail

D1 = Diag1;D2 = Diag2;D3 = Diag3;D4 = Diag4;
D1 = D1./sd;
D2 = D2./sd;
D3 = D3./sd;
D4 = D4./sd;

% Greyscale diagonal detail

D1s = D1.s+0.5;D2s = D2.s+0.5;D3s = D3.s + 0.5;D4s = D4.s+0.5;
SD1 = quaternion(D1s,D1s,D1s);
SD2 = quaternion(D2s,D2s,D2s);
SD3 = quaternion(D3s,D3s,D3s);
```

```
SD4 = quaternion(D4s,D4s,D4s);

% Colour diagonal detail

D1 = quaternion(D1.x+0.5,D1.y+0.5,D1.z+0.5);
D2 = quaternion(D2.x+0.5,D2.y+0.5,D2.z+0.5);
D3 = quaternion(D3.x+0.5,D3.y+0.5,D3.z+0.5);
D4 = quaternion(D4.x+0.5,D4.y+0.5,D4.z+0.5);

% We have found all the approximation and detail pure
% quaternion arrays, but we only display the analysis to three
% levels

% Greyscale analysis

SC3 = [[[SA3,SH3;SV3,SD3],SH2;SV2,SD2],SH1;SV1,SD1];

% Lines between detail images and approximation in the
% greyscale analysis

SC3(N/2+1,1:N) = quaternion(1,1,1);
SC3(1:N,N/2+1) = quaternion(1,1,1);
SC3(N/4,1:N/2) = quaternion(1,1,1);
SC3(1:N/2,N/4+1) = quaternion(1,1,1);
SC3(N/8,1:N/4) = quaternion(1,1,1);
SC3(1:N/4,N/8+1) = quaternion(1,1,1);

% Greyscale analysis to level three

figure
image(SC3)
axis off
axis square

% Colour analysis

I3 = [[[A3,H3;V3,D3],H2;V2,D2],H1;V1,D1];

% Lines between detail images and approximation in the colour
% analysis

I3(257,1:512) = quaternion(1,1,0);
I3(1:512,257) = quaternion(1,1,0);
I3(128,1:256) = quaternion(1,1,0);
```

```
I3(1:256,129) = quaternion(1,1,0);
I3(65,1:128) = quaternion(1,1,0);
I3(1:128,65) = quaternion(1,1,0);

% Colour analysis to level three

figure
image(I3)
axis off
axis square

% Standard deviation (t), skewness (s) and kurtosis (k) of red
% in Horiz1

r = Horiz1.x;
t = std2(r);
r = reshape(r,[],1);
s = skewness(r);
k = kurtosis(r);

% Total detail in original image

a0 = image_detail(A0);

% Detail in first level

a1 = image_detail(App1);
h1 = image_detail(Horiz1);
v1 = image_detail(Vert1);
d1 = image_detail(Diag1);

% Force sum to be a0

e = a0/(a1+h1+v1+d1);
a1 = e*a1;
h1 = round(e*h1*100/a0,2);
v1 = round(e*v1*100/a0,2);
d1 = round(e*d1*100/a0,2);

% Detail in second level

a2 = image_detail(App2);
h2 = image_detail(Horiz2);
v2 = image_detail(Vert2);
```

```
d2 = image_detail(Diag2);

% Force sum to be a1

f = a1/(a2+h2+v2+d2);
a2 = f*a2;
h2 = round(f*h2*100/a0,2);
v2 = round(f*v2*100/a0,2);
d2 = round(f*d2*100/a0,2);

% Detail in third level

a3 = image_detail(App3);
h3 = image_detail(Horiz3);
v3 = image_detail(Vert3);
d3 = image_detail(Diag3);

% Force sum to be a2

g = a2/(a3+h3+v3+d3);
h3 = round(g*h3*100/a0,2);
v3 = round(g*v3*100/a0,2);
d3 = round(g*d3*100/a0,2);

% Detail in remaining approximation

a3 = 100-(h3+v3+d3+h2+v2+d2+h1+v1+d1);
```

The image_detail function used in the above script:

```
function detail_total = image_detail(Im)

% Assume input image is a full quaternion

S = Im.s;
R = Im.x;
G = Im.y;
B = Im.z;

S = S-mean(mean(S));
R = R-mean(mean(R));
G = G-mean(mean(G));
B = B-mean(mean(B));

```

```
detail_total = sum(sum(sqrt(S.^2+R.^2+G.^2+B.^2)));

end
```