EXPLORATION IN QUATERNION COLOUR

by

Lilong Shi
Bachelor of Applied Science
Simon Fraser University 2003

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

In the School of Computing Science

© Lilong Shi 2005

SIMON FRASER UNIVERSITY

Summer 2005

All rights reserved. This work may not be
reproduced in whole or in part, by photocopy
or other means, without permission of the author.

APPROVAL

Name:	**Lilong Shi**
Degree:	**Master of Applied Science**
Title of Research Project:	**Reviewing Your Own Work: Exploration in Quaternion Colour**

Examining Committee:

Chair: **Dr. Mark Drew**
Associate Professor, School of Computing Science

Dr. Brian Funt
Senior Supervisor
Professor, School of Computing Science

Dr. Greg Mori
Supervisor
Professor, School of Computing Science

Dr. Ze-Nian Li
Examiner
Professor, School of Computing Science

Date Defended/Approved: June 8, 2005

SIMON FRASER UNIVERSITY

PARTIAL COPYRIGHT LICENCE

The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the right to lend this thesis, project or extended essay to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users.

The author has further granted permission to Simon Fraser University to keep or make a digital copy for use in its circulating collection.

The author has further agreed that permission for multiple copying of this work for scholarly purposes may be granted by either the author or the Dean of Graduate Studies.

It is understood that copying or publication of this work for financial gain shall not be allowed without the author's written permission.

Permission for public performance, or limited permission for private scholarly use, of any multimedia materials forming part of this work, may have been granted by the author. This information may be found on the separately catalogued multimedia material and in the signed Partial Copyright Licence.

The original Partial Copyright Licence attesting to these terms, and signed by this author, may be found in the original bound copy of this work, retained in the Simon Fraser University Archive.

W. A. C. Bennett Library
Simon Fraser University
Burnaby, BC, Canada

ABSTRACT

Quaternions have been a subject for research in mathematics, physics, and engineering for decades. Using quaternions to represent colours, however, has recently been proposed and studied. Colour sensitive filtering can be achieved using a quaternion-valued filter. There are two colour edge detection methods defined on quaternions. One is based on the chromaticity cancellation that generates a grey colour for non-boundary regions; the other is based on estimation of homogeneity of regions. Trilateral filtering is proposed by locally adapting colour and changing the shape of the filter to achieve the effect of smoothing colours yet preserving edges. Furthermore, quaternion Fourier analysis has been implemented efficiently by decomposition into complex Fourier equivalence, which allows convolution and cross-correlation on quaternion-valued images in the frequency domain. Finally, based on quaternion singular value decomposition, quaternion principle analysis is implemented and applied to several applications, such as segmentation of colour images.

ACKNOWLEDGEMENTS

I would like to express my gratitude to Brian Funt, who has been greatly supportive throughout the duration of this thesis, for his patience, thoughtfulness, and perceptiveness in the field of computer vision. I am also indebted to Greg Mori and Ze-Nian Li for their researching insight and enthusiastic input.

I would like to thank my fellow graduate students in the Computational Vision Lab. They contribute greatly to the general working environment and more importantly supply support and friendship. I would also extend my love and appreciation to my parents, who have been the mostly supportive throughout my academic study.

Finally, I would like to acknowledge Simon Fraser University for their generous financial support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION TO QUATERNIONS

1.1 A Brief History

Complex numbers have been a subject for research since the early eighteenth century. Over the last century, theories based on complex numbers have been widely applied in mathematics, physics and engineering. Since it is so useful in other fields, it is natural to wonder, with a rule for multiplying two numbers together, whether it is possible to multiply numbers as a whole unit. Even the most famous mathematician of the eighteenth century, Hamilton had been thinking of a way for triple number multiplication, and finally invented quaternions [1].

Before Hamilton's discovery, Gauss received the credit for first seeing quaternions in one of his notebooks. In the early years of this century, Albert Einstein found a use for four dimensions, in order to make the speed of light constant for all inertial observers, space and time had to be united [15]. Today, quaternions are of great interest to historians of mathematics. For instance, vector analysis performs the daily mathematical routine that could be done with quaternions as an alternative. In addition, quaternions have been widely used in computer graphics for rotation in 3D space. Most recently, the use of quaternion to represent colours in image processing has been proposed and studied. Sangwine[23]-[30], as a pioneer to quaternion image process, has successfully used quaternions for colour image filtering, cross-correlation, and compression.

1.2 Hypercomplex Numbers

In Clifford algebra, Clifford algebraists name their higher dimensional numbers hypercomplex, even though these kinds of numbers do not share all the properties of complex numbers and no classical function theory can be constructed over them. A hypercomplex number is defined as a number that has properties departing from those of the real and complex numbers. The most common examples of hypercomplex numbers are biquaternions, octonions, and quaternions [22,13].

For example, one particular type of hypercomplex number defined by Davenport (1996) is sometimes called "the" hypercomplex number and is defined according to the multiplication table $ij = ji = k, jk = kj = -i, ki = ik = -j$, and also satisfies $i^2 = j^2 = k^2 = -1$.

Note that hypercomplex numbers are not equivalent to quaternions, as the multiplication of these hypercomplex numbers is commutative, as opposed to quaternions, which I will be discussing in this thesis. Also, unlike real and complex numbers, not all nonzero hypercomplex numbers have a multiplicative inverse. More discussion of quaternions and the difference between them and hypercomplex numbers will be given in the rest of this thesis.

1.2.1 Non-commutative 4D Hypercomplex Numbers

The quaternions of Hamilton are a system of hypercomplex numbers defined in four dimensions, with their multiplication being a non-commutative operation. Many other hypercomplex systems are possible, but these interesting hypercomplex systems do not have all the required properties of regular, two-dimensional complex numbers which make possible the development of the theories on functions of complex variables.

First of all, let us derive the multiplication of two quaternions based on the multiplication table and explain why this multiplication is non-commutative. Usually, to define any hypercomplex number, we first need to define the bases for the multiplication table. For example, Table 1 is the multiplication table for quaternions.

Table 1: The multiplication table for quaternions

	1	i	j	k
1	1	i	j	k
i	i	-1	k	-j
j	j	-k	-1	i
k	k	j	-i	-1

As we have noticed that this table is asymmetric. For example, the product of *k* in row 4 and *i* in column 2 is *j*, and the product of *i* in row 2 and *k* in column 4 is *–j*. The asymmetry of the multiplication table implies the non-commutative property of quaternion multiplication. Suppose we have two quaternions $a = (t,x,y,z)$ and $b = (t',x',y',z')$, the multiplication between each two components can be summarized in Table 2.

Table 2: The multiplication rules for quaternion a and b

	t'	x'	y'	z'
t	1	i	j	k
x	i	-1	k	-j
y	j	-k	-1	i
z	k	j	-i	-1

If we multiply quaternion *a* and *b* by every component as two vectors, the multiplication rule of two quaternions can be derived as

$$a \times b = (tt' - xx' - yy' - zz') + (tx' + xt' + yz' - zy')i + (ty' - xz' + yt' + zx')j + (tz' + xy' - yx' + zt')k \quad (1)$$

When *a* and *b* are pure quaternion, then $t = t' = 0$, we have

$$\begin{aligned} a \times b &= (-xx' - yy' - zz') + (yz' - zy')i + (-xz' + zx')j + (xy' - yx')k \\ &= [-dot(a,b), cross(a,b)] \end{aligned} \quad (2)$$

1.2.2 Commutative 4D Hypercomplex Numbers

A distinct system of hypercomplex numbers in n dimensions has been described in [17], for which the multiplication is associative and commutative, and which are rich enough in properties such that exponential and trigonometric forms exist and the concepts of analytic n-complex function, contour integration and residue can be defined. Here, two instances of commutative hypercomplex numbers are defined in four dimensions – the circular complex number and the hyperbolic complex number.

Circular Complex Number

First of all, let us derive the multiplication of two circular complex numbers based on the multiplication table and explain why this multiplication is commutative. To start, we need to define the bases for the multiplication table shown in Table 3.

Table 3: The multiplication table for circular complex numbers

	1	i	j	k
1	1	i	j	k
i	i	-1	k	-j
j	j	-k	-1	-i
k	k	-j	-i	1

As you can see in Table 3, the entries are symmetric. The asymmetry of the multiplication table for circular complex numbers implies the commutative property of circular complex number multiplication. Suppose we have two circular complex numbers *a = (t,x,y,z)* and *b = (t',x',y',z')*, if we multiply *a* and *b* component by component as two vectors, the multiplication rule of two circular complex numbers can be derived as

$$a\times b=(tt'-xx'-yy'+zz')+(tx'+xt'-yz'-zy')i+(ty'-xz'+yt'-zx')j+(tz'+xy'+yx'+zt')k \quad (3)$$

If we let $t = t' = 0$, we have

$$a\times b=(-xx'-yy'+zz')+(-yz'-zy')i+(-xz'-zx')j+(xy'+yx')k \quad (4)$$

Hyperbolic Complex Number

Let us derive the multiplication of two hyperbolic complex numbers based on the multiplication table and explain why this multiplication is commutative. Firstly, we need to define the bases for the multiplication table shown in Table 4.

Table 4: The multiplication table for hyperbolic complex numbers

	1	i	j	k
1	1	i	j	k
i	i	1	k	j
j	j	k	1	i
k	k	j	i	1

As you can see in Table 4, the entries are symmetric. The symmetry of the multiplication table for hyperbolic complex number implies the commutative property of circular complex number multiplication. In addition, note that, for this particular example of hypercomplex numbers, there is no minus sign in the table. The multiplication should be the in the form of summation and the result should be non-negative, given that all its components are positive. Suppose we have two hyperbolic complex numbers $a = (t,x,y,z)$ and $b = (t',x',y',z')$, if we multiply a and b component by component as two vectors, the multiplication rule of two hyperbolic complex numbers can be derived as

$$a \times b = (tt'+xx'+yy'+zz') + (tx'+xt'+yz'+zy')i + (ty'+xz'+yt'+zx')j + (tz'+xy'+yx'+zt')k \quad \mathbf{(5)}$$

If we let $t = t' = 0$, we have

$$\begin{aligned} a \times b &= (xx'+yy'+zz') + (yz'+zy')i + (xz'+zx')j + (xy'+yx')k \\ &= [dot(a,b),(yz'+zy'),(xz'+zx'),(xy'+yx')] \end{aligned} \quad \mathbf{(6)}$$

However, this hypercomplex number is not as mathematically sound as quaternions for two reasons. First, the magnitude of a real number, complex number and quaternions can be defined as $|a|^2 = a \times \bar{a}$. For quaternions, $a \times \bar{a} = [dot(a,a), 0, 0, 0]$, which is a scalar number, but for the pure hyperbolic number defined above, $a \times \bar{a} = [dot(a,a), -2yz, -2xz, -2xy]$, which is still of hyperbolic number instead of a scalar. Hence, the magnitude of a hyperbolic complex number can only be defined as $|a| = \sqrt{(tt + xx + yy + zz)}$. The second reason is that the hyperbolic complex number can not be used as a generalization of the complex number because the product of i and i yields a positive 1 rather than -1, based on the multiplication table. Therefore, it does not inherit the properties of complex number as quaternions do.

1.3 Quaternion

1.3.1 Definition of Quaternion

Definition: Quaternions are associative but non-commutative, and they are a single example of a more general class of hypercomplex numbers discovered by Hamilton. In analogy to the complex numbers that are representable as a combination of a real and an imaginary part $a \cdot 1 + b \cdot i$, a quaternion can be written as a linear combination as $a = a_0 \cdot 1 + a_1 \cdot i + a_2 \cdot j + a_3 \cdot k$. The quaternions satisfy the multiplication rules, sometimes known as Hamilton's rules, according to multiplication Table 1.

Conjugation. The quaternion conjugate of a, denoted by $\bar{a}$ or a^*, is computed as

$$\bar{a} = a_0 - a_1 \cdot i - a_2 \cdot j - a_3 \cdot k$$

Addition. The sum of two quaternions is then

$$a + b = (a_0 + b_0) + (a_1 + b_1)i + (a_2 + b_2)j + (a_3 + b_3)k$$

Multiplication. The product of two quaternions is

$$\begin{aligned} a \times b = \ & (a_0b_0 - a_1b_1 - a_2b_2 - a_3b_3) \\ & + (a_0b_1 + a_1b_0 + a_2b_3 - a_3b_2)i \\ & + (a_0b_2 + a_1b_3 + a_2b_0 - a_3b_1)j \\ & + (a_0b_3 + a_1b_2 + a_2b_1 - a_3b_0)k \end{aligned}$$

Quaternions can also be denoted in a vector form as a combination of a scalar plus a vector by writing $a = (a_0, a_1, a_2, a_3) = [S(a), V(a)]$, where $S(a) = a_0$ and $V(a) = (a_1, a_2, a_3)$.

In this notation, quaternion multiplication[1] has the particularly simple form:

[1] "$\circ$" denotes the dot-product operator, and "$*$" denotes the cross-product operator of two vectors.

$$\begin{aligned} a \times b &= (S[a], V[a]) \cdot (S[b], V[b]) \\ &= (S[a] \cdot S[b] - V[a] \circ V[b], S[a]V[b] + S[b]V[a] + V[a] * V[b]) \end{aligned}$$

Norm. The quaternion norm is defined as

$$norm(a) = \sqrt{a \times \overline{a}} = \sqrt{\overline{a} \times a} = \sqrt{a_0^2 + a_1^2 + a_2^2 + a_3^2}$$

and the norm is multiplicative such that $norm(a \times b) = norm(a) \cdot norm(b)$

Division. Division is uniquely defined (except by zero), so quaternions form a division algebra. The inverse of a quaternion is defined by $a^{-1} = \dfrac{\overline{a}}{a \times \overline{a}}$, given that the norm of a is non-zero.

Rotation. A rotation about the unit vector $\hat{\mu}$ by an angle θ can be computed using the quaternion $a = (s, \vec{v}) = \cos(\frac{1}{2}\theta), \hat{\mu} \cdot \sin(\frac{1}{2}\theta)$ [Arvo 1994, Hearn and Baker 1996]. After the rotation, a point p is then given by $p' = \hat{\mu} \times p \times \hat{\mu}^{-1} = \hat{\mu} \times p \times \overline{\hat{\mu}}$ since $norm(\hat{\mu}) = 1$. A concatenation of two rotations, first $\hat{\mu}_1$ and then $\hat{\mu}_2$, can be computed using the identity $\hat{\mu}_2 \times (\hat{\mu}_1 \times p \times \overline{\hat{\mu}_1}) \times \overline{\hat{\mu}_2} = (\hat{\mu}_2 \times \hat{\mu}_1) \times p \times (\overline{\hat{\mu}_1} \times \overline{\hat{\mu}_2}) = (\hat{\mu}_2 \times \hat{\mu}_1) \times p \times \overline{(\hat{\mu}_1 \times \hat{\mu}_2)}$ [Goldstein 1980].

Quaternion to Complex. The quaternions can be represented using its uniquely defined equivalent complex 2×2 matrices such that

$$a = \begin{bmatrix} z & w \\ -\overline{w} & \overline{z} \end{bmatrix} = \begin{bmatrix} a_0 + a_1 i & a_2 + a_3 i \\ -a_2 + a_3 i & a_0 - a_1 i \end{bmatrix}$$

where z and w are complex numbers, a_0, a_1, a_2 *and* a_3 are real numbers, and $\overline{z}$ and $\overline{w}$ are the complex conjugate of z and w. The operation is particularly useful in many examples of quaternion analysis, such as quaternion Singular Value decomposition, which will be discussed in a later chapter, where the standard SVD approach can be applied to an equivalent complex matrix form to simplify the computation.

1.3.2 Quaternion Arithmetic

In this section, some general quaternion algebra operations are listed, including trigonometry, exponentials and logarithms. The quaternion algebra is somewhat similar to complex algebra. To keep the arithmetic clear and simple, all quaternions are shown in the representation as a pair of a scalar t and a 3 dimensional vector $\boldsymbol{V}$, such that a quaternion $q = (t, \boldsymbol{V})$

Parts

q = t+**V** = (t, **V**)

scalar(q) = t = (q + q*)/2 = (t, 0)

vector(q) = v = (q - q*)/2 = (0, **V**)

Simple Algebra

q+p = (t + t', **V** + **V'**)

q-p = (t - t', **V** - **V'**)

conj(q) = q* = (t, -**V**)

|q| = (q* q)^.5 = ((t^2 + **V** ∘ **V**)^.5, 0)

q^(-1) = q*/(q* q) = (t, -**V**) / |q|^2

norm(q) = (t^2 + **V** ∘ **V**, 0)

adj(q) = q* q* q = (t, -**V**)

|q|^2det(q) = ((t^2 + **V** ∘ **V**)^2, 0)

Multiplication

(t t' + **V** ∘ **V'**, 0) (0, t**V'** - **V**t' - **V** ∗ **V'**) (t t' + **V**.**V'**, t**V'** - **V**t' - **V** ∗ **V'**)

Trigonometry

sin(q) = (sin(t) cosh(|**V**|), cos(t) sinh(|**V**|) **V**/|**V**|),

$\cos(q) = (\cos(t)\cosh(|\mathbf{V}|), -\sin(t)\sinh(|\mathbf{V}|)\ \mathrm{V}/|\mathbf{V}|)$

$\tan(q) = \sin(q) / \cos(q)$

$\operatorname{asin}(q) = -\mathbf{V}/|\mathbf{V}|\ \operatorname{asinh}(q\ \mathbf{V}/|\mathbf{V}|)$

$\operatorname{acos}(q) = -\mathbf{V}/|\mathbf{V}|\ \operatorname{acosh}(q)$

$\operatorname{atan}(q) = -\mathbf{V}/|\mathbf{V}|\ \operatorname{atanh}(q\ \mathbf{V}/|\mathbf{V}|)$

$\sinh(q) = (\sinh(t)\cos(|\mathbf{V}|), \cosh(t)\sin(|\mathbf{V}|)\ \mathrm{V}/|\mathbf{V}|)$

$\cosh(q) = (\cosh(t)\cos(|\mathbf{V}|), \sinh(t)\sin(|\mathbf{V}|)\ \mathrm{V}/|\mathbf{V}|)$

$\tanh(q) = \sinh(q) / \cosh(q)$

$\operatorname{asinh}(q) = \ln(q + (q^2 + 1)^{.5})$

$\operatorname{acosh}(q) = \ln(q \pm (q^2 - 1)^{.5})$

$\operatorname{atanh}(q) = .5\ \ln((1 + q)/(1 - q))$

Powers

$\exp(q) = (\exp(t)\cos(|\mathbf{V}|), \exp(t)\sin(|\mathbf{V}|)\ \mathbf{V}/|\mathbf{V}|)$

$q \wedge p = \exp(\ln(q) * p)$

Logarithms

$\ln(q) = (0.5\ \ln(t^2 + \mathbf{V}.\mathbf{V}), \operatorname{atan}(|\mathbf{V}|, t)\ \mathbf{V}/|\mathbf{V}|)$

$\log(q) = \ln(q)/\ln(10)$

The logarithm rule of real and complex numbers is define as: $\log(p*q) = \log(p) + \log(q)$. It should be noticed that the logarithm rule does not hold anymore for quaternions as it does for real and complex numbers, because of the non-commutative properties of the quaternion multiplications. Suppose we have two quaternions *p* and *q*, if the logarithm rule holds, then log(pq)=log(p)+log(q)=log(q)+log(p)=log(qp) => pq = qp. However, we already know this is not generally true for quaternions multiplication.

