

# Optimized Fast Algorithms for the Quaternionic Fourier Transform<sup>\*</sup>

Michael Felsberg and Gerald Sommer

Christian-Albrechts-University of Kiel  
Institute of Computer Science and Applied Mathematics  
Cognitive Systems  
Preußenstraße 1–9, 24105 Kiel, Germany  
Tel: +49 431 560433, Fax: +49 431 560481  
{mfe,gs}@ks.informatik.uni-kiel.de

**Abstract.** In this article, we deal with fast algorithms for the quaternionic Fourier transform (QFT). Our aim is to give a guideline for choosing algorithms in practical cases. Hence, we are not only interested in the theoretic complexity but in the real execution time of the implementation of an algorithm. This includes floating point multiplications, additions, index computations and the memory accesses. We mainly consider two cases: the QFT of a real signal and the QFT of a quaternionic signal. For both cases it follows that the row-column method yields very fast algorithms. Additionally, these algorithms are easy to implement since one can fall back on standard algorithms for the fast Fourier transform and the fast Hartley transform. The latter is the optimal choice for real signals since there is no redundancy in the transform. We take advantage of the fact that each complete transform can be converted into another complete transform. In the case of the complex Fourier transform, the Hartley transform, and the QFT, the conversions are of low complexity. Hence, the QFT of a real signal is optimally calculated using the Hartley transform.

## 1 Introduction

In image processing, the complex and the quaternionic Fourier transform are important tools. Hence, it is advantageous to have fast algorithms for these transforms. In the past, many algorithms for the complex Fourier transform have been proposed [1, 9, 4]. Most of them use the radix- $n$  principle. By new algebraic embeddings, the theoretic complexities of the algorithms have been decreased. Mostly, the considerations only deal with the number of floating point multiplications.

In this paper, we mainly present fast algorithms for the quaternionic Fourier transform [6, 2]. We specialize in radix-2 algorithms since we think that higher radices yield unnecessary extensions of the spatial domain (the domain must be a power of  $n$ ). Furthermore, we consider both, the additive and the multiplicative complexity since there is no difference in the execution time nowadays. Our aim is to equip the reader with the necessary basics on choosing the right algorithm in a certain case.

We underline our results by experiments on a real computer. Many algorithms and combinations have been tested, though we have not implemented every algorithm in the same environment.

---

<sup>\*</sup> This work was supported by the DFG (So-320/2-1).

## 2 Basics

In this section we define the *discrete quaternionic Fourier transform* (DQFT), which is based on the continuous transform proposed independently by Ell and Bülow [6, 2]. Furthermore, we state the standard 1-D *fast Fourier transform* (FFT1) and the 1-D *fast Hartley transform* (FHT1). We suppose that the reader is familiar with the algebra of quaternions  $\mathbb{H}$ , the Fourier transform and the Hartley transform.

### 2.1 The Quaternionic Fourier Transform

Let  $\{1, i, j, k\}$  denote the basis of the algebra of quaternions  $\mathbb{H}$ , where we have  $i^2 = j^2 = k^2 = -1$  and  $ij = -ji = k$ . The coefficient-selection operators are defined by  $\text{Re}(q) + i\text{Im}(q) + j\text{Jm}(q) + k\text{Km}(q) = q$  for  $q \in \mathbb{H}$ . The *discrete quaternionic Fourier transform* (QFT) is defined by sampling the QFT

$$F_{u,v}^q = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} e^{-i2\pi um/M} f_{m,n} e^{-j2\pi vn/N} , \quad (1)$$

according to [3].

The *inverse transform* is obtained by changing the signs in the exponential terms and by multiplying an additional normalizing factor  $(MN)^{-1}$ . Except for the different imaginary units, the QFT is calculated in the same way as the complex Fourier transform.

The quaternionic spectrum of a real signal is *quaternionic Hermite symmetric*, i.e. we have

$$F^q(u, -v) = \alpha(F^q(u, v)) \quad \text{and} \quad F^q(-u, v) = \beta(F^q(u, v)) , \quad (2)$$

where  $\alpha(q) = -iqi$  and  $\beta(q) = -jqj$  are two *non-trivial involutions* of the quaternion algebra. Hence, the QFT of a real signal consists of 75% redundant data.