1.4 Hypercomplex Representation of Colour

As we have mentioned previously, quaternions can be considered a generalization of the complex numbers with one real part and three imaginary parts. Thus, colour images consisting of three colour channels can be represented using pure quaternion-valued pixels. For images in RGB colour space, the three imaginary parts can be used to represent the red, green and blue colour components, leaving the real part zero [26,27,28,29,30].

Using quaternions to represent the RGB colour space, the three colour channels are processed equally in operations such as multiplication. The advantage of using quaternion based operations to manipulate colour information in an image is that we do not have to process each colour channel independently, but rather, treat each colour triple as a whole unit. I believe, by using quaternion operations, higher colour information accuracy can be achieved because a colour is treated as an entity. In quaternion multiplication, each of the three imaginary components is multiplied in a similar manner with other components. For pure quaternion multiplication, the real component of the product is the negated dot product of two pure quaternions, and the imaginary components are the cross product of the two pure quaternions.

The colour space adopted for the image processing procedures is essential. The results of the procedure vary if pure quaternions are used to represent, for example, colours in Lab space, or Luv space, which are nonlinear transformations of the RGB colour space. In the colour sensitive filtering discussed in Chapter two, the optimal colour space and colour sensitivity for colour-specific boundary detection vary depending on the applications. On the other hand, for colour spaces like Luv and Lab, where L represents the brightness while uv and ab are the chromaticity, the three channels in Lab or Luv do not represent information in the "same" sense. In this case, it is not the best choice to use quaternion presentations.

In comparison to quaternions, the commutative hypercomplex numbers such as the circular complex number and the hyperbolic complex number can also be used for colour representation too. Each colour pixel can be manipulated with the operations corresponding to these commutative hypercomplex numbers. Using commutative hypercomplex numbers to represent colours is especially attractive when the commutative rule needs to hold. Although the commutativity does not seem to be that important in simple colour multiplication, the commutative rule is essential for the logarithm rule of the hypercomplex numbers. It has been shown in the previous section of this chapter that the non-commutative property of quaternion makes it impossible for the logarithm rule to hold. The circular complex number and the hyperbolic complex number, on the other hand, provide a way for the logarithm based image processing such as the homomorphic filtering [4] and Retinex [10]. Many operations on the commutative hypercomplex numbers have been well defined, and their Fourier transform has been discussed in [17]. However, one limitation of the commutative hypercomplex number is that it may not be a generalization of the complex number and many powerful techniques can not be implemented, due to the lack of support from the mathematical theories. Also, not all hypercomplex numbers may be a good choice to represent colour. For example, based on Equation(4), the multiplication of two circular complex numbers does not treat each colour channel equally, which makes it not satisfactory for processing the three "equivalent" colour channels.

In this thesis, due to the soundness and completeness of the theories of quaternions, only the quaternion representation of colour will be studied and discussed, so that a variety of colour image applications can be developed, namely, filtering, clustering, 2D Fourier transform and principle component analysis for colour images.

Quaternion Multiplication – Matrix point of view

One of the advantages of quaternion for colour representation is that it allows different colour channels to talk to each other rather than each channel being independently manipulated. A more detailed explanation will be given on how each component is calculated by showing the arithmetic of the multiplication operation. When two quaternions are multiplied, each component interacts with others and the effect spreads over all dimensions. I consider it as a good feature of quaternion because now each component also contains some information of the others. For example, suppose $a_q = a_0 + a_1 \cdot i + a_2 \cdot j + a_3 \cdot k$ and $x_q = x_0 + x_1 \cdot i + x_2 \cdot j + x_3 \cdot k$, then

$$y_q = a_q \cdot x_q = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} a_0x_0 - a_1x_1 - a_2x_2 - a_3x_3 \\ a_1x_0 + a_0x_1 - a_3x_2 + a_2x_3 \\ a_2x_0 + a_3x_1 + a_0x_2 - a_1x_3 \\ a_3x_0 - a_2x_1 + a_1x_2 + a_0x_3 \end{bmatrix} \qquad \textbf{(7)}$$

Now, consider two real number *a* and *x*. If *x* is a variable and *a* is a coefficient, then there is a $y = a \cdot x$. In the quaternion case, if *a* and *x* are both quaternions, the multiplication of the two quaternions causes the interaction between any two components of a_q and x_q. By (7, each component in the product is the summation of weighted variables x_0, x_1, x_2 *and* x_3, with the coefficients a_0, a_1, a_2, a_3. Let *x* represent a RGB colour triple, every element of the final product of *x* and *a* is related to all r, g and b channels. Therefore, for $y_q = a_q \cdot x_q$, we can consider a_q as a quaternion coefficient to the variable x_q.

2. QUATERNION-VALUED FILTERING

2.1 Quaternion Convolution

The most widely used approach for image filtering is convolving the image with a mask. In the quaternion case, a quaternion-valued mask can be defined instead of a real-valued mask. Since quaternion multiplication is non-commutative, convolving on the left, on the right, or on both sides will yield different filtering results. Generally, we can define a quaternion-based filtering by both left and right masks as below [9, 25]:

$$Q'(x,y) = \sum_{s=-r}^{r}\sum_{t=-r}^{r} h_L(s,t) \times Q(x+s, y+t) \times h_R(s,t) \qquad \textbf{(8)}$$

where $Q'(x,y)$ is the filtered pixel, and $Q(x,y)$ is a pixel in the original image, $h_L(s,t)$ and $h_R(s,t)$ are the left and right masks respectively, both of them have dimension (2r+1)x(2r+1). The operator "×" denotes multiplication of two quaternion numbers. More details of quaternion convolution will be discussed in Chapter 3. This this chapter, I will first review the quaternion convolution based filters such as the colour sensitive smoothing filter and the colour edge detector, based on which more sophisticated adaptive filters are proposed and discussed.

2.2 Colour Smoothing

Evans and Sangwine [25] have proposed a colour-sensitive filtering method to smooth a particular component of a colour in an image. Suppose we want to only smooth the component of colours in an image that are parallel to the direction of a colour *C*, but preserve the component orthogonal to *C*, given a colour image and *C* represented by a pure quaternion. Such a smoothing operation may be performed on the quaternion-valued image *Q* with a 3x3 uniformly weighted quaternion mask defined as:

$$U_c = \frac{1}{9}\begin{array}{|c|c|c|}\hline \hat{c} & \hat{c} & \hat{c} \\ \hline \hat{c} & \hat{c} & \hat{c} \\ \hline \hat{c} & \hat{c} & \hat{c} \\ \hline \end{array} \tag{9}$$

where $\hat{c}$ is the unit pure quaternion on the direction of C. For a pixel q_p at location *p*, convolve with the mask U_c on the left and we have:

$$q_p = \frac{1}{9} \sum_{s \in neighbour(p)} \hat{c} \times q_s = \sum_{s \in neighbour(p)} \frac{\hat{c}}{9} \times q_s \tag{10}$$

Convolving the quaternion mask U_c on the left side of Q is not the same as convolving it with the mask on the right, because of the non-commutative property of quaternion multiplication, so we should convolve the quaternion mask on either side of the image individually,

$$R_c = U_c \otimes Q + Q \otimes U_c \tag{11}$$

where $\otimes$ denotes the "convolution" operator[1]. For pure quaternions, the imaginary part of $U \otimes Q$ always equals to the negative imaginary part of $Q \otimes U$. i.e., $V[U \otimes Q] = -V[Q \otimes U]$(see footnote[2]). In addition, the scalar part of $U \otimes Q$ always equals to the scalar part of $Q \otimes U$. i.e., $S[U \otimes Q] = S[Q \otimes U]$. If the left and right convolution results are added together, the vector parts will cancel out, leaving only a scalar part, which is the double of the scalar resulted from a single convolution on either the left or right side alone. Hence, R_c on the right side of Equation(11) should be a scalar resulting from summing two scalar parts and cancelling two vector parts. Furthermore, this scalar can be oriented in the $\hat{c}$ direction of RGB colour space simply by multiplying it with $\hat{c}$, i.e. $R_c \cdot \hat{c}$. In this manner, the quaternion mask of

[1] Throughout the rest of the paper, we always use $\otimes$ to denote the convolution operator.

[2] For pure quaternions p and q, $S[p \times q] = -dot(p,q)$ and $V[p \times q] = cross(p,q)$

colour C can be used to smooth the colour image component in the direction of $\hat{c}$ yet preserving the colour image components perpendicular to $\hat{c}$, based on the following calculations

$$q'_p = \frac{R_c}{2} \times c + (q_p - \frac{r_c}{2} \times q_p) \tag{12}$$

$$r_c = c \times q + q \times c \tag{13}$$

where R_c is defined in Equation(11) and r_c can be obtained from Equation(13), which is two times the projection of quaternion q onto $\hat{c}$. The first term of Equation(12) is the average of neighbouring pixels parallel to colour C, and the second term is the perpendicular component of q to colour C.

By extending the above colour sensitive smoothing method with a filter defined by a constant colour C, I proposed a colour adaptive filter that changes colour while convolving with the image. The adapted colour is always the colour under the centre of the mask during convolution; in other words, it is the colour of the pixel that is to be smoothed. Then I design the new low pass filter by:

$$q'_p = \frac{R_{\hat{q}_p}}{2} \cdot \hat{q}_p = \frac{1}{2}\left[\sum_{s \in S} \frac{\hat{q}_p}{9} \times q_s + \sum_{s \in S} q_s \times \frac{\hat{q}_p}{9}\right] \cdot \hat{q}_p \tag{14}$$

Note that the perpendicular component of $\bar{q}$ in Equation(14) disappears, as it exists in Equation(12). The reason is that for the adaptive colour sensitive filter, the parallel component of q in the direction $\hat{q}$ is the $\|q\| \cdot \hat{q}$, and the perpendicular component of q on $\hat{q}$ is *0*. The result of the colour sensitive smoothing is illustrated in Figure 1:

Based on the adaptive filtered defined above, the colour under the centre of the mask is used to smooth a pixel. Some edges between two colours are blurred; while some are persevered, depending on the similarity of the two colours across the edges. Suppose there are two colours under the filter and this adaptive filtering is based on one of the colours, if these two

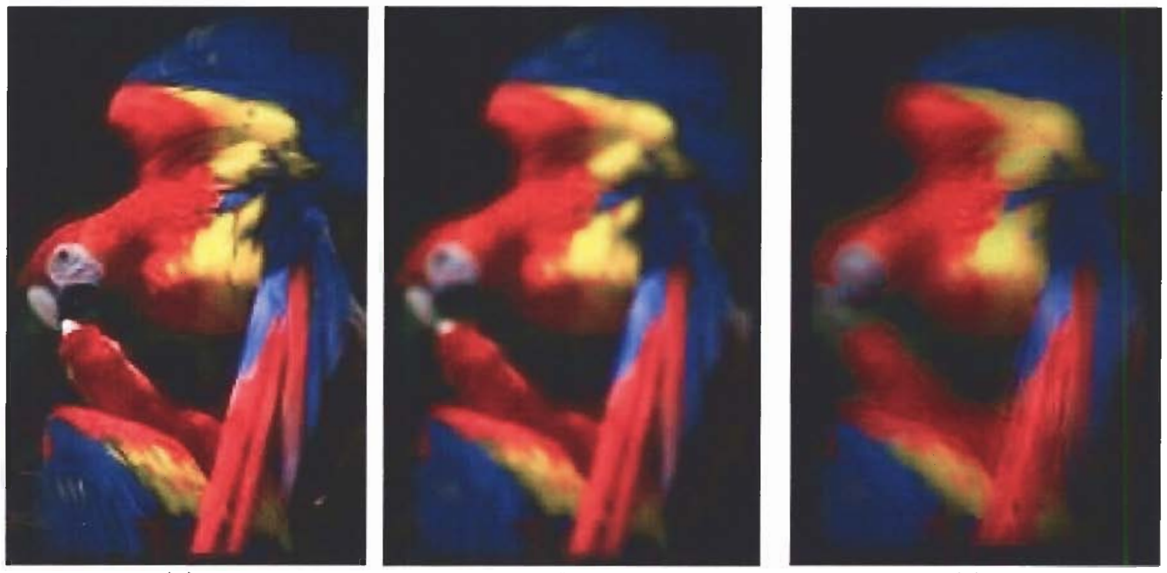


Figure 1: The result of colour sensitive smoothing with a 3x3 and 7x7 colour sensitive smoothing filter.
(a) - original image, (b) – adaptive colour sensitive filtered image with a 3x3 quaternion-valued mask, (c) – adaptive colour sensitive filtered image with a 7x7 quaternion-valued mask. Some edges that cross two colours are smoothed out, and some edges are preserved well, depending on the similarity of the two colours. (Image copyright © CorelGallery database by permission)

colours are more perpendicular than parallel to each other, the edge between them should be well preserved, because we only smooth the parallel component while preserving the perpendicular component. The result of this filtering is shown above and will be compared with the result of bilateral filtering later this section. As illustrated in Figure 1: , the red colour and the blue colour on the wings are more orthogonal than parallel, so the edge between them is better preserved. On the other hand, the red and the yellow colours on the wings have more parallel components, so the boundaries between them are blurrier. Also, on the head area of the bird, the black eye and the white face have only parallel colour component (both parallel to grey), therefore this region is smoothed the most. In addition, bigger mask generates a blurrier image than small mask yet still preserving edges and even enhancing, as observed in Figure 1: (b)-(c).

Additonally, this filter is only sensitive to the change of chromaticity, but much less sensitive to the colour difference caused by shadows and illumination, because the colours are normalized to the same intensity. Based on the property of the colour sensitive adaptive filter, we can create applications so that the edges of similar colours are smoothed, and preserve the edges across two colours that are dissimilar, i.e. colours that are perpendicular with each other. One particular filter derived from the colour sensitive adaptive filter is the adaptive bilateral filter that will be discussed in the following section.

2.3 Colour Edge Detection

Edge detection has been an interesting subject since the beginning of image processing. Lots of edge detection techniques on grey-scaled images have been proposed in the last decades. Now, it would be interesting to try the colour edge detection using quaternion based method. The quaternion-valued filters we have discussed above can also be modified to serve as colour edge detectors instead of the adaptive colour sensitive low-pass filter in the previous section. There are two approaches to achieve this edge detection task: one approach is based on quaternion rotation to cancel the chromaticity of two colours in RGB colour space [28, 30]; the other approach is based on the fact that colour edges only exist in a non-homogenous region [9].

2.3.1 Quaternion Rotation Edge Detection

The following pair of masks defines the new filter for detecting horizontal edges by Sangwine [30]**Error! Reference source not found.**. The vertical edges can be detected in the same manner by simply interchanging the rows and columns of the two masks.

$$\begin{bmatrix} R & R & R \\ 0 & 0 & 0 \\ \overline{R} & \overline{R} & \overline{R} \end{bmatrix} \times [\,] \times \begin{bmatrix} \overline{R} & \overline{R} & \overline{R} \\ 0 & 0 & 0 \\ R & R & R \end{bmatrix} \tag{15}$$

where [] denotes the image space for the pixels to be rotated, $\overline{R}$ denotes the conjugate of R, and R is defined as $R = [\cos(\pi/4) + \mu\sin(\pi/4)]/\sqrt{6}$. The quaternion operation $R\times[]\times\overline{R}$ defines a rotation about the axis μ (in RGB colour space) by an angle π/2 discovered by Hamilton. The axis μ, represented by a pure unit quaternion such that $\mu = \frac{i+j+k}{\sqrt{3}}$, has the same R, G and B values, so that it defines the grey line. All rotation with respect to μ is done in a plane perpendicular to the grey line. The top rows of the two masks rotate the pixel values by +π/2 angle, and the bottom rows rotate the pixels by –π/2 from their original positions.

Another simple filter for edge detection in a grey-scale image can be defined such that the top row is all ones, and the bottom row is all -1. This filter is introduced by the famous Prewitt filter for real valued image as we can see below. When this filter is applied, the pixels covered by the top row are unchanged, while the pixels under the bottom row are flipped by 180 degrees.

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

In the quaternion case, we can still apply the principle of the Prewitt filter, but in a slightly modified version. Since we have defined the quaternion rotation in RGB colour space, that the rotation operation $R\times[]\times\overline{R}$ rotates a pixel through an angle in a plane normal to the certain axis, we can create a quaternion-valued Prewitt filter for colour images. Again, we should convolve the mask on both sides of the image space. Therefore, the filter is defined as [28]:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ \overline{R} & \overline{R} & \overline{R} \end{bmatrix} \times[]\times \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ R & R & R \end{bmatrix}$$

where [] denotes the space for the pixels to be rotated, $\overline{R}$ denotes the conjugate of R, and R is defined as $R = [\cos(\pi/2) + \mu\sin(\pi/2)]/\sqrt{6}$, where $\mu = \frac{i+j+k}{\sqrt{3}}$ is a unit pure quaternion. Since the operation $R \times [] \times \overline{R}$ rotates a pixel by an angle π, the top rows of the two masks keep the pixels as where they are, and the bottom rows rotate the pixel values by 180 degree, on the plane perpendicular to the axis μ.

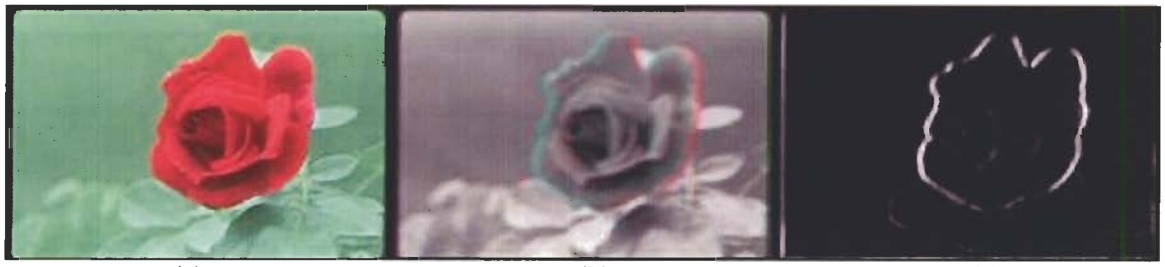


Figure 2: Quaternion edge detection by rotation
(a) the red flower input image (b) the result of the chromaticity cancellation by colour rotation. For regions not covering red edges, the effect of such rotation and cancellation causes achromatic pixel values close to the gray line, such as on the green leaves background areas. On the other hand, on the boundary of the red flower regions, the rotation generates colours other than gray. In this case, cyan and light purple on the left side and the right side around the flower boundaries, respectively. (c) the binary image representation of the detected edges based on the chromaticity information obtained in (b) (Image copyright © CorelGallery database by permission)

In image areas where the upper and lower row of the masks cover pixels of similar colours, the summation of the rotations of the upper and lower rows produces an achromatic pixel value on or near to the grey line, due to the "chromaticity cancellation". In contrast, if the pixel values under the upper and lower rows are different, i.e. the central pixel is on the edges, the addition of the rotations does not cancel in this chromatic sense completely, and hence the resulting pixel values are not close to the grey line and appear colourful. Based on Sangwine's method, I have tested this edge detection technique on colour images. Figure 2 has shown an example of the chromaticity cancellation method based on quaternion Prewitt filter for edge detection in a red flower picture. Figure 2 (b) shows the result of the chromaticity cancellation by colour rotation.

For regions not covering red-green edges, the effect of such rotation and cancellation causes achromatic pixel values close to the grey line, such as on the green leaves background areas. On the other hand, the colours on the boundary of the red flower regions are not grey, but cyan and light purple on the left side and the right side around the flower area. Figure 2 (c) shows the detected edges based on the chromaticity information in Figure 2 (b). A pixel is on an edge if its chromaticity value is greater than a threshold value. The chromaticity value for a pixel is calculated based on the equation c = (r+g+b)/3-min(r,g,b).

2.3.2 Edge Detection by Homogeneous Regions

Suppose we want to find the boundary of an object of a colour C in an image. The object boundary is made up of a set of 'boundary pixels'. Sangwine [30] has defined the *C-coloured homogenous region* in the following manner.

Definition: *A boundary pixel to a C-coloured region is a pixel that is not centred in a locally homogeneous, C- coloured region, but is 8-connected to a pixel that is centred within a locally homogenous, C-coloured region.*

The colour-sensitive smoothing mask can be used to determine if a pixel is centred within locally homogenous region, namely, the C-coloured region. Here, the colour sensitive mask defined in (**9** can be re-used.

If a pixel q of colour C is centred in a 3x3 homogeneous area, the convolution $U \otimes \text{Neigh}(q)$ yields a result with a negative scalar part and a zero vector part by Equation(10). In fact, if the whole 3x3 area contains quaternions only representing colour C, and by the fact that the dot product of C and C yields negative $|C|$, and the cross product of C and C yields zero, the expected result is precisely $(-|C|, 0)$. Therefore, a pixel on the boundary of a C-coloured region can be detected based on this convolution result. Note that this operation is only a one-sided convolution, either on the left side or the right side. More specifically, to decide whether a pixel is on the boundary of an object of colour *C*, the whole quaternion representation of the colour image is first normalized so that the convolution result at a pixel located completely inside a 3x3 homogenous *C*-coloured region should be (-1, 0). In practice, the colour-sensitivity of the edge detection procedure should be reduced by relaxing the definition of homogeneity. That is, instead of requiring a 3x3 region to contain 9 pixels of some colour that is exactly the colour *C*, we allow only most of the 9 pixels to be similar to colour *C*, such that the threshold is lowered. In this case, we define the threshold *t* for the detection by $t = |V| + |S|$. If a pixel is in the homogenous region, then $|S|$ should be close to 1 and $|V|$ should be close to zero. Thus, *t*, as the summation of $|S|$ and $|V|$, should be close to 1 as well. The choice of colour *C* can be either fixed throughout the entire image or adaptive based on the current pixel to be smoothed. For mask of a fixed colour *C*, as discussed in [9], it detects all the boundaries, where regions on one side of the boundary of colour similar to *C*. On the other hand, I have proposed an adaptive filter that changes *C* according to the centre pixel under the mask as it convolving on the image, it detects the boundaries of all homogenous regions of different colour.

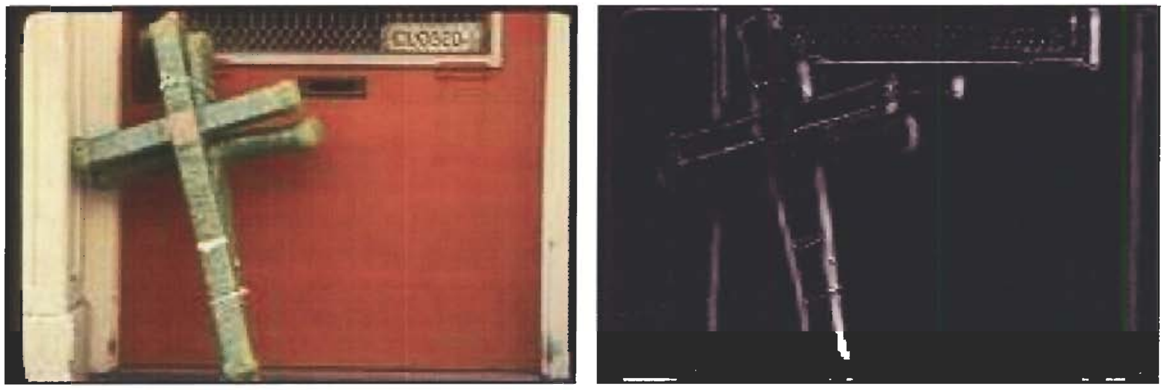


Figure 3: Edge detection by homogeneous regions
(a) the input image for edge detection by homogenous regions. (b) the detected colour edges in (a). The "adaptive" edge detector is used in this example, so that the colour edges caused by two arbitrary colours are identified rather than for edges of certain colour only, as proposed in [9] (Image copyright © CorelGallery database by permission)

Figure 3: illustrates edge detection using the adaptive filtering. The relaxed definition of a 3x3 homogeneous-coloured region is used here, with the threshold $t < 0.95$. The white pixels in Figure 3: (b) are identified as pixels on the boundary of two different colours, by convolving the natural image in Figure 3: (a) with the 3x3 mask U. However, many boundaries cannot be identified when strictly based on the definition of homogenous regions. By relaxing the definition of homogeneity as discussed above when searching for boundaries pixels, more detailed and finer edges can be detected(in Figure 3: (b)). In order to find only the edges of the green cross, for example, a filter of a fixed colour can be applied to Figure 3: (a) with $C = (19; 180; 33)$ (dark green). However, the edges of the door and the wall cannot be identified. Therefore this edge detection works when the edges are caused by a particular colour. Similar or better results may be obtained if quaternions are used to represent a colour space other than the RGB, or a different level of colour-sensitivity is set. However, our aim in Figure 3: is not to obtain optimal results, but just to demonstrate the results of the edge detection by homogeneity of regions.

2.4 Trilateral Filtering

2.4.1 Bilateral Filtering

Bilateral filtering was developed by Tomasi and Manduchi to smooth the noise in an image while preserving the edge information and is an alternative to anisotropic diffusion [32]. Bilateral filtering uses a nonlinear filter whose output is a weighted average of the input. The basis of the bilateral filter is a standard Gaussian filter represented by a spatial kernel function $f(x,y)$. However, the weight of a pixel is also affected by a function $g(x,y)$, defined in the intensity domain. The kernel function $g(x,y)$ lowers the weight of pixels with large intensity difference such that "different" neighbouring pixels do not have as much affect on the filtering as the "similar" pixels. Hence g is the function that is sensitive to the change of intensities and it functions similarly to an edge detection function. The shape of the mask changes according to the intensity difference of the central pixel to its neighbours. Note that due to the function g that is dependent on the local intensity information, it is not possible to perform the filtering in the frequency domain but only in spatial domain.

2.4.2 Quaternion Bilateral Filtering

As an adaptive filter, the bilateral filter redefines its shape (the weight of neighbouring pixels) based on the pixel intensities. The weight of the neighbouring pixels changes with respect to the similarity between each pixel and the central pixel. Based on the properties of the bilateral filter and the quaternion colour smoothing filter, a quaternion adaptive bilateral filter(the trilateral filter) can be defined. In this case, the output of the quaternion-valued bilateral filter for a quaternion representation of pixel $\vec{q}$ (see footnote 1) at location p is then:

$$\begin{aligned} \vec{q}_p' &= \frac{1}{k(p)} \sum_{s \in Neigh(p)} f(p-s) \cdot g(\vec{q}_p - \vec{q}_s) \cdot \vec{q}_s \\ &= \frac{1}{k(p)} \sum_{s \in Neigh(p)} f(\Delta x) \cdot g(\Delta \vec{d}) \cdot \vec{q}_s \end{aligned} \tag{16}$$

where k(p) is a normalization term based on the following equation:

$$k(p) = \sum_{s \in Neigh(p)} f(p-s) \cdot g(\vec{q}_p - \vec{q}_s) \tag{17}$$

both f and g are Gaussian functions defined as $f(\nabla x) = e^{\frac{-\nabla x^2}{2\sigma_f^2}}$ and $g(\nabla \vec{d}) = e^{\frac{-\nabla d^2}{2\sigma_g^2}}$. (Note that f and g can be functions other than Gaussian). Suppose σ_f and σ_g are two control parameters of the size and shape of the Gaussian kernels. Suppose $\vec{q}_p$ and $\vec{q}_s$ are the quaternion representation of pixel values at position p and s, respectively. Let Δx be the spatial Euclidian distance between

[1] $\vec{q}$ is the quaternion representation of the colour at a pixel p

pixel s and pixel p, i.e. $\Delta x = s - p \Rightarrow \| \Delta x \|^2 = \Delta x \cdot \Delta x$. Let Δd is the magnitude of difference between quaternion q_p and q_s, i.e. $\Delta \vec{d} = \vec{q}_p - \vec{q}_s \Rightarrow \| \Delta \vec{d} \|^2 = \Delta \vec{d} \times \Delta \vec{d}^*$.

In the previous section, I have designed a colour sensitive adaptive filter using quaternions. "Adaptive" here refers to adapting to the local colour, rather than adapting the current intensity information as with bilateral filtering. Therefore, it seems that the integration of the bilateral filter and the proposed adaptive colour smoothing provides a new way for smoothing colour yet preserving the edges. The adaptive colour filter can be applied to bilateral smoothing by modifying the quaternion mask U_q. The intuition is that in addition to the change of the shape of the mask while performing the convolution, the "colour" of the whole mask also changes depending on the colour of central pixel. By doing this, the filter smooths the component of the colours of an area in the direction parallel to the colour in the centre pixel. I design the adaptive bilateral filter based on the adaptive filter and bilateral filter by the following steps:

Firstly, we modify the weights in the mask of Equation(14) as:

$$\vec{q}_p = \frac{1}{2}[\sum_{s \in Neigh(p)} (f(s-p) \cdot \hat{q}_p) \times \vec{q}_s + \sum_{s \in Neigh(p)} \vec{q}_s \times (f(s-p) \cdot \hat{q}_p)] \cdot \hat{q}_p \tag{18}$$

to be a Gaussian functon. Then, we re-write Equation(16) as:

$$\vec{q}_p = \frac{1}{k(p)}[\sum_{s \in Neigh(p)} g(\vec{q}_s - \vec{q}_p) \cdot [f(s-p) \vec{q}_s] \tag{19}$$

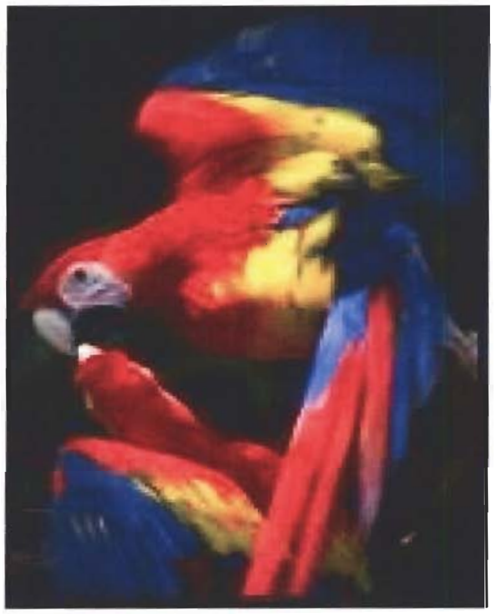
where k(p) is defined in Equation(17). Combine the above two equations, we have:

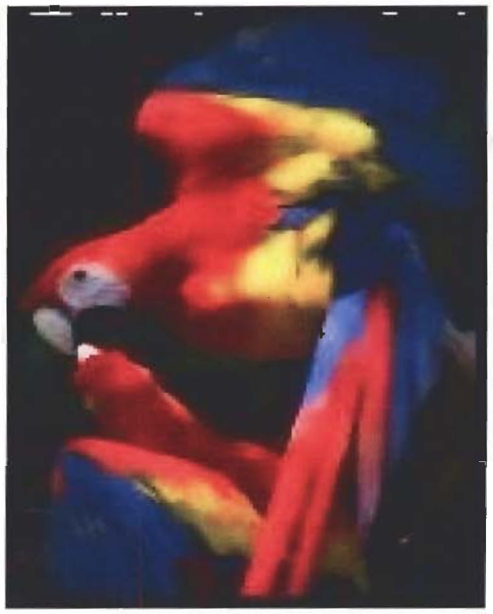
$$\vec{q}_p = \frac{1}{2k(p)} \sum_{s \in Neigh(p)} [g(\vec{q}_s - \vec{q}_p) \cdot f(s-p) \cdot \hat{q}_p] \times \vec{q}_s \cdot \hat{q}_p + \sum_{s \in Neigh(p)} \vec{q}_s \times [g(\vec{q}_s - \vec{q}_p) \cdot f(s-p) \cdot \hat{q}_p] \cdot \hat{q}_p \tag{20}$$

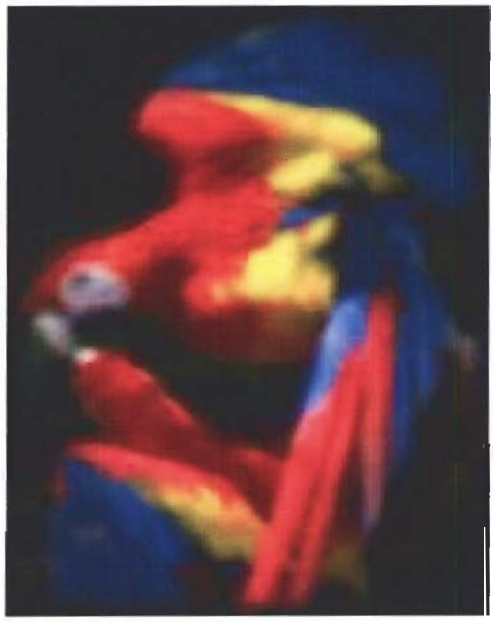
and, then the colour adaptive bilateral filter can be defined based on Equation(20):

$$\vec{q}_p = \frac{1}{2k(p)} \sum_{s \in Neigh(p)} g(\vec{q}_s - \vec{q}_p) \cdot f(s-p) \cdot (\hat{q}_p \times \vec{q}_s) \cdot \hat{q}_p + \sum_{s \in Neigh(p)} g(\vec{q}_s - \vec{q}_p) \cdot f(s-p) \cdot (\vec{q}_s \times \hat{q}_p) \cdot \hat{q}_p \tag{21}$$

I now demonstrate the trilateral filter, by comparing it with the smoothing results using bilateral filter on RGB channels separately, and the adaptive quaternion-valued filter discussed in the previous section. The bilateral filter is designed to smooth the pixels with similar colours while preserving the boundaries between different colours; the adaptive colour sensitive quaternion filter smooths only the parallel component of a particular colour while preserving the perpendicular component of that colour. Thus the adaptive bilateral filter, inheriting the property of both bilateral filter and colour sensitive filter, is expected to smooth similar pixel colours on their parallel components yet preserve the perpendicular colour component and the edges between two different colours. In Figure 5, the images show the result of the convolution with different filters. In Figure 5 (b), the bilateral filtering is applied to the RGB channels separately. The results show that the edges across different colour regions are clearly preserved, but not so for similar colours such as yellow and red, where the boundaries between them are blurred. Figure 5 (c) is a filtered image after applying the adaptive colour sensitive filter that preserves the edges across two colours that are orthogonal, for example, the edges of red and blue pair, and red and yellow pair. However, the whole figure of the wings on the black background is blurred because there is only the parallel component of the colour grey (black) so that the colour should be blurred out to the dark background. The proposed adaptive bilateral is designed to better preserve the edges and be less sensitive to shadows, having the properties of the other two filters as we can see in Figure 5 (d). Here, the kernel function *f(x,y)* computes the average of all neighbouring pixels, and the function *g(x,y)* computes the difference between two pixels specified in Equation(16).


(a)


(b)


(c)

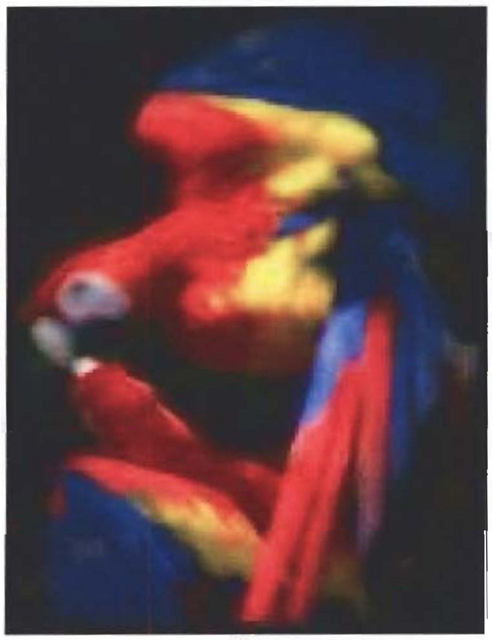

(d)

Figure 4: Illustration of the adaptive bilateral filter
(a) original image, (b) bilateral filtered image on 3 channels separately with $\sigma_f = 3$ and $\sigma_g = 3$, (c) adaptive filtered image with 3x3 quaternion-valued mask, (d) colour adaptive bilateral filtered image with filter size 3x3. (Image copyright © CorelGallery database by permission)

3. BQMP THRESHOLDING

3.1 Introduction BQMP

In this chapter we discuss the quaternion moment based operators, based on a thresholding technique called binary quaternion-moment-preserving (BQMP) thresholding. BQMP thresholding generalizes conventional grey-level moment-based operators [19, 20] into four-dimensions by expressing the input colour space as a quaternion-valued space. Through the definition of quaternion moments of input colour data, the moment preserving principle from 1-D grey-level data is extended to 3D colour data. An analytic solution for BQMP thresholding is also obtained by the use of quaternion arithmetic. The computation time for BQMP thresholding is of order N , the data size, so it is quite efficient. Moreover, this quaternion-moment-based technique can be applied to problems of colour image processing, such as image compression and image quantization and sharpening [19, 33].