### 2.2 The Fast Fourier Transform and the Fast Hartley Transform

In this paper, we use the *discrete Fourier transform* (DFT) without a normalizing factor. Applying the *decimation of time* method, we obtain the formula for the FFT

$$F_u = F_u^e + e^{-i2\pi u2^{-k}} F_u^o , \quad (3)$$

where  $F, F^e$ , and  $F^o$  denote the spectra of  $f, f^e$ , and  $f^o$ , respectively, and  $f_m^e = f_{2m}$  and  $f_m^o = f_{2m+1}$ .

The Hartley transform [1] is a real valued integral transform. The only formal difference to the complex (inverse) Fourier transform is that the sine-function in the kernel is not multiplied by the imaginary unit.

The *discrete Hartley transform* (DHT) is defined by

$$H_u = \sum_{m=0}^{M-1} f_m \text{cas}(2\pi mu/M) = \sum_{m=0}^{M-1} f_m (\cos(2\pi mu/M) + \sin(2\pi mu/M)) . \quad (4)$$

The shift-theorem of the DHT reads [1]

$$\sum_{m=0}^M f_{m-a} \text{cas}(2\pi mu/M) = H_u \cos(2\pi au/M) + H_{M-u} \sin(2\pi au/M) . \quad (5)$$

Consequently, the decimation of time method yields a formula, which is different from (3):

$$H_u = H_u^e + H_u^o \cos(2\pi u 2^{-k}) + H_u^o \sin(2\pi u 2^{-k}) , \quad (6)$$

where  $H$ ,  $H^e$ , and  $H^o$  denote the Hartley transforms of  $f$ ,  $f^e$ , and  $f^o$ , respectively, and  $f_m^e = f_{2m}$  and  $f_m^o = f_{2m+1}$ .

### 3 Fast Algorithms for 2-D Signals

In this section, we describe the *row-column method*, which yields a fast algorithm for an  $n$ -D transform using 1-D transforms. Another approach for developing fast algorithms is the 2-D *decimation method* which divides the signal into four parts [8]. Finally, the mappings between the Hartley transform, the QFT and the complex spectrum of a 2-D signal are given.

#### 3.1 The Row-Column Method

Consider a 2-D integral transform whose kernel is separable. Such a transform can be calculated by applying 1-D transforms on each coordinate. Without loss of generality, we firstly apply the transform with respect to the  $x$ -coordinate and secondly wrt. the  $y$ -coordinate.

Since the kernels of the complex 2-D Fourier transform and the QFT are separable, we can state the *row-column algorithm* of FFT1s (RC-FFT1).

1. The 2-D Fourier transform is separable. Therefore, the complex Fourier transform of a 2-D signal  $f_{m,n}$  can be calculated by applying the 1-D FFT to each column of  $\hat{f}_{u,n}$ . Thereby, the signal  $\hat{f}_{u,n}$  is obtained by the application of the 1-D FFT to each row of  $f_{m,n}$ . Hence, the spectrum can be calculated by  $M + N$  1-D transforms.
2. The QFT is separable, too.

$$\begin{aligned} F_{u,v}^q &= \sum_{n=0}^{N-1} \left( \underbrace{\sum_{m=0}^{M-1} e^{-i2\pi um/M} (\text{Re}(f_{m,n}) + i\text{Im}(f_{m,n}))}_{f_{u,n}^r} \right. \\ &\quad \left. + \underbrace{\sum_{m=0}^{M-1} e^{-i2\pi um/M} (\text{Im}(f_{m,n}) + i\text{Re}(f_{m,n})) j}_{f_{u,n}^k} \right) e^{-j2\pi vn/N} \\ &= \sum_{n=0}^{N-1} \underbrace{(\text{Re}(f_{u,n}^r) + \text{Re}(f_{u,n}^k)j)}_{f_{u,n}^R} e^{-j2\pi vn/N} + i \sum_{n=0}^{N-1} \underbrace{(\text{Im}(f_{u,n}^r) + \text{Im}(f_{u,n}^k)j)}_{f_{u,n}^I} e^{-j2\pi vn/N} . \quad (7) \end{aligned}$$

Therefore, the QFT of a 2-D signal  $f_{m,n}$  can be calculated by applying the 1-D FFT to each column of both  $\hat{f}_{u,n}^R$  and  $\hat{f}_{u,n}^I$ . Hence, the QFT can be calculated by  $2M + 2N$  1-D FFTs.