3.2 Binary Quaternion-Moment-Preserving

The quaternion moments are designated as follows [20]:

$$\begin{aligned} \dot{m}_1 &= E[\dot{q}] \\ \dot{m}_2 &= E[\dot{q} \times \dot{q}^*] \\ \dot{m}_3 &= E[\dot{q} \times \dot{q}^* \times \dot{q}] \end{aligned} \tag{22}$$

where E[] represents the expectation. $E[q] = \frac{1}{N}\sum_{n=1}^{N} q(n)$ and q is a set of quaternions. Suppose m_1, m_2, m_3 are the 1st, 2nd and 3rd order moments of the set of quaternions, respectively. Here we only discuss up to the 3rd order moment because even though statistical distribution of a colour image can be described by knowing every order of moment, it is known that the higher the order of moment, the less important it is, because most energy is concentrated in the lower order moments. Preserving moments up to the third order will not severely affect the statistical distribution of the image, and will simplify the complexity of the algorithm and computations. Suppose we classify a quaternion valued data set into two clusters represented by $\dot{z}_0$ and $\dot{z}_1$, the means of these two clusters, with corresponding portion p_0 and p_1, such that all data above threshold belong to the cluster represented by $\dot{z}_0$ and all data below threshold belong to the cluster represented by $\dot{z}_1$. Based on Equation(22), we then represent the first three-quaternion moments of the two-level data sets $\dot{z}_0$ and $\dot{z}_1$ as [1]

$$\begin{aligned} \dot{m}'_1 &= p_0 \cdot \dot{z}_0 + p_1 \cdot \dot{z}_1 \\ \dot{m}'_2 &= p_0 \cdot \dot{z}_0 \times \dot{z}_0^* + p_1 \cdot \dot{z}_1 \times \dot{z}_1^* \\ \dot{m}'_3 &= p_0 \cdot \dot{z}_0 \times \dot{z}_0^* \times \dot{z}_0 + p_1 \cdot \dot{z}_1 \times \dot{z}_1^* \times \dot{z}_1 \\ 1 &= p_0 + p_1 \end{aligned} \qquad \textbf{(23)}$$

Now we have 4 quaternion-valued equations and 4 unknowns $\dot{z}_0, \dot{z}_1, p_0, p_1$. Additionally, the momentum m_1, m, and m_3 can be calculated by Equation(22), so this system of equations can be uniquely solved [20]. Hence, the two clusters represented by quaternion $\dot{z}_0$ and $\dot{z}_1$ are obtained.

3.3 BQMP Based Image Quantization

There are times that it is desirable to quantize the image so that the edges can be enhanced and sharpened. In Pei's [19] paper, the new moment-preserving thresholding technique, binary quaternion-moment-preserving thresholding, provides a new way to cluster colour image data in quaternion space. Based on preserving the quaternion moments of 4-dimensional input data, I defined an analytic and unsupervised two-class clustering classifier. In quantization, the input colour image is first divided into a number of sequential square blocks. Then for each block, the

[1] $\dot{m}_k$ denotes the kth momentum of a data set

BQMP thresholding method is used to segment the pixel block into two classes with a corresponding colour to represent each class. Since each block is represented by only two colours, only the strongest edge between the two colours is preserved or even enhanced; while the weaker edges are all flattened. Suppose we shift the position of all blocks and quantize the image this way again, each block may be represented by two different colours. If we keep shifting the blocks for a few runs and then we average all resulting images from each run, the result image should be an edge-enhanced image.

3.3.1 Algorithm

In summary, the procedures of the BQMP thresholding can be described as follows.

```
1) Divide the image into blocks

2) For each block, compute m1, m2, and m3 based on Equation(22)

3) Obtain two representatives, z0 and z1, of each block of data by solving
   the moment-preserving equations in Equation(23)

4) Make all pixels in clusters represented by z0 and z1 to have the mean
   colour of these pixels, correspondingly.

5) Shift the position of all blocks by n pixels on both the horizontal
   direction and the vertical direction, then follow the same step as (2)
   for N runs.

6) Average the result image from each run.
```

3.3.2 Results

The quantization result based on the binary quaternion-moment-preserving thresholding technique is illustrated in Figure 5. Figure 5 (b) shows the quantized image after one run of quantization of 16x16 block size. As mentioned in the Algorithm section above, all colours in each block are clustered z_0 and z_1. From Figure 5(b) we can tell that in each block, noise is smoothed while the edges are preserved and even enhanced. However, if a region of one colour covers more than one block, its quantized colour may be different, causing edges between blocks. The solution for reducing these unwanted edges across the blocks is to perform BQMP multiple times on overlapping blocks and average the result. Figure 5 (c) shows the result after 8 runs quantization of 16x16 block size. In each run, the positions of all blocks are shifted by a step = 2 pixels on both vertical and horizontal directions. Here, the edges between blocks are significantly reduced; noise is removed while edges are enhanced. For instance, the edges of the strips on the ball, and the edges of the ball with respect to the black background, are enhanced.

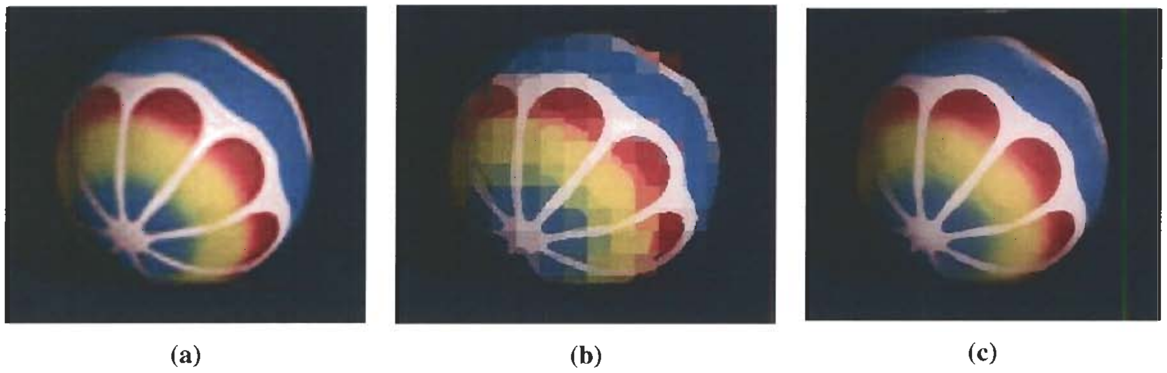


Figure 5: Illustration of the quantization process.
(a) Original blurred image; (b) the quantized image after one run of quantization with 16x16 block size; (c) 8 runs quantization with 16x16 block size, with shift step = 2 pixels on both vertical and horizontal directions (Image copyright © Computational Vision Lab by permission)

It has been shown that BQMP can be an effective tool for colour clustering. However, the number of clusters obtained by this method is limited to two. Suppose we want to have three clusters, *z0, z1* and *z2*, then there will be other three unknowns *p0,p1,* and *p2*. In order to solve for the system similar to Equation(23), we need to compute up to the 5th order of moment. Higher numbers of clusters are possible with this quaternion moment-preserving threshold method, by computing higher order of the momentum.

4. QUATERNION FOURIER ANALYSIS

4.1 Quaternion Fourier Transform

The 2D Fourier Transform is an important image processing tool to decompose a grey-scale image into its sine and cosine components. The output of the transformation represents the image in the Fourier or frequency domain, given the input image in spatial domain, where each "point" represents a particular frequency. The Fourier Transform can be used in a wide range of applications, such as image analysis, image filtering, image reconstruction and image compression. Fourier analysis on real and complex numbers had been well studied and widely used in numerous applications. Therefore, now it would be interesting to study the response of quaternion signals in the frequency domain, as many images processing method can perform more efficiently implemented in the frequency domain.

Based on the concept of quaternion multiplication and exponential, the Quaternion Fourier Transform (QFT) has been introduced[8]. Due to the non-commutative property of the quaternion, there are three different types of QFT defined: the left side QFT, the right side QFT and the two sides QFT. The details of the three types of QFT will be discussed in the following sections. The earliest definition of QFT is the two-side form as following[8]

$$H_q(w,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-iwx} \times h(x,y) \times e^{-jvy} dxdy \tag{24}$$

In fact, the QFT defined above can be generalized as [8]

- Two-Sides QFT: $H_{L-R}(w,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-\mu_1 wx} \times h(x,y) \times e^{-\mu_2 vy} dxdy$ **(25)**

where μ_1 and μ_2 are two unit pure quaternions (i.e., the quaternions with unit magnitude and zero real part) that are orthogonal to each other. i.e. $|\mu_1| = |\mu_2| = 1$ and $S[\mu_1 \times \mu_2] = -\mu_1 \cdot \mu_2 = 0$ ((24 is a special case of Equation(25) when $\mu_1 = i$ and $\mu_2 = j$)

More recently, the left-side the right-side form of QFT were defined in [27, 29] as

- Left-Side QFT: $H_L(w,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-\mu_1(wx+vy)} \times h(x,y)dxdy$

(26)

- Right-Side QFT $H_R(w,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(x,y)\times e^{-\mu_1(wx+vy)}dxdy$

(27)

Similarly, the **Inverse Quaternion Fourier Transforms (IQFT)** can be defined for the three types of DFT respectively as [27, 29]:

- Two-Sides IQFT $h(x,y) = \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-\mu_1 wx} \times H_{L-R}(w,v)\times e^{-\mu_2 vy}dwdv$

(28)

- Left-Side IQFT $h(x,y) = \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-\mu_1(wx+vy)} \times H_L(w,v)dwdv$

(29)

- Right-Side IQFT $h(x,y) = \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} H_R(w,v)\times e^{-\mu_2(wx+vy)}dwdv$

(30)

In the discrete case, the discrete quaternion Fourier transforms (DQFT) and discrete inverse quaternion Fourier transforms are also defined in these three types [29]. Since the discrete and continuous QFT and IQFT are basically the same, only the continuous case will be used to illustrate the QFT in this chapter.

4.2 Efficient Implementation of QFT

QFT and DQFT can be significant for quaternion-valued image processing, such as colour smoothing and data compression, because these analyses can be done much more efficiently in the frequency domain than in the spatial domain. However, the straight forward way based on the definition to compute the quaternion Fourier transforms is not practical, as we can see in the arithmetic formulas for quaternion exponential and quaternion multiplications, involving real number additions and multiplications that requires a great amount of computations. Fortunately, there are algorithms proposed to implement QFT by decomposing it into complex matrices on which standard FFT can be applied. In [6] and [8], Pei and Sangwine have discussed the efficient algorithms of all types of QFT to decompose the quaternion Fourier transform into several complex Fourier transforms, which can be computed individually using existing standard Fourier transform algorithms, such as the fast Fourier transform. Then the transforms are joined in a way that the desired quaternion Fourier transforms can be obtained. The inverse Fourier transform can be obtained in a similar manner with some modifications of this decomposition procedure. This decomposition technique solves the high computational cost problem doing the Fourier transforms for quaternion-valued matrices. The complexity of the fast quaternion Fourier transform will be analyzed in the following sections of this chapter.

4.2.1 Two-Sides QFT

First, lets study the two-sides QFT. To simplify the problem, proposed by Pei [18], we first discuss the special case when $\mu_1 = i$ and $\mu_2 = j$ as in Equation(24). Remember, based on Equation(25) we have

$$\begin{aligned} H_{L-R}(w, v) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-iwx} \times h(x, y) \times e^{-jvy} dxdy \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-iwx} \times h(x, y) \times (\cos(-vy) + j \cdot \sin(-vy))\, dxdy \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-iwx} \times h(x, y) \times \cos(vy)\, dxdy \\ &\quad + [\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-iwx} \times h(x, y) \times (i \cdot \sin(vy))\, dxdy] \times (-k) \end{aligned} \qquad (31)$$

where *h(x,y)* is a quaternion function in spatial domain and $H_{L\text{-}R}(w,v)$ is its type 1 QFT. Since $\cos(vy) = (e^{ivy} + e^{-ivy})/2$ and $i \cdot \sin(vy) = (e^{-ivy} - e^{ivy})/2$, substitute cosine and sine terms into Equation(31), we get

$$
\begin{aligned}
H_{L-R}(w,v) &= \tfrac{1}{2}\Big(\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-iwx} \times h(x,y) \times (e^{ivy} + e^{-ivy})dxdy \\
&\quad + \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-iwx} \times h(x,y) \times (e^{-ivy} - e^{ivy})dxdy \times (-k)\Big) \\
&= \tfrac{1}{2}\Big(\iint e^{-iwx} \times h(x,y) \times e^{ivy} + \iint e^{-iwx} \times h(x,y) \times e^{-ivy}\Big) \\
&\quad + \tfrac{1}{2}\Big(\iint e^{-iwx} \times h(x,y) \times e^{-ivy} - \iint e^{-iwx} \times h(x,y) \times e^{ivy}\Big) \times (-k) \\
&= \tfrac{1}{2}\Big(\iint e^{-iwx} \times h(x,y) \times e^{-iv(-y)} + \iint e^{-iwx} \times h(x,y) \times e^{-ivy}\Big) \\
&\quad + \tfrac{1}{2}\Big(\iint e^{-iwx} \times h(x,y) \times e^{-ivy} - \iint e^{-iwx} \times h(x,y) \times e^{-iv(-y)}\Big) \times (-k)
\end{aligned}
\qquad \textbf{(32)}
$$

Suppose we define $H_c(w,v)$ by

$$H_c(w,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-iwx} \times h(x,y) \times e^{-ivy}dxdy \qquad \textbf{(33)}$$

substitute Equation(33) into Equation(32) =>

$$H_{L-R}(w,v) = \tfrac{1}{2}[H_c(w,v) + H_c(w,-v)] + \tfrac{1}{2}[H_c(w,v) - H_c(w,-v)] \times (-k) \qquad \textbf{(34)}$$

Thus, we have decomposed the QFT into *Hc* representations. Furthermore, *Hc* is exactly the standard complex number Fourier transform by definition, and can be calculated by complex 2D FT method. The negative *v* in *Hc(w,-v)* means the reflection of the function *Hc(w,v)* with respect to *w* axis. For 2D images, such reflection is the same as flipping the image with respect to the y-axis (vertical axis). To compute *Hc(w,v)*, we first need to decompose *h(x,y)* into a complex pair representation:

$$
\begin{aligned}
&h(x,y) = h_a(x,y) + h_b(x,y) \times j \\
&\text{where } h_a(x,y) = h_r(x,y) + h_i(x,y) \cdot i, \ \text{ and } h_b(x,y) = h_j(x,y) + h_k(x,y) \cdot i
\end{aligned}
\qquad \textbf{(35)}
$$

Note that h_a is parallel to i and $h_b \times j$ is perpendicular to i.[1] (see Thm 4.1)

Therefore, based on Equation(33) and Equation(35), Hc can be decomposed as:

$$\begin{aligned}
H_c(w,v) &= \iint e^{-iwx} \times h(x,y) \times e^{-ivy} dxdy \\
&= \iint e^{-iwx} \times h_a(x,y) \times e^{-ivy} dxdy + \iint e^{-iwx} \times (h_b(x,y) \times j) \times e^{-ivy} dxdy \\
&= \iint e^{-iwx} \times h_a(x,y) \times e^{-ivy} dxdy + \iint e^{-iwx} \times (h_b(x,y) \times (j \times e^{-ivy}))dxdy \\
&= \iint e^{-iwx} \times h_a(x,y) \times e^{-ivy} dxdy + \iint e^{-iwx} \times (h_b(x,y) \times (j \times \cos(-vy) + j \times i \times \sin(\\
&= \iint e^{-iwx} \times h_a(x,y) \times e^{-ivy} dxdy + \iint e^{-iwx} \times (h_b(x,y) \times (\cos(vy) \times j + i \times \sin(vy) \times \\
&= \iint e^{-iwx} \times h_a(x,y) \times e^{-ivy} dxdy + \iint e^{-iwx} \times (h_b(x,y) \times e^{ivy} \times jdxdy \\
&= \iint e^{-iwx} \times h_a(x,y) \times e^{-ivy} dxdy + [\iint e^{-iwx} \times (h_b(x,y) \times e^{-iv(-y)} dxdy] \times j \\
&= \iint e^{-iwx} e^{-ivy} h_a(x,y) dxdy + [\iint e^{-iwx} e^{-ivy} h_b(x,-y) dxdy] \times j \\
&= FT(h_a) + FT(h_b^{'}) \times j
\end{aligned}$$

(36)

Then, *Hc* can be represented as an integration of *FT(ha),* and the product of *FT(hb')* and *j*, where *h_b'* represents the flipped version of *$h_b(x,y)$* with respect to x-axis. According to Equation(36), *Hc* can be computed based on two complex 2D FTs, *FT(ha)* and *FT(hb).* Once *Hc* is solved, $H_{L-R}(w,v)$ can be computed by Equation(34). The complicated QFT problem has been simplified into much simpler and more efficient one by solving complex FT sub-problems.

In summary, the algorithm to implement Two-Side Quaternion Fourier Transform can be broken down into the following steps [18]:

1) Decompose the input quaternion-values function into h_a and h_b based to Equation(35)

2) Calculate H_c based on h_a and h_b by Equation(36)

3) Calculate the result QFT based on H_c by Equation(34)

[1] $h_r(x,y), h_i(x,y), h_j(x,y)$ and $h_k(x,y)$ are the real part, i-part, j-part and k-part of the quaternion h(x,y), respectively

Remember the above algorithm only deals with the special case when $\mu_1 = i$ and $\mu_2 = j$. For the general case when μ_1 and μ_2 are arbitrary pure unit orthogonal quaternions, we need to modify the above algorithm. The idea is that, instead of using *i, j* and *k* to represent a 3 dimensional variable, we change the basis to μ_1, μ_2 and μ_3, respectively (given μ_1, μ_2 and μ_3 have unit length and are orthogonal to each other). For example, $x = x_1\hat{i} + x_2\hat{j} + x_3\hat{k} = x_1' \cdot \hat{\mu}_1 + x_2' \cdot \hat{\mu}_2 + x_3' \cdot \hat{\mu}_3$. So the transformed variable is equivalent to the original variable, in 4D space, with different bases. The decomposition of *h(x,y)* in Equation(35) should be modified correspondingly. The following theorem is provided by Pei [18]:

Theorem 4.1. *Given a pure quaternion u, and a second pure unit quaternion v, u may be decomposed into components parallel and perpendicular to v as*

$$u_{para} = \frac{1}{2}(u - v \times u \times v)$$
$$u_{perp} = \frac{1}{2}(u + v \times u \times v)$$
such that $(u_{para}) \parallel v,\ (u_{perp}) \perp v$ and $u = u_{para} + u_{perp}$

Proof. (Omitted)

According to Theorem 4.1, the parallel and perpendicular decomposition of function *h(x,y)* with respect to μ_1 can be performed as follows:

$$\begin{aligned} h_{\parallel}(x,y) &= h_r(x,y) + h_1(x,y) \times \mu_1 \\ h_{\perp}(x,y) &= h_2(x,y) + h_3(x,y) \times \mu_1 \end{aligned} \quad \text{and} \quad \begin{bmatrix} h_1(x,y) \\ h_2(x,y) \\ h_3(x,y) \end{bmatrix} = \begin{bmatrix} \mu_{1,i} & \mu_{2,i} & \mu_{3,i} \\ \mu_{1,j} & \mu_{2,j} & \mu_{3,j} \\ \mu_{1,k} & \mu_{2,k} & \mu_{3,k} \end{bmatrix} \begin{bmatrix} h_i(x,y) \\ h_j(x,y) \\ h_k(x,y) \end{bmatrix} \quad \textbf{(37)}$$

such that $h(x,y) = h_{\parallel}(x,y) + h_{\perp}(x,y) \times \mu_2$, given that μ_1, μ_2 and μ_3 are unit, pure and orthogonal. The following theorem is provided by Pei [18]

Theorem 4.2. *Given a quaternion function h, and two pure unit orthogonal quaternions* μ_1 and μ_2 *, if we can decompose h into two complex functions* $h_{\parallel}$ and $h_{\perp}$, *such that* $h_{\parallel} + h_{\perp}$ $(h_{\parallel}) \parallel \mu_1$, $(h_{\perp} \times \mu_2) \perp \mu_1$ and $h = h_{\parallel} + h_{\perp} \times \mu_2$, *then we have*

$e^{\mu_1} \times h_{\parallel} = h_{\parallel} \times e^{\mu_1}$ and $e^{\mu_1} \times h_{\perp} = h_{\perp} \times e^{-\mu_1}$

Proof.(Omitted)

Based on Theorem 4.2, we can change the basis to μ_1, μ_2 and μ_3, and Equation(36) becomes

$$H_c(w,v) = FT(h_{\parallel}) + FT(h_{\perp}^{'}) \times \mu_2 \tag{38}$$

Now, suppose we have three unit pure quaternions μ_1, μ_2 and μ_3 that are orthogonal to each other. Then we can implement the general Two-Side QFT as [18]:

1) First, decompose function h(x,y) as $h_{\parallel}(x,y)$ and $h_{\perp}(x,y)$ using Equation(37)

2) Then calculate Hc based on Equation(38)

3) Then calculate the transform result of DFT based on Equation(32) by

$$H_{L-R}(w,v) = \tfrac{1}{2}[H_c(w,v) + H_c(w,-v)] + \tfrac{1}{2}[H_c(w,v) - H_c(w,-v)] \times (-\mu_3)$$

Therefore the Two-Sides QFT can be implemented by two MxN 2D discrete FTs. Since each 2D discrete FT requires MNlog2(MN) real number multiplications [8], the amount of real number multiplications required for implementation is 2MNlog2(MN). Similarly, inverse Two-Sides quaternion Fourier transform (IQFT) can be calculated in the same manner, with *h(x,y)* and *H(w,v)* exchanged and *FT(h)* replaced with *IFT(h).* Hence, the computational cost of Two-Side IQFT is also 2MNlog2(MN).