*Note 1.* The Hartley kernel is *not* separable, since

$$\text{cas}(2\pi(um + vn)) \neq \text{cas}(2\pi um)\text{cas}(2\pi vn) . \quad (8)$$

Nevertheless, the application of the row-column method to FHT1s (RC-FHT1) yields an interesting transform, since the DQFT of a real signal can be synthesized from the RC-FHT1 (see Sect. 3.3). The only difference between the 2-D DHT and the result of the RC-FHT1 is the sign of the component, which is odd wrt. both coordinates.

### 3.2 The 2-D Decimation Method

Fast algorithms for the 2-D transforms can be developed by dividing the spatial domain with respect to *both* coordinates:

$$f_{m,n}^{ee} = f_{2m,2n} \quad f_{m,n}^{oe} = f_{2m+1,2n} \quad f_{m,n}^{eo} = f_{2m,2n+1} \quad f_{m,n}^{oo} = f_{2m+1,2n+1} . \quad (9)$$

Note that the signal must be of quadratic shape  $2^k \times 2^k = N \times N$ . The (quaternionic) Fourier transforms of  $f^{ee}$ ,  $f^{oe}$ ,  $f^{eo}$ , and  $f^{oo}$  are denoted by  $F^{ee}$ ,  $F^{oe}$ ,  $F^{eo}$ , and  $F^{oo}$ , respectively. The corresponding Hartley transforms are denoted by  $H^{ee}$ ,  $H^{oe}$ ,  $H^{eo}$ , and  $H^{oo}$ . Using this notation, we can state the following equations:

$$F_{u,v} = F_{u,v}^{ee} + F_{u,v}^{oe} e^{-i2\pi u/N} + F_{u,v}^{eo} e^{-i2\pi v/N} + F_{u,v}^{oo} e^{-i2\pi(u+v)/N} \quad (10)$$

$$F_{u,v}^q = F_{u,v}^{ee} + e^{-i2\pi u/N} F_{u,v}^{oe} + F_{u,v}^{eo} e^{-j2\pi v/N} + e^{-i2\pi u/N} F_{u,v}^{oo} e^{-i2\pi v/N} \quad (11)$$

$$H_{u,v} = H_{u,v}^{ee} + H_{u,v}^{eo} \cos(2\pi v/N) + H_{u,\bar{v}}^{eo} \sin(2\pi v/N) + H_{u,v}^{oe} \cos(2\pi u/N) \\ + H_{u,v}^{oe} \sin(2\pi u/N) + H_{u,v}^{oo} \cos(2\pi(u+v)/N) + H_{u,\bar{v}}^{oo} \sin(2\pi(u+v)/N) \quad (12)$$

where  $N = 2^k \geq 2$ ,  $\bar{u} = N - u$  and  $\bar{v} = N - v$ .

*Note 2.* We cannot apply the  $n$ -D decimation method to the *Clifford Fourier transform* (CFT), since this method requires a commutative algebra if  $n > 2$ . The CFT is the generalization of the QFT for higher dimensions [2]. This problem concerning the decimation can be solved by embedding the transform into a different, commutative algebra [7].

### 3.3 The Mappings between the Spectra

Since we can convert each complete transform into each other, we have six mappings which describe the relation between the DHT, the DQFT and the DFT of a real signal. We define four operators, which yield the even and odd part of a 2-D signal wrt. the  $x$ - and the  $y$ -coordinate

$$\begin{aligned} \text{Ee}(f(x, y)) &= \frac{1}{4}(f(x, y) + f(-x, y) + f(x, -y) + f(-x, -y)) \\ \text{Eo}(f(x, y)) &= \frac{1}{4}(f(x, y) + f(-x, y) - f(x, -y) - f(-x, -y)) \\ \text{Oe}(f(x, y)) &= \frac{1}{4}(f(x, y) - f(-x, y) + f(x, -y) - f(-x, -y)) \\ \text{Oo}(f(x, y)) &= \frac{1}{4}(f(x, y) - f(-x, y) - f(x, -y) + f(-x, -y)) . \end{aligned} \quad (13)$$

Using this operators, we obtain the following Table 1.

Note that the row-column method applied to 1-D FHTs yields a transform which differs from the 2-D DHT wrt.  $O_o(H)$  only. Let  $\hat{H}$  denote the transform which is calculated by the RC-FHT1. Then,  $O_o(H) = -O_o(\hat{H})$ . Hence,  $\hat{H}$  is a complete representation and we can obtain the DQFT by applying the RC-FHT1 algorithm

transform	relation to DHT
$F =$	$Ee(H) - iEo(H) - iOe(H) + Oo(H)$
$F^q =$	$Ee(H) - jEo(H) - iOe(H) - kOo(H)$
relation to DFT	
$H =$	$Ee(F) + iEo(F) + iOe(F) + Oo(F)$
$F^q =$	$Ee(F) + kEo(F) + Oe(F) - kOo(F)$
relation to DQFT	
$H =$	$Ee(F^q) + jEo(F^q) + iOe(F^q) + kOo(F^q)$
$F =$	$Ee(F^q) - kEo(F^q) + Oe(F^q) + kOo(F^q)$

**Table 1:** Relations between the transforms

$$F^q = Ee(\hat{H}) - jEo(\hat{H}) - iOe(\hat{H}) + kOo(\hat{H}) . \quad (14)$$

## 4 Complexities

In this section, we consider the complexities of the presented algorithms, in order to decide which one is the fastest. Since the DFT and the DQFT of a real signal contain *redundancy*, we present one approach which reduces the redundancy in order to speed up the algorithm. Besides the theoretic complexities, we consider the execution time of real implementations.

### 4.1 Optimizations

First of all, we have one effect which can be used to speed up *any* of the presented transforms. We know that a phase-shift by  $\pi$  yields a change of sign. Due to this fact, we can calculate the values at the frequencies  $u$  and  $u + N/2$  using the same addends, e.g.

$$\begin{pmatrix} F_u \\ F_{u+N/2} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} F_u^e \\ e^{-i2\pi u/N} F_u^o \end{pmatrix} . \quad (15)$$

Consequently, the number of multiplications for each  $n$ -D fast transform is divided by  $2^n$ . Furthermore, we decrease the number of additions for the FHT since we have

$$\begin{pmatrix} H_u \\ H_{u+N/2} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} H_u^e \\ H_u^o \cos(2\pi u/N) + H_{N-u}^o \sin(2\pi u/N) \end{pmatrix} \quad (16)$$

in the 1-D case.

Due to Hermite symmetry, we have a redundancy of 50% for the DFT and of 75% for the DQFT. We can use the Hermite symmetry directly by copying the data and changing the signs in the imaginary parts. This method yields a

lower complexity of arithmetic operations, but increases the number of memory accesses. Therefore, it is advantageous to use an implicit method, called *overlapping* [4].

The idea of the latter is to create a new, complex signal  $\bar{f} = f^e + if^o$  of the length  $N/2$ . Since the DFT is linear, the spectra  $F^e$  and  $F^o$  can be extracted from the spectrum  $\bar{F}$  of  $\bar{f}$  ( $\bar{F} = F^e + iF^o$ ). Hence, the complexity is divided by an asymptotic factor of two. For the FQFT, we can apply the same approach, i.e. we have  $f = f^{ee} + if^{oe} + jf^{eo} + kf^{oo}$ . This increases the asymptotic complexity by four.

## 4.2 Evaluation of the Complexities

Since modern processors evaluate floating point multiplications as fast as additions [11, 5, 12], we do not only consider the number of floating point multiplications, but also the number of additions and the total number of floating point operations (flops). Since we can easily convert one spectrum to another, it is not crucial for the estimation of the complexity, if the calculated spectrum is real, complex or quaternionic. Nevertheless, our aim is to develop the fastest way to calculate the DQFT.

In the Table 2, the flops of the presented 1-D fast algorithms can be found.

Every algorithm uses the effect of the  $\pi$ -phase, and the FFT1 algorithm for real signals uses overlapping. For the sake of clarity we suppose that overlapping halves the complexity. Note that this is *not* true for finite signal lengths. The complexity

	transform	multiplications	additions	flops
	FHT1	$N \log_2 N$	$\frac{3}{2}N \log_2 N$	$\frac{5}{2}N \log_2 N$
	FFT1 (IR)	$N \log_2 N$	$\frac{3}{2}N \log_2 N$	$\frac{5}{2}N \log_2 N$
	FFT1 (C)	$2N \log_2 N$	$3N \log_2 N$	$5N \log_2 N$

**Table 2:** Complexities of the presented 1-D algorithms

is reduced by a factor *less* than two in this case.