With the fast QFT and IQFT, a colour image can be transformed into frequency space efficiently. The easiest way to determine the frequency composition of signals is to inspect them in their frequency domain. Now let's study the frequency response of a 2D colour image in the Fourier domain. For standard Fourier transform, the frequency domain image shows the magnitude of different frequency components. For quaternion Fourier transform, the frequency domain is also represented by quaternions of different frequency components. A simple example is given below to illustrate the quaternion Fourier transforms by generating cosine waves (Figure 6:). Suppose in the frequency domain of a 256x256 image, a pixel q at location (200,200) is set to be a particular quaternion value. We would like to study the relationship between the frequency response of a single point and its image in spatial domain. Figure 6: illustrates the images of different frequency responses by different q values. The images are all cosine waves of different colours. The colours in each subfigure are the result of the interaction between each component, so there is no obvious correspondence of the frequency q and the colour in spatial domain.

Another example is included to show the effectiveness of the Type 1 QFT by filtering the colour image with a low pass filter (see Figure 7). Figure 7 (b) only displays the imagery parts as a colour image, even though the entries in Fourier image are not necessarily pure quaternions. The low-pass filtered image in Fourier domain is shown in Figure 7(c) where the high frequency signals are around the centre and low frequency signals are around the corners. If we take IQFT of (c), we obtain a blurred version of the original image. This should have the same effect as convolving the image with a low pass filter. I believe using quaternion allows us to have a better colour preservation because the three colour channels are processed as a single unit.

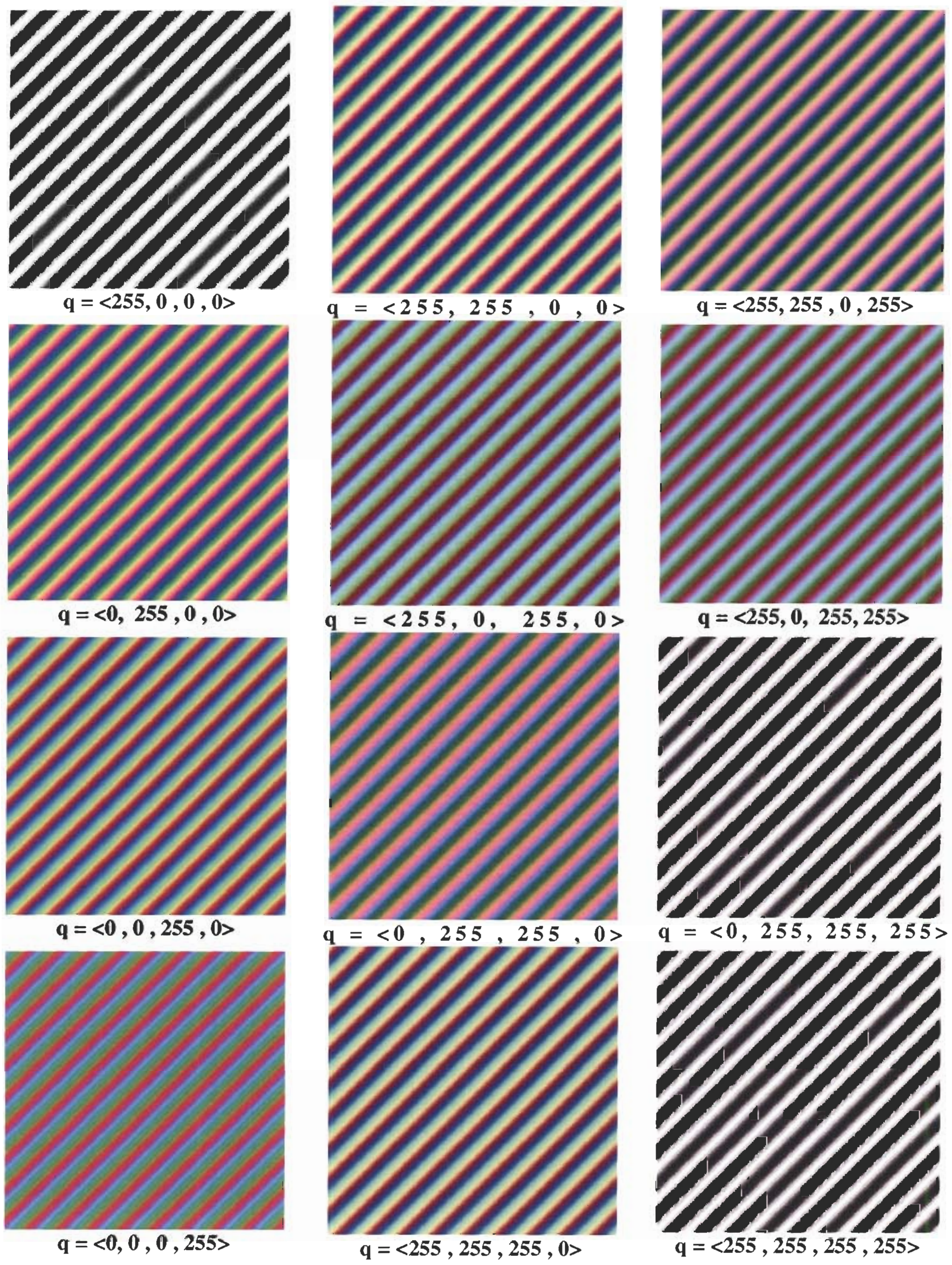


Figure 6: Quaternion cosine waves generated from a quaternion pulse in frequency domain. Only imagery components are displayed as a colour image, even though the entries in Fourier image are not necessarily pure quaternions. The two axis are the frequencies in both x and y directions.

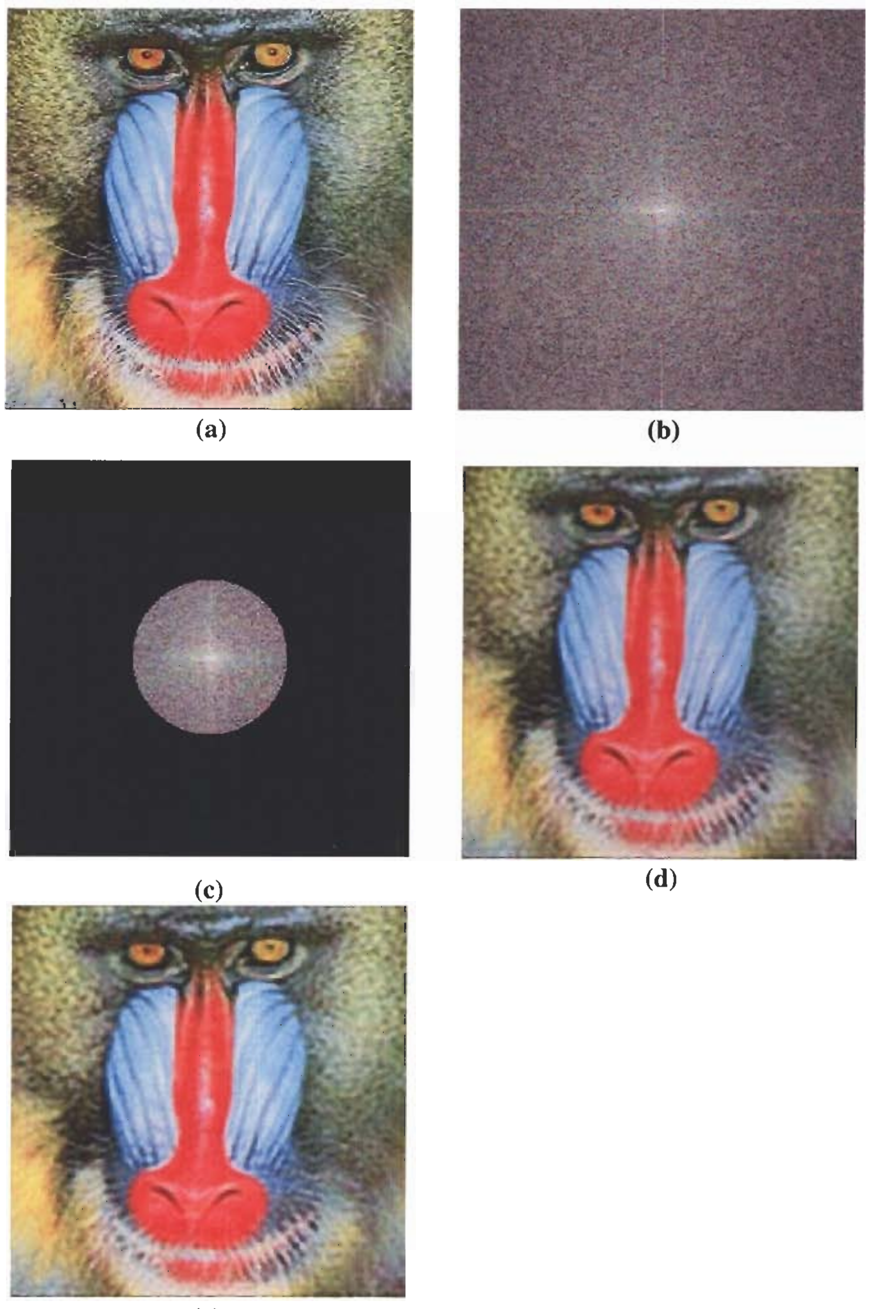

(a) (b) (c) (d) (e)

Figure 7: The result of low-pass filtering based on type 1 QFT.
(a) the original image (b) the result of Two-Sides QFT, with the axis μ1=[1 1 1]/√3 and μ2=[0 1 -1]/√2, the real parts of all frequencies are not displayed (c) The low quaternion frequencies are cut off by a mask with Radius = 50. (d) the type 1 inverse QFT image, with details and noises removed. (e) the result of process RGB three channels separately(Image copyright © CorelGallery)

4.2.2 Left-Side/Right-Side QFT

I now review the efficient algorithms for Left-Side and Right-Side QFT. The reason we define three types of QFT is the non-commutativity of quaternions. However, conceptually, the conclusion cannot be drawn whether one type of QFT is better or more useful than the others. The derivations of the algorithms are quite similar to but simpler than the Two-Sides QFT. That is, we still use the idea of decomposing the quaternion functions into several complex function representations, and then integrate the Fourier transform of those complex functions in a specific way. Based on Equation(36) , we need to decompose *h(x,y)* into its parallel and perpendicular components with respect to the Fourier axis μ:

- **For Left-Side QFT**:

$$\begin{aligned} H_L(w,v) &= \iint e^{-\mu_1(wx+vy)} \times h(x,y)dxdy \\ &= \iint e^{-\mu_1(wx+vy)} \times (h_{\parallel}(x,y) + h_{\perp}(x,y) \times \mu_2)dxdy \\ &= \iint e^{-\mu_1(wx+vy)} \times h_{\parallel}(x,y)dxdy + \iint e^{-\mu_1(wx+vy)} \times h_{\perp}(x,y) \times \mu_2 dxdy \\ &= FT(h_{\parallel}) + FT(h_{\perp}) \times \mu_2 \end{aligned} \quad \textbf{(39)}$$

- **For Right-Side QFT:**

$$\begin{aligned} H_R(w,v) &= \iint h(x,y) \times e^{-\mu_1(wx+vy)}dxdy \\ &= \iint (h_{\parallel}(x,y) + h_{\perp}(x,y) \times \mu_2) \times e^{-\mu_1(wx+vy)}dxdy \\ &= \iint h_{\parallel}(x,y) \times e^{-\mu_1(wx+vy)}dxdy + \iint h_{\perp}(x,y) \times \mu_2 \times e^{-\mu_1(wx+vy)}dxdy \\ &= \iint e^{-\mu_1(wx+vy)} \times h_{\parallel}(x,y)dxdy + \iint h_{\perp}(x,y) \times e^{\mu_1(wx+vy)} \times \mu_2 dxdy \,(\text{By Thm } 4.2) \\ &= \iint e^{-\mu_1(wx+vy)} \times h_{\parallel}(x,y)dxdy + \iint e^{\mu_1(wx+vy)} \times h_{\perp}(x,y) \times \mu_2 dxdy \\ &= FT(h_{\parallel}) + IFT(h_{\perp}) \times \mu_2 \end{aligned} \quad \textbf{(40)}$$

The algorithm to implement the Left-Side QFT and Right-Side QFT, can be summarized in following steps.

1) First, decompose h(x,y) as $h(x,y)=h_{\parallel}(x,y)+h_{\perp}(x,y)\times\mu_2$ by the same method of Two-Side QFT.
2) Use Equation(39) and Equation(40) to calculate Left-Side QFT and Right-Side QFT, respectively.

Thus, we need two complex 2D FTs for Left-Side QFT, and one 2D FT plus one 2D IFT for Right-Side QFT. The amounts of real multiplications required for Two-Sides QFT is 2MNlog$_2$(MN). Similarly, Left-Side and Right-Side IQFT can also be calculated in the same manner, with *h(x,y)* and *H(w,v)* exchanged and *FT(h)* and *IFT(h)* switched. Hence, the computational cost of Left-Side and Right-Side IQFT is also 2MNlog$_2$(MN).

4.3 Quaternion Correlation

Measuring the correlation between two signals is an approach for feature detection, or as a component of more sophisticated techniques. Cross-correlation is well-known for its role in template matching procedures[3, 5]. If the shape of a target is made into a kernel whose values are multiplied by the pixel values at every location in the image and then normalized for the absolute values of the pixels, the largest response will result when the grey image contains the same pattern of grey scale values. Although this operation can be done in spatial domain, it will be more efficiently performed in the Fourier domain by multiplying the transforms of the target and the image and performing an inverse transformation. Textbook presentations of correlation describe the convolution theorem in the frequency domain using the fast Fourier transform. Especially if the convolving mask is large, it is more efficient to do the correlation in Fourier domain. With either implementation, the locations in the image that match the target are found efficiently even in the presence of noise, shading, or partial obscuration of the target.

The cross-correlation functions have recently been generalized to colour images based on quaternions [7, 24]. However, direct evaluation of the summation over the whole image for each pixel results in a Q(N^4) computational cost. This is not practical for all but only when one of the correlating images is fairly small. Therefore, a practical correlation method requires the utilization of Fourier transform as we have discussed in the previous section. To have a better understanding of the relationship between the cross-correlation and the Fourier transform, we will review the Wiener-Khintchine theorem for complex numbers and extend the Wiener-Khintchine theorem for quaternions.

4.3.1 Cross-Correlation in Spatial Domain

Suppose the cross-correlation *c* of two complex functions *f(m,n)* and *h(m,n)* is defined by

$$c = f \circ h = f(x,y) \otimes \overline{h(-x,-y)} \tag{41}$$

where $\circ$ denotes the correlation operation and $\otimes$ denotes the convolution operation. The complex function $\overline{h(-x,-y)}$ is the conjugated result of reflection of function *h(x,y)* with respect to x-axis and y-axis. The convolution of function *f(x,y)* and *h(x,y)* is

$$con(x,y) = f(x,y) \otimes h(x,y) = \iint f(x-\tau, y-\eta) \times h(\tau,\eta) d\tau d\eta$$

and therefore the correlation is defined as

$$c(x,y) = \int\int_{-\infty}^{\infty} f(x+\tau, y+\eta)\overline{h(\tau,\eta)} d\tau d\eta .$$

Similarly, for two quaternion-valued functions *f(x, y)* and *h(x,y),* the quaternion convolution *con(x,y)* is defined as

$$con(x,y) = f(x,y) \otimes h(x,y) = \iint f(x-\tau, y-\eta) \times h(\tau,\eta) d\tau d\eta \tag{42}$$

Based on the extended Wiener-Khintchine theorem for quaternions, the cross-correlation *c(x,y)* is defined as follows in [8]

$$c(x,y) = \int\int_{-\infty}^{\infty} f(x+\tau, y+\eta) \times \overline{h(\tau,\eta)} d\tau d\eta$$

And in discrete case, it is defined in [24] as

$$C(m,n)=\sum_{p=0}^{M-1}\sum_{q=0}^{N-1}f(p,q)\overline{h(p-m,q-n)} \qquad \textbf{(43)}$$

One of the remarkable applications of cross-correlation in image processing is in template matching, where a signal is searched for the existence of some pattern [5]. Given a test image, we are interested in finding the location of the template within this image. Figure 8 illustrates the target searching by quaternion cross-correlation according to Equation(43) in the spatial domain. The target or model is a coloured capital letter "A", and the test image contains different letters of various colours. The task is to detect the location of the pattern, which has the greatest similarity to the target in the sense of both structure and colour. Figure 8(c) shows the result of the cross-correlation of (a) and (b) represented by the magnitude of the quaternion at each pixel. The magnitude represents the likelihood that the target is detected at this location. On row 2 column 6 of Figure 8(c), the blue letter A corresponds to the highest peak because it is the most similar to the template. On row 7 column 8 and row 8 column 2, the cyan letter A and purple letter A have the 2nd and 3rd highest peaks, respectively, because both colours have blue component. This quaternion cross-correlation algorithm successfully identifies the candidate locations of the model in the test image by estimating the pattern and colour information as a whole unit.

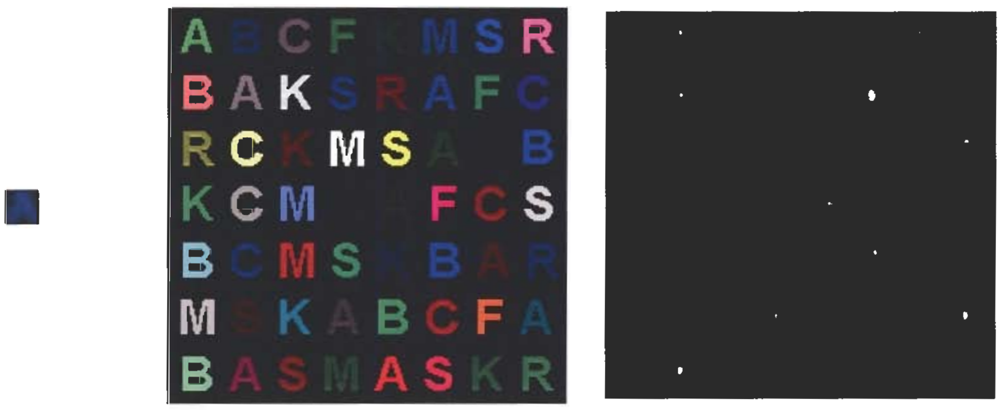


Figure 8: Colour template matching by cross-correlation in spatial domain
(a) the target navy letter A. (b) the test image containing a number of different letters and colours. (c). The cross-correlation of the images in (a) and (b). The peaks indicate the possible locations where the patterns and colours are mostly similar to the navy letter A.

4.3.2 Cross-Correlation in Frequency Domain

For the conventional convolution operation, if *con(t)* is the convolution result of real or complex functions *f (t)* and *g(t)*, then *con(t)* can be calculated based on the Equation(44)

$$con(t) = IFT(FT(f(t)) \cdot FT(g(t))) \tag{44}$$

Therefore, we can calculate the conventional convolution by the efficient algorithms of the complex FT and IFT. The correlation, on the other hand, can also be implemented as [8]

$$c(t) = IFT(FT(f(t)) \cdot \overline{FT(g(-t))}) \tag{45}$$

It is reasonable to seek for simple relations between the QFT and the quaternion correlation, so that we can use efficient QFT algorithms to implement correlation in frequency domain rather than in spatial domain.

In the previous section, we have shown that a quaternion can be decomposed into two orthogonal components with respect to a pure unit quaternion (axis), one parallel to this axis and the other perpendicular to this axis. In addition, the QFT itself can be decomposed into components parallel to and perpendicular to this axis. In this section, Wiener-Khintchine theorem is extended to quaternion space for colour image correlations[7].

Theorem 4.3 *Based on the Right-Side QFT/IQFT, a quaternion generalization of the Wiener-Khintchine theorem is as follows:*

$$C(m,n) = IFT[F(v,u) \times \overline{G_{para}(u,v)}] + FT[F(v,u) \times \overline{G_{perp}(u,v)}]$$

where

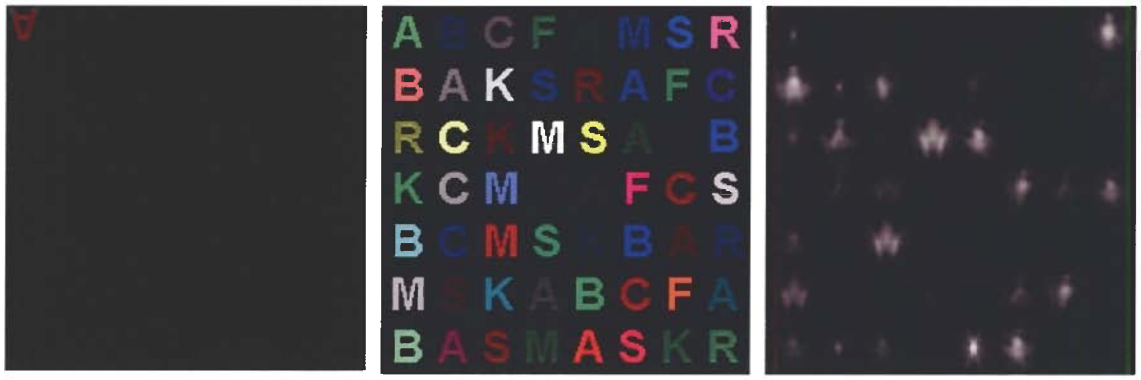
$$F(u,v) = FT(f(u,v)) \text{ and } G(u,v) = FT(g(u,v))$$
$$G_{para}(u,v) \parallel \mu \text{ and } G_{perp}(u,v) \perp \mu, \text{ and}$$
$$\mathrm{G(u,v)} = G_{para}(u,v) + G_{perp}(u,v)$$

Proof(Omitted)

A generalization of the Wiener-Khintchine theorem to quaternion images requires quaternion Fourier transforms that have been discussed and implemented in the previous section. Suppose *F(u,v)* and *G(u,v)* are the Fourier transform of quaternion functions *f(u,v)* and *g(u,v)*, respectively. Basically, this theorem requires decomposing the Fourier spectrum *G(u,v)* of quaternion function *g(u,v)* into the parallel and perpendicular components with respect to the Fourier transform axis μ . Then we combine the inverse Fourier transform of the product of the *F(u,v)* and the conjugation of the parallel component of *G(u,v)*, with the Fourier transform of the product of the *F(u,v)* and the conjugation of the perpendicular component of *G(u,v)*. The result is the cross-correlation of two quaternion-valued vectors.

Although all the three types of QFT/IQFT we have discussed in the previous section can be possibly used in correlations, the algorithms that have been studied in [8] to implement the generalized Wiener-Khintchine theorem is only defined on Right-Side QFT/IQFT. In addition, as suggested by Pei [8], the Right-Side QFT based quaternion correlation is simpler and easier to analyze. Therefore, in this thesis, the Right-Side QFT is adopted to analyze the quaternion cross-correlation.

The Figure 10 shows the template matching by quaternion cross-correlation in the frequency domain based on Theorem 4.3. The target is a dark red coloured capital letter "A", and the database image contains different letters of various colours. The task is to detect the location of the pattern, which has the greatest similarity to the template in the test image. The letter "A" is is flipped 180 degree in both the vertical and horizontal directions. This flipping is made corresponding to the efficient cross-correlation when we need to convolve with the function *g(-x,-y)*(suppose *g(x,y)* represents the letter A). Zeros are appended to function *g(x,y)* so that the images in Figure 10 (a) and Figure 10 (b) have the same size. Figure 10 (c) shows the result of the cross-correlation of (a) and (b) represented by the magnitude of the quaternion at each pixel. The magnitude represents the likelihood that the target is detected at this location. On row 8 column 5



(a) (b) (c)

Figure 9: The template matching based on quaternion cross-correlation in frequency domain (a) the target dark red letter A. (b) the database image containing a number of different letters and colours. (c). The cross-correlation of the images in (a) and (b). The peaks indicate the possible locations where the patterns and colours are mostly similar to the dark red letter A.

of Figure 10 (c), the red letter A corresponds to the highest peak because it is most similar to the target, based on the structural and colour information in combination. This quaternion cross-correlation algorithm successfully identifies the candidate locations of the template by estimating the pattern and colour information as a whole unit. This cross-correlation in Fourier domain can be computed much more efficiently than in the spatial domain, yet generates the same matching result.

4.4 Quaternion Convolution

Quaternion based image convolution has been introduced in Chapter 2 as a tool for colour image filtering (see (8). Here, we will focus on the one-sided quaternion convolution rather than the two-sided one, and the theoretical background of quaternion convolution will be discussed in both spatial and frequency domains.

4.4.1 Convolution in Spatial Domain

The definition of one side quaternion convolution in spatial domain has been given in Equation(42). The operations required for convolution are only multiplication, conjugation and addition. Furthermore, the multiplication of two pure quaternions involves only dot product and cross product. For example, when an image is convolved with a quaternion valued filter, every pixel under the filter is multiplied with the corresponding pixel in the image, and then the result is summed. More specifically, the average of negative of the dot product for each pixel pair is saved in the real part of the convolution result; and the average of the cross product for each pixel pair is saved in the three imaginary parts. In this case, the real "layer" of the quaternion convolution result contains the summation of the convolution on each RGB channel separately.

4.4.2 Convolution in Frequency Domain

Based on the relationship between complex convolution and cross-correlation in Equation(41), we have $f \otimes h = f(x, y) \circ \overline{h(-x,-y)}$, where *f* and *h* are two quaternion-valued functions. Suppose we want to convolve two colour images *f* and *h*. This can be achieved by using the quaternion correlation method in Equation(45) with the negated image *h* flipped on both horizontal and vertical directions. With the quaternion convolution in frequency domain, many image filtering processes can be done in a much more efficient manner. Moreover, the colour-sensitive filtering we have studied in Chapter two can be performed this way by integrating the result of the convolution on the left side and on the right side separately.

5. QUATERNION WAVELET

5.1 Quaternion Harr Wavelet Transform

One of the key features of the Fourier transform is that it allowed us to decompose a signal into a range of frequencies and then analyse the signal one frequency at a time. Similarly, wavelets are a class of functions used to localize a given function in both translating and scaling. A family of wavelets can be constructed from a function, which is sometimes known as a "mother wavelet". In this chapter, an introduction to the simple quaternion wavelet transform is given based on the Harr function.

To wavelet transform a vector of real numbers, the first step is to separate the high and low frequencies. We can use a lowpass filter l and highpass filter h to define a decomposition of a signal into a lowpass and highpass components. Suppose we have a vector X of length N, then we convolve it with the low pass and high pass filter such that

$$l * X(r) = \frac{X(r) + X(r-1)}{2} \bmod N$$
$$h * X(r) = \frac{X(r) - X(r-1)}{2} \bmod N$$

We call $l*X$ the smooth approximation to X and $h*X$ the high resolution details of X. Then if we throw away every second component of $l*X$ and $h*X$, we get two vector C_l and C_h, such that the vector X can be reconstructed from C_l and C_h:

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

where the operator $\oplus$ is a "shuffle" operator that alternatively takes one element from one of the two vectors at a time and appends this element to the end of the result vector.

Similarly, in the quaternion case, for a quaternion-valued vector Q, we define the lowpass and highpass filter so that

$$l \times Q(r) = \frac{Q(r)+Q(r-1)}{2} \bmod N$$
$$h \times Q(r) = \frac{Q(r)-Q(r-1)}{2} \bmod N$$

In the 2D quaternion wavelet transform, we still let *l* and *h* be the low pass and high pass filters and let Q be an MxN quaternion matrix, where we assume both M and N are divisible by reasonably large powers of 2. Then we can do the quaternion wavelet transform on columns first followed by the rows. The information stored in a quarter of the result after one-stage of the wavelet transform is shown in the table

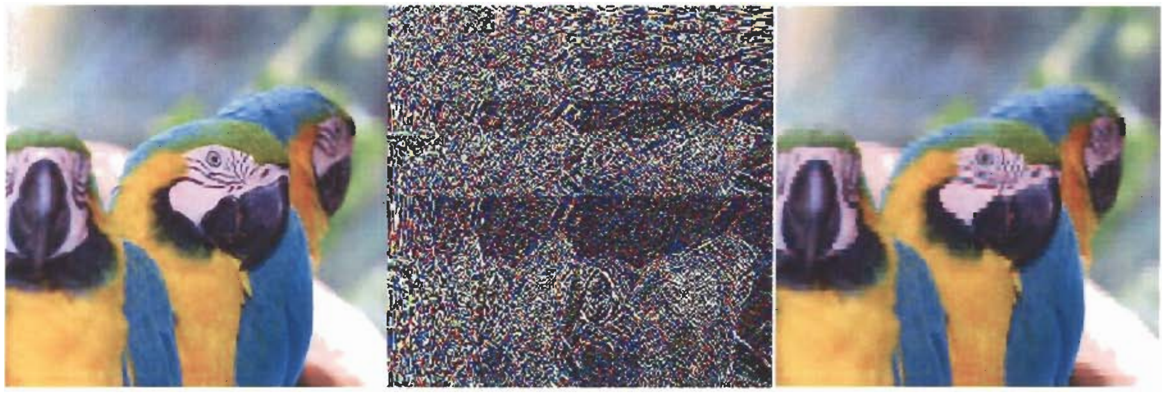
$$\begin{array}{c|c|c|} & \text{low} & \text{high} \\ \hline \text{low} & LL & HL \\ \hline \text{high} & LH & HH \\ \hline \end{array} = \begin{array}{c|c|c|} & \text{low} & \text{high} \\ \hline \text{low} & \text{approximation} & \text{vertical details} \\ \hline \text{high} & \text{horizontal details} & \text{diagonal details} \\ \hline \end{array}$$

However, the operator + and – for quaternion arithmetic is exactly the same as adding/subtracting the real and each imaginary components separately. In addition, scaling by a real number ½(or by a quaternion *s=<1/2,0,0,0>*) is equivalent to scaling each component separately. Therefore, the quaternion wavelet transform is simply equivalent to the standard wavelet transform on each component independently and combining the transformed result. For the inverse wavelet transform, the same analysis applies.

5.2 Quaternion Wavelet Based Compression

The following images illustrate the multi-stage quaternion wavelet transform of a colour image. Figure 11(a) is the input 256x256 colour image. Figure 11 (b) shows the wavelet transform of the original image according to the Harr transform. Figure 11 (c) is the reconstructed image based on only 64x64 pixels on the top-left corner of the wavelet transform. The reconstruction based on only 1/16 of the original image illustrates a good approximation because the top-left corner contains the most essential low frequency information, while the discarded pixels on the rest of the image only contain high frequency information. Although the implementation of quaternion

Harr wavelet transform produces the same results as wavelet transform of RGB channels independently, it introduces a new technique for image processing. The wavelet transform contains both colour and structural information, which makes it interesting to develop applications to find colour edges from the wavelet transform that contains multi-resolution details of the input image [2].



(a) (b) (c)

Figure 10: Illustration of the multi-stage quaternion wavelet transform of the colour image. The image on the left is the input 256x256 image; the one is the middle is a wavelet transform of the original image; the image on the right is the compressed image constructed based on the most top-left 64x64 pixels corner of the wavelet transform. (Image copyright © CorelGallery database by permission)

6. QUATERNION SVD/PCA

6.1 Quaternion Singular Value Decomposition

The singular value decomposition is an importance technique in linear algebra. It plays an interesting, fundamental role in many different applications, for instance, in digital image processing. Based on the SVD technique, the Principle Components Analysis (PCA), also known as Karhunen-Loève expansion or Eigen-XY analysis, has found a number of applications in the fields of computer vision and pattern recognition.

The singular value decomposition of a matrix factors an m x n matrix A into the form

$$A = U\Sigma V^T$$

where U is an m x m orthogonal matrix; V an n x n orthogonal matrix, and Σ an m x n matrix containing the singular values of A with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$ along its main diagonal.

The following section reviews the quaternion Principle Component Analysis based on Bihan and Pei's work[14][21}. The implementation of QPCA is based on QSVD, the quaternion singular value decomposition can be considered as a generalization of real or complex number singular value decomposition, and inherits similar properties. Many papers have proposed how to compute the eigen-values of a quaternion-valued matrix [7, 14, 34]. Every quaternion matrix can be decomposed into the multiplication of two matrices, Q and R, where Q is a unitary matrix and R is an upper triangular matrix [34]. Therefore, a quaternion matrix can be decomposed into its singular value form by converting it into complex matrix representation. Simple applications of QSVD have been demonstrated in colour image compression in [14] and [21].

However, as discussed in [34], only the right eigen-values and eigen-vectors of a quaternion-valued matrix are defined. So, in this thesis only the right side singular value decomposition is studied. Each quaternion matrix has an equivalent complex matrix. The relations (isomorphism: C <=> Q, where Q represents a quaternion number and C represents its complex equivalence) between the quaternion matrix and its equivalent complex matrix can be found in [14]. Therefore, the existing complex SVD algorithm can be applied to this equivalent complex matrix to obtain the eigenvectors and singular values of the corresponding quaternion matrix.

- **Theorem:** *Existence of the SVD of a quaternion matrix* [34]

Let Q_q be a n-by-n quaternion valued matrix with rank m, then there exist two unitary quaternion matrices U_q and V_q such that

$$U_q^H \cdot Q_q \cdot V_q = \begin{bmatrix} \Lambda_r & 0 \\ 0 & 0 \end{bmatrix}_{n \times n} \tag{46}$$

where $\Lambda_r = diag\{\lambda_1, ..., \lambda_m\}$ *with* $1 \le m \le n$. $\lambda's$ are real positive singular values of Q_q. Unitary quaternion matrices U_q and V_q have the property that $U_q \cdot U_q^H = V_q \cdot V_q^H = I_r$, so the multiplication of quaternion matrices actually yields the real identity matrix I_r. (*ie.* all imagery components of $U_q \cdot U_q^H$ and $V_q \cdot V_q^H$ are zero). We can re-write Equation(46) as

$$Q_q = U_q \cdot \begin{bmatrix} \Lambda_r & 0 \\ 0 & 0 \end{bmatrix}_{n \times n} \cdot V_q^H \tag{47}$$

(^{H} represents the Hermitian transpose operator, or conjugate-transposition operator)

- **Definition:** *Equivalent complex matrix of a quaternion matrix* [34]

Each quaternion $q = w + x \cdot i + y \cdot j + z \cdot k$ can be decomposed into a form as complex numbers, i.e. $q = (w + x \cdot i) + (y + z \cdot i) \cdot j = a + b \cdot j$, where a and b are two complex numbers.

Suppose we have an nxn quaternion valued matrix Q_q, with $Q_q = A_c + B_c \cdot j$, where A and B are nxn complex matrices, then the equivalent 2nx2n complex matrix C_c of Q_q is

$$C_c = \begin{bmatrix} A_c & B_c \\ -\overline{B}_c & \overline{A}_c \end{bmatrix}_{2n\times 2n} \tag{48}$$

Hence, the classical complex SVD algorithm can be applied to C_c to generate the eigen-basis (eigenvectors) and corresponding singular values ranked in descending order (all singular values are real numbers).