In the Table 3, the flops of the presented 2-D decimation algorithms and the row-column algorithms can be found. Every algorithm uses the effect of the  $\pi$ -phase, where the matrix of combinations of the signs can be separated [9]:

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix}. \quad (17)$$

The separation decreases the number of additions needed for the four quadrants by two thirds. Furthermore, the FQFT algorithm for real signals uses overlapping. For the sake of clarity we provide that overlapping quarters the complexity. Again, this is *not*

	transform	multiplications	additions	flops
	RC-FHT1	$2N^2 \log_2 N$	$3N^2 \log_2 N$	$5N^2 \log_2 N$
	RC-FFT1 (IR)	$3N^2 \log_2 N$	$\frac{9}{2}N^2 \log_2 N$	$\frac{15}{2}N^2 \log_2 N$
	RC-FFT1 (IH)	$8N^2 \log_2 N$	$12N^2 \log_2 N$	$20N^2 \log_2 N$
	FHT2	$\frac{3}{2}N^2 \log_2 N$	$\frac{11}{4}N^2 \log_2 N$	$\frac{17}{4}N^2 \log_2 N$
	FFT2 (IH)	$6N^2 \log_2 N$	$11N^2 \log_2 N$	$17N^2 \log_2 N$
	FQFT (IR)	$2N^2 \log_2 N$	$2N^2 \log_2 N$	$4N^2 \log_2 N$
	FQFT (IH)	$8N^2 \log_2 N$	$8N^2 \log_2 N$	$16N^2 \log_2 N$

**Table 3:** Complexities of the presented 2-D algorithms

true for finite signal sizes.

Note that the RC-FFT1 (R) algorithm uses three overlapping FFT1s since in (7)  $f^{jk} = 0$  and both  $\hat{f}^R$  and  $\hat{f}^I$  are real valued. The FFT2 (R) is omitted since overlapping cannot be applied in this case, unless the DQFT is used [4].

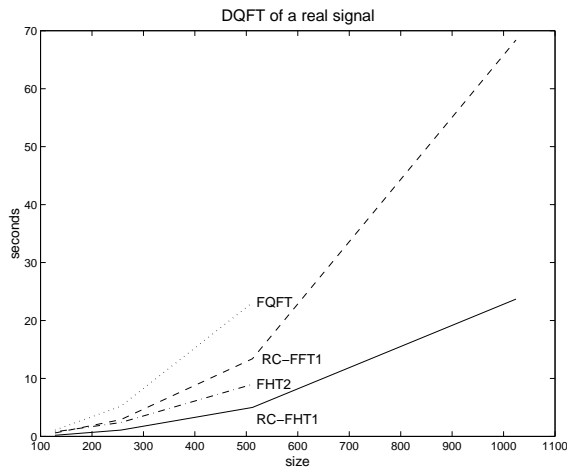
### 4.3 Comparison of the Algorithms

According to Table 2, the fastest algorithm for calculating the 1-D Fourier transform of a real signal is the FHT1 or the FFT1 with overlapping. Since the overlapping yields a high linear complexity, we prefer the FHT1.

In a real implementation, the time for the memory access cannot be neglected. Therefore, the FHT1 and the FFT1 with overlapping are not twice as fast as the FFT1 without overlapping. Our implementation of the FHT1 takes less than two thirds of the execution time of the FFT1 from the numerical recipes [10], but it takes more than one half of the time (e.g. 5.0 s versus 8.9 s).

For complex signals, the FFT1 is the fastest algorithm. The execution of two FHT1s takes more execution time than one FFT1.

The quaternionic Fourier transform of a 2-D signal can be calculated in many ways. According to Table 3, we have the following ranking of algorithms for real signals: FQFT, FHT2, RC-FHT1, RC-FFT1. Note that for the execution time we have another ranking. The overlapping in the FQFT algorithm yields a high quadratic complexity. Hence, for realistic sizes the Hartley transforms have lower complexities. In real implementations, the RC-FHT1 algorithm takes the least execution time, since the memory access of the FHT2 is more complicated. Depending on the implementation, we have the execution time ranking RC-FHT1, FHT2, RC-FFT1, FQFT (Fig. 1).