- **Theorem**: *The relations between the SVD of a quaternion matrix and the SVD of its equivalent complex matrix [14]. Let the SVD of a quaternion matrix and its equivalent complex matrix be $Q_q = U_q \cdot \Lambda \cdot V_q^H$ and $C_c = U_c \cdot \Lambda' \cdot V_c^H$, respectively, then*

1) $\Lambda = row_{odd}(col_{odd}(\Lambda'))$, *and*

2) if $U_c = \begin{bmatrix} [U_c^1]_{n\times 2n} \\ [U_c^2]_{n\times 2n} \end{bmatrix}_{2n\times 2n} = \begin{bmatrix} A_c \\ B_c \end{bmatrix}$, *then* $\begin{aligned} U_q &= col_{odd}(U_c^1) + col_{odd}(-\overline{U}_c^2) \cdot j \\ &= col_{odd}(A_c) + col_{odd}(-\overline{B}_c) \cdot j \end{aligned}$

$row_{odd}(M)$ and $col_{odd}(M)$ means the odd rows and odd columns of matrix M, respectively.

Some of the most significant properties of the QSVD, when applied to colour images, are listed below by Sangwine [14]:

- Invariance to spatial rotation(also true in the case of greyscale images with SVD)
- Invariance to spatial shift (vectors in U and V are shifted by the same amount)
- Invariance to colour space rotation

6.2 Comparison of QSVD and SVD

Since we have both the standard SVD for real number and QSVD, it would be interesting if we can compare the performance of them applied to the same data. More specifically, the distribution of the singular values implies information distribution of the basis. Here, an experiment is demonstrated to compare the distribution of singular values for both methods. However, this experiment is only preliminary. More sophisticated experiments may be conducted in future work.

The input data consists of 100 50-by-50 colour patterns randomly selected from the patterns database. For QSVD, each image is reshaped to form a 2500x1 quaternion-valued vector. With 100 images, the input matrix for the QSVD algorithm is 2500x100 dimensional (note that each entry is a quaternion with 4 components). For the standard SVD, since each image has three channels, by reshaping each channel to a 2500x1 vector of real numbers, the three channels are appended to form a 7500x1 vector. With 100 images, the input matrix for standard SVD is then 7500x100 dimensional. After decomposition, their ranked singular value distributions are illustrated in Figure 11. In order to better observe the distribution difference around the first several singular values, we take logarithm on both the x and y directions in the two plots . In this experiment, it has been shown in the plots that the first several singular values based on QPCA decrease faster than those based on standard PCA. This could be an advantage of QPCA over PCA because it implies that fewer basis vectors are needed to represent the whole data set in some applications such as image compression. However, in this experiment, the memory space required for each basis vector of the QPCA is 2500x4 real numbers, while the memory space for each basis of standard PCA is only 7500 real numbers. Therefore, in order for QPCA to do a better compression, the ratio of the bases required for the two methods has to be smaller than ¾. If we use the first 3 bases of QPCA and first 4 bases of standard PCA, QPCA can do a 2%~5% better compression than PCA with the same memory requirement. The measure of the compression is based on the total errors in all pixels between the reconstructed image and the original image.

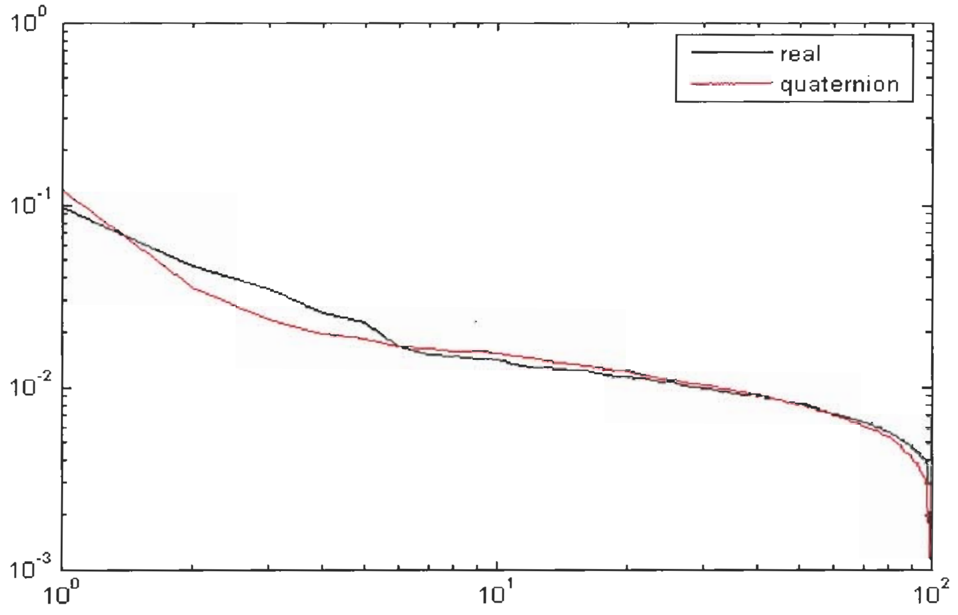


Figure 11: Plots of the singular value distributions of standard PCA(in red) and QPCA(in black). x-axis is the rank of each singular value. y-axis is the value. The x-axis and y-axis are the log-scaled such that the different between the first a few singular values are more obvious. That is, the singular values obtained by QPCA decreases faster than standard PCA.

6.3 QSVD-Based Colour Image Compression

Based on SVD of a colour image, many useful image processing methods by SVD can be extended to a colour images without separating the colour image into three channels. Here, we will introduce some useful colour image processing applications. The beauty of the SVD within its digital applications is that it provides a robust method of storing large images as smaller, more manageable ones. This is accomplished by reproducing the original image with each succeeding nonzero singular value. Furthermore, to reduce storage size even further, one may approximate a "good enough" image using even fewer singular values. If we use quaternion-valued matrix Q to represent a colour image, where each pixel is represented by a pure quaternion, then by QSVD, the image can be can decomposed as

$$Q = U \cdot \Lambda \cdot V^H$$

where U and V can be computed by the methods in the previous section.

Eigen-Images :

Similar to the SVD of a grey image, the SVD of a colour image Q can be decomposed into the summation of vector outer

$$Q = U \cdot \Lambda \cdot V^H = \sum_{i=1}^{R} \lambda_i \cdot (u_i \times v_i^H) \tag{49}$$

where u_i and v_i are the column vectors of matrix U and V, respectively. Suppose λi's are the diagonal terms of real matrix Λ(i.e. the singular value), and R is the rank of Q. Every product $u_i \times v_i^H$ generates an eigen-image. Hence, the colour image f can be considered as the linear combination of R colour eigen-images. Similar to the complex matrix, the preceding eigen-images represent the low-frequency components of the original image, and the later ones represent the high-frequency components.

The singular values distribution in Figure 11 shows the same phenomenon as in the conventional complex case, that the singular values decay very fast. Hence, an approximate of a colour image can be obtained by summing the first k eigen-images:

$$Q_{approx} = \sum_{i=1}^{k} \lambda_i \cdot (u_i \times v_i^H) \tag{50}$$

The real part of Q_{approx} is small and will decrease to zero when K increases to R. In general, even small K can provide a good approximation of the original colour image. Consequently, the storage requirements for this colour image drop from 3xNxN to K(2*4N+1). (including K real singular values; 2K*N quaternion vectors). Figure 12 illustrates four estimated images based on

QPCA. In Figure 12(a)-(c), the images are reconstructed with K=3, 16, 50, 255(the perfect reconstruction). Usually, when the image has more high frequency components, greater K is necessary to have a good approximation. On the other hand, for images containing mostly low frequency signals, the performance of this reconstruction is good with small K. By comparing the two approximations in Figure 12(c) and (d), it is clear that a "good enough" representation may be found with far fewer singular values. This substantially reduces the amount of information necessary to store the exact image. This is a real life example of the power and

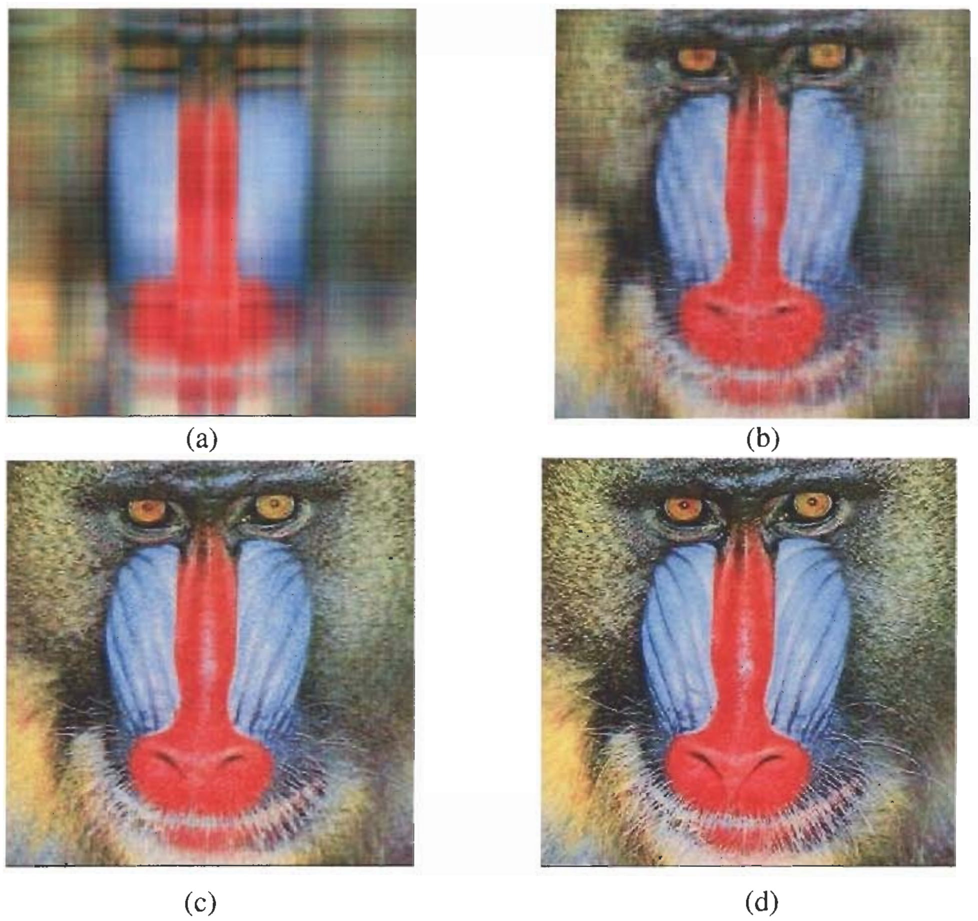


Figure 12: QPCA based image compression.
(a) –(d) are the reconstructed images with k=3,16,50,255. Note that (d) is the perfect reconstruction of the original image (Image copyright © CorelGallery)

efficiency of the singular value decomposition. Let us look at the numbers involved. Because the original image before the SVD was 255x255, it requires 255x255x3=195075 entries for storage.

The image produced by the first 16 singular values requires only 16*255*4 entries; the image with the first 50 requires 50*255*4. This drastically reduces the information necessary–to 8.4% and 26% of the original image, respectively.

6.4 QPCA-Based Colour Texture Segmentation

6.4.1 Introduction

In the computer vision and image processing literature, a wide variety of approaches to extract texture features from image neighbourhoods have been proposed to characterize texture patterns. A texture feature extraction method is a process applied to every pixel of a given image to generate a feature presentation within the texture pattern to which that pixel and its neighbours belong. The performance of different topologies of texture methods depends on the type of operations required, the neighbouring pixels involved, and the texture content.

Due to the large amount of texture features, one of the most difficult problems is how to select an appropriate set of texture features that represent the most significant characteristics of the patterns for classification. Therefore, finding a feature vector that has the greatest discrimination power has been a widely researched topic in the field of pattern analysis and texture classification. For colour imagery, Hoang et. al.[12] have shown that using colour and texture in combination results in better discrimination than using the colour and texture features separately. The extracted feature that combines both colour and texture information has attracted the most interest recently.

In this chapter, we propose a colour texture segmentation method that makes possible encoding the structural and colour feature as a whole unit by using the quaternion represented colour. Quaternion colour provides a new way to treat colour and texture in combination. Here we use quaternion principal component analysis (QPCA) with colours encoded as quaternions to calculate a basis for colour texture, and obtain good experimental results in classifying colour textures in real images.

The quaternion colour texture analysis we propose proceeds in several stages. In the first stage (feature extraction stage) of classification, the feature vector is generated by reducing the dimensionality of the raw feature, which consists of the colours of all neighbouring pixels around the centre pixel to be processed, using quaternion Principle Component Analysis (QPCA). The feature vector that represents a texture patch can have high dimensions, resulting in high computational cost. The purpose to use PCA is that it can reduce the dimensionality of the feature vector such that the classification can be done in an efficient and effective manner, by projecting the feature vector onto the most important basis. For colour textures, by the means of QPCA, such dimensionality reduction can be done without separating the RGB layers. Instead, each RGB triple is treated as a single unit represented by a quaternion. This extracted feature from a colour texture patch then contains both texture and colour information. In the second stage (texture segmentation stage), the features extracted in the previous stage are used to segment the image into regions of pixels that have similar colour texture patterns. It can be shown that the features generated by using our method can be used for multi-level segmentation by selecting a certain threshold.

6.4.2 Methodology

In this section, we propose the algorithm for the estimation of colour texture for segmentation. A segmentation algorithm is used to illustrate the performance of the proposed feature estimation. Given an MxN colour image, we can construct a quaternion matrix Q that contains in each column a quaternion vector composed of spatial neighbours of size W^2 of each pixel. Thus the size of Q is $(W^2)\times(MN)$. This matrix, on which we can apply the analysis, can be interpreted as containing in each column a representation of random variables in a colour image. For a 256 by 256 colour image, with window size 15 by15, the number of feature vectors is 65536, and the size of each original (un-reduced) feature vector is 225. Since each element is represented by a 4-dimensional quaternion, the total dimension of the input is 255x65536x4, which is apparently infeasible for the QSVD algorithm and is the reason we need to down-sample the input image as a training set for the training stage above.

Feature Extraction

The first stage is to extract an orthogonal basis for the colour textures in an image. This basis is calculated by sampling n-by-n sub-windows from the image and expressing the contents of each sub-window as a vector of quaternions of length n^2 and arranging these vectors as the columns of a matrix. QPCA applied to this matrix yields an orthogonal basis for the contents of the sub-windows ordered in terms of the variance accounted for by each basis vector. As with standard PCA, the dimensionality of the feature space can then be reduced by selecting just the first few bases that account for the majority of the variance. These basis vectors are vectors of quaternions. The contents of any image window can then be approximated concisely by projecting them onto the reduced basis. The projection can then be used as the color texture feature of the window.

Due to the computational cost, we first train our classifier on a relatively small amount of variables from the input image. Suppose the appropriate size of the texture feature evaluation window that is large enough to cover significant features of a kind of texture is W, we subdivide the input image into a number of blocks whose size is WxW. These blocks are used to construct the training data T_q, a quaternion matrix, with each column representing a single texture that is reshaped to a quaternion vector v_q of size W^2, the so-called feature vector. By applying the QSVD algorithm, T_q is then decomposed into a product of eigenvectors and singular values in decreasing magnitude as in Equation(47). Each column of eigenvectors U_q represents a basis function of the transformation. To reduce the size of feature vector down to K with $K \leq W^2$, we first obtain U_q^K by cropping U_q to the first K vectors, and multiply it to the training matrix T_q. In this experiment, we choose K=1 (ie. the first basis only). Therefore, a quaternion feature vector v_q of size W^2 is reduced down to size 1, a single quaternion number(Note that a quaternion has 4 dimensions. A quaternion vector v_q of size n has 4n dimensions). The whole training set T_q is reduced to T_q^1, where $T_q^K = U_q^K \cdot T_q$. Generally, a large K allows a more accurate feature measure as v_q^K contains more features. However, a small K value makes the measurement more stable, since with large K noise and non-regular textures may cause "outlier" feature elements which often cause problems in classification steps. In other words, a less reduced feature vector may contain too much information. On the other hand, with a small K, the reduced feature vector is smaller and leads to more efficient performance.

Clustering and Classification

The second stage of the quaternion colour texture analysis is texture clustering. The features obtained for each sub-window are used to cluster the image textures into groups. While the texture features could be a vector of quaternions, in practice it turned out that the best texture feature was the projection onto only the first QPCA basis vector. As a result, each texture is simply represented by a quaternion. The k-means algorithm was used for clustering based on the 4 components of the quaternion describing each window. K-means generates *k* centroids describing the mean features of *k* texture clusters. Each image pixel is then classified according to its texture by comparing the quaternion resulting from a window centred on it to the *k* centroids to find which it is closest to.

Suppose T_q^K is the reduced feature vector matrix with K equal to 1, the classification algorithm is based on clustering the textures in the training set using their associated reduced feature vectors T_q^1. These 4 dimensional feature vectors (each column of T_q^1) are fed into the clustering algorithm as input. The k-means algorithm is chosen as the clustering method, and is applied to the feature space to generate the initial clusters (they will be merged later), with large *k*. The output of the k-means clustering is *k* centroids describing the mean features of the texture clusters.

Now we can classify every pixel of the input image based on the set of centroids of the clusters derived from the training step. So, for each pixel in the image, we take a WxW window centered at this pixel as an input variable. We reshape each window to a vector and project it onto the basis U_q^1 to reduce this feature vector to v_q^1. Each feature vector is averaged with the neighbouring feature vectors to suppress the variation in the local color-texture region. Then we identify the cluster to which v_q^1 is closest based on the Euclidean distance to all centroids. This texture cluster is hence the class that the pixel belongs to.

Texture Segmentation

The final stage is texture segmentation. The initial k-means clustering uses a large k. After clustering, regions are segmented based on their texture by iteratively merging statistically similar regions.

Remember we use the k-means clustering with a large number k. After clustering, a merging method is required to combine statistically similar adjacent regions to achieve the expected number of clusters. At each iteration, the two most similar clusters are merged according to the region similarity measure in [16]:

$$S_{i,j} = (\mu_i - \mu_j)^T [\Phi_i + \Phi_j]^{-1} (\mu_i - \mu_j) \tag{51}$$

where μ_i and μ_j are the mean vectors(centroids), and Φ_i, Φ_j are the covariance matrices derived from the feature vectors of regions R_i, R_j. Smaller $S_{i,j}$ implies greater similarity between two clusters. Hence, at each iteration, the two regions with the smallest $S_{i,j}$ are merged until the threshold is reached. Finally, due to the fact that the sample window may cover more than one region, a post-processing step is necessary to remove these cross-boundary regions. If in a small neighbourhood around a pixel includes 3 or more different texture labels then the pixel in the middle is assigned to the nearest of the other two regions.

6.4.3 Experiments

The results of segmentation using the QPCA based feature extraction method are shown below. In Figure 13(a), the input image is composed by five colour textures with different colour or patterns. The top-left and bottom-left subimages are chosen to have similar colour and texture but different orientations, while the left top and right top subimages have exactly the same pattern but different colour. Our method successfully discriminates the five regions. By selecting different thresholds(loosen the threshold), more regions are merged. Firstly, the top-left and bottom-left textures are merged due to the similarity of colour and pattern. Then the shadow areas in the bottom- right subimage can be removed. For comparison, the segmentation done based on intensity only does not distinguish the colour difference between the top-left and top-right subimages. On the other hand, the segmentation on colour only can not discriminate the left two textures either. Further more, our result has been compared with the results in [Hoang] to illustrate the natural flexibility of this our feature extraction measure against illumination and shadows. More segmentation examples can be seen in Figure 14.