**Fig. 1:** Execution time for the estimation of the DQFT on a SPARCstation 10

For quaternionic signals, we have the following ranking according to Table 3: FQFT, FFT2, RC-FFT1. Though the FFT2 should be faster than the RC-FFT1 (as well for the evaluation of the DQFT as for the DFT), most implementations of the FFT2 are RC-FFT1 algorithms [10]. Depending on the implementation, we have the execution time ranking RC-FFT1, FFT2<sup>1</sup>, FQFT<sup>2</sup>.

For both cases of the DQFT (real and quaternionic signal), the row-column method has the advantage that both coordinates are independently extended to a power of two. For the 2-D decimation algorithms, the signal must in addition be quadratic.

<sup>1</sup> We provide that the FFT2 is implemented by FFT1s.

<sup>2</sup> The memory access is distributed over the whole data, which yields a slower execution.

## 5 Conclusion

We have presented different algorithms for evaluating the DQFT of real and quaternionic signals. The comparison of these algorithms, their complexities and their execution times yield some characterizing properties:

- We have algorithms which are based on well known transforms (e.g. on the FFT1) and we have some which are new (FQFT).
- Some algorithms are modular (row-column) and some are monolithic.
- The implementation can yield a simple and short code (FFT1) or a complicated and extensive code (FQFT with overlapping).
- The algebraic representation can be simple (FHT1) or complicated (FQFT).

In our experiments we ascertained that those algorithms which can be implemented most easily and which are modular have the shortest execution time though their theoretic complexity is not the lowest. Obviously, the implementations are faster, if they use simple algebraic representations. On the other hand, the FHT1 (6) has more addends than the FFT1 (3). Hence, the FFT1 is the optimal transform for signals which are not real valued.

Since we can construct a very fast algorithm using the standard FFT1, there is only little programming effort to obtain an implementation which evaluates the DQFT. For real signals, the FHT1 should be used instead of the FFT1, which yields a shorter execution time and less programming effort than an implementation of overlapping.

## References

- [1] R. N. Bracewell. *The Fourier transform and its applications*. McGraw Hill, 1986.
- [2] T. Bülow and G. Sommer. Algebraically Extended Representation of Multi-Dimensional Signals. In *Proceedings of the 10th Scandinavian Conference on Image Analysis*, pages 559–566, 1997.
- [3] T. Bülow and G. Sommer. Multi-Dimensional Signal Processing Using an Algebraically Extended Signal Representation. In G. Sommer and J.J. Koenderink, editors, *Int'l Workshop on Algebraic Frames for the Perception-Action Cycle, AF-PAC'97, Kiel*, volume 1315 of *LNCS*, pages 148–163. Springer, 1997.
- [4] V. M. Chernov. Discrete orthogonal transforms with data representation in composition algebras. In *Proceedings of the 9th Scandinavian Conference on Image Analysis*, pages 357–364, 1995.
- [5] Digital Equipment Corporation. Digital Semiconductor Alpha 21164PC Microprocessor Data Sheet<sup>3</sup>, 1997.
- [6] T. A. Ell. *Hypercomplex Spectral Transformations*. PhD thesis, University of Minnesota, 1992.
- [7] M. Felsberg. Signal Processing Using Frequency Domain Methods in Clifford Algebra<sup>4</sup>. Master's thesis, Christian-Albrechts-University of Kiel, 1998.
- [8] M. Felsberg et al. Fast Algorithms of Hypercomplex Fourier Transforms. In G. Sommer, editor, *Geometric Computing with Clifford Algebra*, Springer Series in Information Sciences. Springer, Berlin, 1999. to appear.
- [9] B. Jähne. *Digitale Bildverarbeitung*. Springer, Berlin, 1997.
- [10] W. Press et al. *Numerical Recipes in C*. Cambridge University Press, 1994.
- [11] Silicon Graphics, Inc. MIPS RISC Technology R10000 Microprocessor Technical Brief<sup>5</sup>, 1998.
- [12] Sun Microsystems, Inc. UltraSPARC-II Data Sheet<sup>6</sup>, 1998.

<sup>3</sup> <http://ftp.digital.com/pub/DECinfo/semiconductor/literature/164pcds.pdf>

<sup>4</sup> <http://www.ks.informatik.uni-kiel.de/~mfe/research.html>

<sup>5</sup> [http://www.sgi.com/processors/r10k/tech\\_info/Tech\\_Brief.html](http://www.sgi.com/processors/r10k/tech_info/Tech_Brief.html)

<sup>6</sup> <http://www.sun.com/microelectronics/datasheets/stp1031/index2.html>