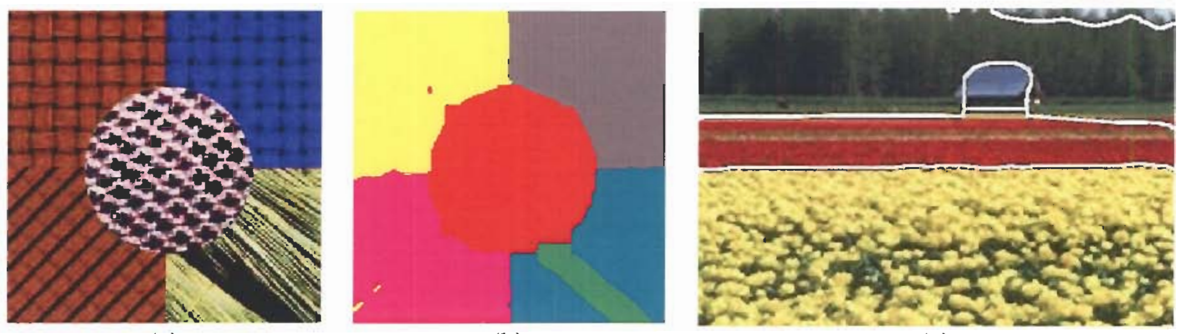
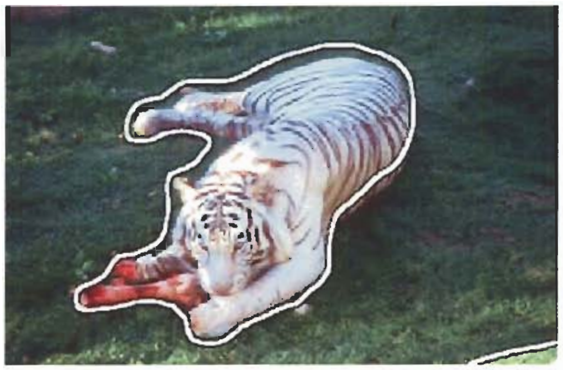
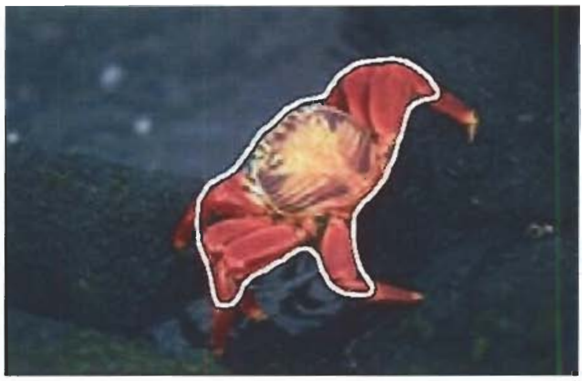

(a) (b) (c)

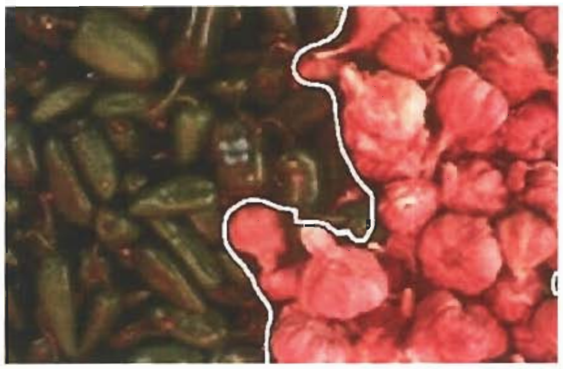
Figure 13: Texture segmentation results (colour images reproduced here in grayscale) (a) synthetic input image with brown regions on the top left and bottom left, a blue region in the top right, a pale green region in the bottom right and a grayish circular region in the center [25](Image copyright © J. M. Geusebroek by permission). (b) Segmentation result for (a) showing that the QPCA method successfully separates the top left and top right regions which differ in colour but have similar (although rotated) grayscale structure, and simultaneously separates the top-left and bottom-left regions which differ in grayscale structure but have similar colour. The shadow in the lower right region is identified. The result in (b) is similar to that in [25] (page 272, figure 3(d)) without shadow invariance. (c) Result on a natural image (from the Corel database) in which regions of yellow flowers (lower section), red flowers (middle), a gray barn roof, green grass/trees, and blue sky are each segmented. (Parameters used were: image size 192x128, window size 17, abutting windows, d=1, initial k-means clusters 15, similarity threshold 5.0, Gaussian smoothing σ=4) (Images copyright © CorelGallery database by permission)



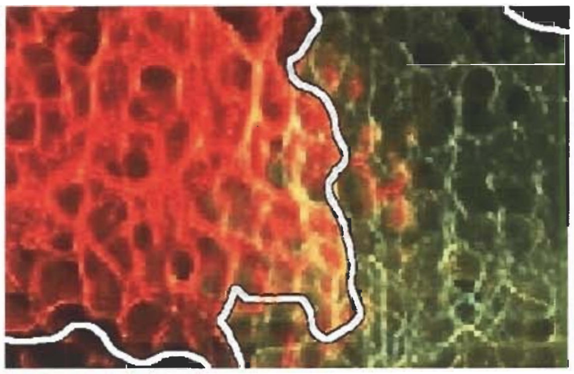
(a)



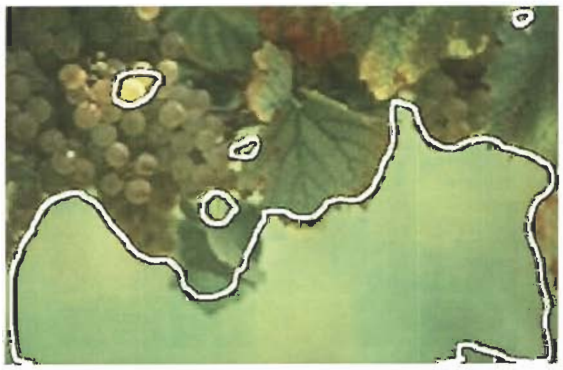
(b)



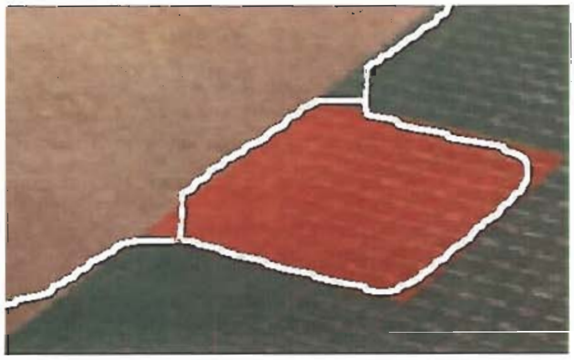
(c)



(d)



(e)



(f)

Figure 14: Result of the QPCA based texture segmentation method applied to natural images. In (d) and (f), regions of similar pattern but different colour are distinguished. In (e), areas of similar colour but different textures are segmented. (Images copyright © CorelGallery database by permission)

7. DISCUSSION AND CONCLUSION

Quaternions have been applied in a new way as a means of representing pixel values in a colour image [25, 26, 28]. When colour images are processed, instead of individually manipulating the three real numbers representing the RGB values of the pixel, the triple is considered as a whole entity and processed as a pure quaternion number. As a generalization of the image analysis techniques on grey images, different quaternion-based analysis have been studied in this thesis based on the corresponding mathematical theories.

7.1 Colour Image Analysis

Several colour image analysis techniques have been introduced by Sangwine and Pei [19, 20, 25, 26, 28] based on the quaternion arithmetic, such as the colour sensitive low pass filtering, colour edge detection. When an image is filtered with a quaternion-valued mask, dot product and cross product between two corresponding pixels are taken as the measure. The colour edge detection is based on the chromaticity cancellation by rotating colours by 180 degree angle to cancel with the un-rotated colours. Based on Sanwine and Pei's methods, I proposed the adaptive colour image smoothing filter, colour edge detection filter and the trilateral filter, The trilateral filtering, inspired by the bilateral filtering for grey-scale image, defines a quaternion-values mask whose shape and colour changes adaptively depending on the local pixels. The trilateral filter is designed to smooth the colour only on the direction parallel to the central pixel, given the neighbouring pixels similar to the central pixel, yet persevering edges. BQMP thresholding technique has been used to cluster a subwindow in an image into two colour representatives, such that the edges between the two regions are enhanced(Suppose the total number of segments in an image is greater than two).

7.2 Fourier Analysis

In Chapter Four we discussed the quaternion Fourier transform and its applications. Similar to grey images, quaternion-valued images can also be transformed into the Fourier domain and can be represented as quaternion frequency signals, based on which different image processing techniques such as filtering can be performed efficiently on the three colour channels. Three types of Fourier transforms were studied – Left-Side QFT, Right-side QFT and Two-sides QFT. Also, for each type of QFT, a Fourier axis needs to be specified in order to perform the QFT algorithm by decomposition. Therefore, for a given colour image, it will have different Fourier transforms according to the type of QFT and the Fourier axis we choose. Filtering in quaternion frequency domain has the advantage that the colour triples are processed as a whole unit rather than dealing with RGB channels separately. We believe more accurate colour information can be persevered this way, since all colour channels are processed as a single unit.

Based on the efficient implementation QFT, it is possible to implement quaternion convolution and cross-correlation in the frequency domain. Pattern matching is an important technique in digital image processing. Correlation based solutions predominantly use a cross correlation to find the potential locations of the template. As an extension of cross-correlation on grey-scale images, quaternion based cross-correlation can be done spatially and it also can be done more efficiently in quaternion Fourier domain. The successful template matching experiments to match colour letters based on quaternion cross-correlation have been shown in this chapter. The experiment result shows all the candidate position of the template in a given image. The matching process encodes the structural and colour information as a unit while measuring the similarity between the template and each individual region in the test image. Additionally, the quaternion filters we have defined in Chapter Two can be transformed into Fourier domain and convolution may be efficiently done there. The quaternion Fourier transform and cross-correlation have proven useful tools for many colour image processing applications.

7.3 Quaternion Principle Component Analysis

QSVD based QPCA has been studied in Chapter Six as an extension of the standard PCA technique in quaternion space. By comparing the distribution of singular values of the standard PCA and QPCA, the advantage of QPCA has been shown in image compression. Moreover, QPCA generates a set of basis vectors where each basis vector is quaternion-valued. Feature vectors extracted based on the most representative basis vectors can be used for texture segmentation and image indexing, as they encode both structural and colour information. I believe that the advantage of using quaternion representation of colour image in texture analysis is that each colour is processed as a unit. In quaternion multiplications, each component of a quaternion interacts with others, and each component of the product of two quaternions can be considered as the weighted sum of all components. Moreover, every element of the feature vector is related to all colour channels (every element of the feature vector is the combination of contribution from all 4 layers), as opposite to processing RGB colour channels independently. Therefore, the feature vector by QPCA is better and more representative for colour texture patterns.

In conclusion, the quaternion number, as a 4D generalization of the complex number, demonstrates its power in different approaches of colour image processing. In general, most image analysis techniques on real or complex numbers can also be extended to quaternions, such as image filtering, Fourier transform and principle component analysis. The value of quaternions representation of colour information is significant as its ability to integrate colour and structure characteristics as an entity. In this thesis, I have tried to show the importance of quaternions concepts as a means to process colour images, which indicates the use of quaternions is non-trivial. Moreover, a large number of theories and applications of quaternions are potentially waiting to be discovered and applied.

MY CONTRIBUTIONS

In Chapter Two

- Design of adaptive colour sensitive smoothing filter, based on Sangwine's colour sensitive smoothing filter that smooth image with respect to a certain colour.
- Design of adaptive colour edge detection filter, based on Sangwine's colour edge detection filter that detects a certain colour edges.
- Proposal of the trilateral filter, based on the bilateral filter and adaptive colour sensitive smoothing filter in combination.

In Chapter Three

- Design of the algorithm for BQMP based Image sharpening.

In Chapter Four

- Experiment on quaternion cross-correlation for template matching in both spatial and frequency domain, based on QFT.

In Chapter Five

- Experiment on quaternion wavelet for image compression.

In Chapter Six

- Design of QPCA-based colour texture segmentation.

References:

[1] Alleysson, D. and Süsstrunk, S, "Spatio-chromatic PCA of a Mosaiced color image," *Proc. IS&T Second European Conference on Color in Graphics, Image, and Vision (CGIV)*, April 2004, pp. 311-314.

[2] Arivazhagan S., Ganesan, L., "Texture segmentation using wavelet transform," *PRL(24)*, No. 16, December 2003, pp. 3197-3203.

[3] Briechle, K., Hanebeck, U. D. , "Template Matching using Fast Normalized Cross Correlation" , *Proceedings of SPIE*, v. 4387, 2001

[4] Burgiss, Samuel G., Goodridge, Steven G., "Multiframe averaging and homomorphic filtering for clarification of dark and shadowed video scenes," *Proc. SPIE*, Vol. 4232, p. 480-488

[5] Caelli, T., McCabe, A., "Complex Images and Complex Filters: A Unified Model for Encoding and Matching Shape and Colour," *ICAPR 2001*, 321-330

[6] Ell, T. A. and Sangwine, S. J., "Decomposition of 2D Hypercomplex Fourier Transforms into Pairs of Complex Fourier Transforms", *Proceedings of EUSIPCO 2000, Tenth European Signal Processing Conference*, 2000, II, 1061-1064.

[7] Ell, T. A. and Sangwine, S. J., "Hypercomplex Wiener-Khintchine Theorem with Application to Color Image Correlation", *IEEE International Conference on Image Processing (ICIP 2000)*, Vancouver, Canada, September 11-14, 2000, II, 792-795.

[8] Ell, T. A., "Quaternion-Fourier transforms for analysis of two-dimensional linear time-invariant partial differential systems," in *Proc. 32nd Conf. Decision Contr.*, Dec. 1993, pp. 1830–1841

[9] Evans, C.J., Ell, T. A. and Sangwine, S. J., 'Hypercomplex Color-Sensitive Smoothing Filters', *ICIP 2000*, Vancouver, Canada, September 11-14, 2000, I, 541-544.

[10] Funt, B., Ciurea, F., McCann, J., "Retinex in Matlab", *Journal of the Electronic Imaging*, pp 48-57, Jan. 2004

[11] Hamilton, W.R., "On quaternions," *Proceeding of the Royal Irish Academy*, Nov. 11, 1844.

[12] Hoang, M. A., Geusebroek, J. M., and Smeulders, A. W. M.. "Color texture measurement and segmentation," Signal Processing, 2005, 85(2):265-275.

[13] Kantor, I.L. and Solodovnikov A.S., *Hypercomplex numbers, an elementary introduction to algebras*, Springer-Verlag, 1989.

[14] Le Bihan, N. and Sangwine, S.J., "Quaternion Principal Component Analysis of Colour images," *IEEE International Conference on Image Processing (ICIP)*, 2003, I, 809-812.

[15] Malonek, H., "Quaternions in Applied Sciences a Historical Perspective of Mathematical Concept ", *Proc. International Kolloquium applications of computer science and mathematics in architecture and building industry*, IKM, 16, 2003

[16] Nguyen, H.T., Worring, M., and Dev, A.. "Detection of moving objects in video using a robust motion similarity measure," *IEEE Trans. on Image Proc.*, 2000, 9(1):137-141.

[17] Olariu S, "Complex numbers in n Dimensions", BOOK Elsevier, xv+269 (2002)

[18] Pei, S. C., Ding, J. J. and Chang, J. H., "Efficient implementation of Quaternion Fourier transform, convolution, and correlation by 2-D complex FFT," *IEEE Trans on Signal Processing*, Vol.49, pp.2783-2797, Nov 2001.

[19] Pei, S.C. and Cheng, C.M., "Color image processing by using binary Quaternion-Moment-Preserving thresholding technique," *IEEE Trans. on Image Processing*, Vol.8, No.5, pp.614-628, May 1999.

[20] Pei, S.C. and Cheng, C.M., "A novel block truncation coding of color images by using quaternion-moment preserving principle," *IEEE International Symposium on Circuits and systems*, Atlanta, GA, 12-15 May, 1996, vol. 2, pp. 684–687.

[21] Pei, S.C., Chang, J.H., Ding, J.J.,"Quaternion matrix singular value decomposition and its applications for color image processing," *ICIP03*, 805-808.

[22] Rönn, S., "Bicomplex algebra and function theory", *ArXiv Mathematics:math*, Jan. 2001

[23] S.J. Moxey E., Sangwine and T.A. Ell, "Hypercomplex operators and vector correlation," *Proceedings of the eleventh European Signal Processing Conference (EUSIPCO)*, Toulouse, France, 2002, vol. III, pp. 247–250.

[24] Sangwine S.J. and Ell T.A., "Hypercomplex auto- and cross-correlation of color images," *IEEE International Conference on Image Processing (ICIP'99)*, Kobe, Japan, 1999, number IV, pp. 319–322.

[25] Sangwine, S. J. and Ell, T. A., "Colour image filters based on hypercomplex convolution", *IEE Proceedings - Vision, Image and Signal Processing*, 147, (2), April 2000, 89-93.

[26] Sangwine, S. J. and Ell, T. A., "Mathematical approaches to linear vector filtering of color images", *CGIV 2002*, University of Poitiers, France, 2-5 April 2002, The Society for Imaging Science and Technology, 348-351.

[27] Sangwine, S. J. and Ell, T. A.,"Hypercomplex Fourier Transforms of Color Images", *IEEE International Conference on Image Processing (ICIP 2001)*, Thessaloniki, Greece, October 7-10, 2001, I, 137-140.

[28] Sangwine, S.J., "Colour image edge detector based on quaternion convolution," *Electronics Letters*, vol. 34, no. 10, pp. 969–971, 14 May 1998.

[29] Sangwine, S.J., "Fourier transforms of colour images using quaternions, or hypercomplex, numbers," *Electronics Letters*, vol. 32, no. 21, pp. 1979–1980, 1996.

[30] Sangwine, S.J., Evans, C.J. and Ell T.A., “Colour-sensitive edge detection using hypercomplex filters,” *Proceedings of the tenth European Signal Processing Conference (EUSIPCO)*, Tampere, Finland, 2000, vol. I, pp. 107–110.

[31] Shi, L. and Funt, B, “Quaternion Color Texture”, *AIC'2005 Proc. 10th Congress of the International Color Association*, May 2005.

[32] Tomasi, C. and Manduchi, R. , "Bilateral Filtering for Gray and Color Images", *Proceedings of the 1998 IEEE International Conference on Computer Vision*, Bombay, India

[33] Yang, C. K., Wu T. C., Lin, J. C. and Tsai, W. H., "Color image sharpening by moment-preserving technique," *Signal Processing*, Vol. 45, pp. 397-403, 1995

[34] Zhang, F., “Quaternions and matrices of quaternions,” *Linear algebra and its applications*, vol. 251, pp. 21–57, 1997.